Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

T. Aura
Aalto University
M. Sethi
Ericsson

October 22, 2018

Nimble out-of-band authentication for EAP (EAP-NOOB) draft-aura-eap-noob-04

Abstract

Extensible Authentication Protocol (EAP) provides support for multiple authentication methods. This document defines the EAP-NOOB authentication method for nimble out-of-band (OOB) authentication and key derivation. This EAP method is intended for bootstrapping all kinds of Internet-of-Things (IoT) devices that have a minimal user interface and no pre-configured authentication credentials. The method makes use of a user-assisted one-directional OOB channel between the peer device and authentication server.

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 April 25, 2019.

Copyright Notice

Table of Contents

1. Introduction

This document describes a method for registration, authentication and key derivation for network-connected ubiquitous computing devices, such as consumer and enterprise appliances that are part of the Internet of Things (IoT). These devices may be off-the-shelf hardware that is sold and distributed without any prior registration or credential-provisioning process. Thus, the device registration in a server database, ownership of the device, and the authentication credentials for both network access and application-level security must all be established at the time of the device deployment. Furthermore, many such devices have only limited user interfaces that could be used for their configuration. Often, the interfaces are limited to either only input (e.g. camera) or output (e.g. display screen). The device configuration is made more challenging by the fact that the devices may exist in large numbers and may have to be deployed or re-configured nimbly based on user needs.

More specifically, the devices may have the following characteristics:

- no pre-established relation with a specific server or user,
- no pre-provisioned device identifier or authentication credentials,
- limited user interface and configuration capabilities.

Many proprietary OOB configuration methods exits for specific IoT devices. The goal of this specification is to provide an open standard and a generic protocol for bootstrapping the security of network-connected appliances, such as displays, printers, speakers, and cameras. The security bootstrapping in this specification makes

use of a user-assisted out-of-band (OOB) channel. The security is based on the assumption that attackers are not able to observe or modify the messages conveyed through the OOB channel. We follow the common approach of performing a Diffie-Hellman key exchange over the insecure network and authenticating the established key with the help of the OOB channel in order to prevent impersonation and man-in-the-middle (MitM) attacks.

The solution presented here is intended for devices that have either an input or output interface, such as a camera or display screen, which is able to send or receive dynamically generated messages of tens of bytes in length. Naturally, this solution may not be appropriate for very small sensors or actuators that have no user interface at all. We also assume that the OOB channel is at least partly automated (e.g. camera scanning a bar code) and, thus, there is no need to absolutely minimize the length of the data transferred through the OOB channel. This differs, for example, from Bluetooth simple pairing [BluetoothPairing], where it is critical to minimize the length of the manually transferred or compared codes. Since the OOB messages are dynamically generated, we do not support static printed registration codes. This also prevents attacks where a static secret code would be leaked.

2. 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].

In addition, this document frequently uses the following terms as they have been defined in [RFC5216]:

authenticator The entity initiating EAP authentication.

peer The entity that responds to the authenticator. In [IEEE-802.1X], this entity is known as the supplicant.

server The entity that terminates the EAP authentication method with the peer. In the case where no backend authentication server is used, the EAP server is part of the authenticator. In the case where the authenticator operates in pass-through mode, the EAP server is located on the backend authentication server.

3. EAP-NOOB protocol

This section defines the EAP-NOOB protocol. The protocol is a generalized version of the original idea presented by Sethi et al. [Sethi14].

3.1. Protocol overview

One EAP-NOOB protocol execution spans multiple EAP conversations, called Exchanges. This is necessary to leave time for the OOB message to be delivered, as will be explained below.

The overall protocol starts with the Initial Exchange, in which the server allocates an identifier to the peer, and the server and peer negotiate the protocol version and cryptosuite (i.e. cryptographic algorithm suite), exchange nonces and perform an Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) key exchange. The user-assisted OOB Step then takes place. This step requires only one out-of-band message either from the peer to the server or from the server to the peer. While waiting for the OOB Step action, the peer MAY probe the server by reconnecting to it with EAP-NOOB. If the OOB Step has already taken place, the probe leads to the Completion Exchange, which completes the mutual authentication and key confirmation. On the other hand, if the OOB Step has not yet taken place, the probe leads to the Waiting Exchange, and the peer will perform another probe after a server-defined minimum waiting time. The Initial Exchange and Waiting Exchange always end in EAP-Failure, while the Completion Exchange may result in EAP-Success. Once the peer and server have performed a successful Completion Exchange, both endpoints store the created association in persistent storage, and the OOB Step is not repeated. Thereafter, creation of new temporal keys, ECDHE rekeying, and updates of cryptographic algorithms can be achieved with the Reconnect Exchange.

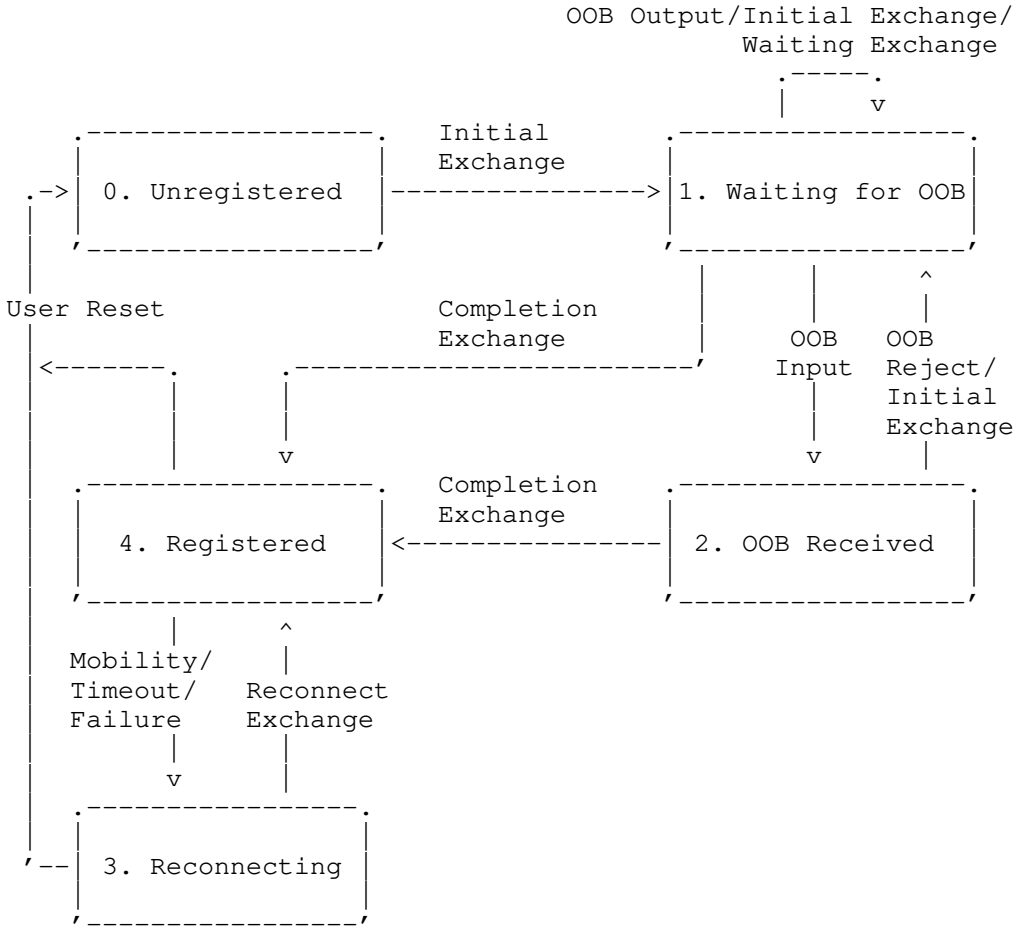

Figure 1: EAP-NOOB server-peer association state machine

Figure 1 shows the association state machine, which is the same for the server and for the peer. (For readability, only the main state transitions are shown. The complete table of transitions can be found in Appendix A.) When the peer initiates the EAP-NOOB method, the server chooses the ensuing message exchange based on the combination of the server and peer states. The EAP server and peer are initially in the Unregistered state, in which no state information needs to be stored. Before a successful Completion Exchange, the server-peer association state is ephemeral in both the server and peer (ephemeral states 0..2), and either endpoint may cause the protocol to fall back to the Initial Exchange. After the Completion Exchange has resulted in EAP-Success, the association

state becomes persistent (persistent states 3..4). Only user reset or memory failure can cause the return of the server or the peer from the persistent states to the ephemeral states and to the Initial Exchange.

The server MUST NOT repeat a successful OOB Step with the same peer except if the association with the peer is explicitly reset by the user or lost due to failure of the persistent storage in the server. More specifically, once the association has entered the Registered state, the server MUST NOT delete the association or go back to states 0..2 without explicit user approval. Similarly, the peer MUST NOT repeat the OOB Step unless the user explicitly deletes the association with the server or resets the peer to the Unregistered state. The server and peer MAY implement user reset of the association by deleting the state data from that endpoint. If an endpoint continues to store data about the association after the user reset, its behavior SHOULD be equivalent to having deleted the association data.

It can happen that the peer accidentally or through user reset loses its persistent state and reconnects to the server without a previously allocated peer identifier. In that case, the server MUST treat the peer as a new peer. The server MAY use auxiliary information, such as the PeerInfo field received in the Initial Exchange, to detect multiple associations with the same peer. However, it MUST NOT delete or merge redundant associations without user or application approval because EAP-NOOB internally has no secure way of verifying that the two peers are the same physical device. Similarly, the server might lose the association state because of a memory failure or user reset. In that case, the only way to recover is that the user resets also the peer.

A special feature of the EAP-NOOB method is that the server is not assumed to have any a-priori knowledge of the peer. Therefore, the peer initially uses the generic identity string "noob@eap-noob.net" as its network access identifier (NAI). The server then allocates a server-specific identifier to the peer. The generic NAI serves two purposes: firstly, it tells the server that the peer supports and expects the EAP-NOOB method and, secondly, it allows routing of the EAP-NOOB sessions to a specific authentication server in the AAA architecture.

EAP-NOOB is an unusual EAP method in that the peer has to have multiple EAP conversations with the server before it can receive EAP-Success. The reason is that, while EAP allows delays between the request-response pairs, e.g. for repeated password entry, the user delays in OOB authentication can be much longer than in password trials. In particular, EAP-NOOB supports also peers with no input

capability in the user interface. Since these output-only devices cannot be told to perform the protocol at the right moment, they have to perform the initial exchange opportunistically and hope for the OOB Step to take place within a timeout period (NoobTimeout), which is why the timeout needs to be several minutes rather than seconds. For example, consider a printer (peer) that outputs the OOB message on paper, which is then scanned for the server. To support such high-latency OOB channels, the peer and server perform the Initial Exchange in one EAP conversation, then allow time for the OOB message to be delivered, and later perform the Waiting and Completion Exchanges in different EAP conversations.

3.2. Protocol messages and sequences

This section defines the EAP-NOOB exchanges, which correspond to EAP conversations. The protocol messages in each exchange and the data members in each message are listed in the diagrams below.

Each EAP-NOOB exchange begins with the authenticator sending an EAP-Request/Identity packet to the peer. From this point on, the EAP conversation occurs between the server and the peer, and the authenticator acts as a pass-through device. The peer responds to the authenticator with an EAP-Response/Identity packet, containing the network access identifier (NAI), which conveys both the peer identity and the current peer state as defined in Section 3.3.1.

After receiving the NAI, the server chooses the EAP-NOOB exchange, i.e. the ensuing message sequence, based on the combination of the peer and server states. The peer recognizes the exchange based on the message type field (Type) of the EAP-NOOB request received from the server. The available exchanges are defined in the following subsections. Each exchange comprises one or more EAP requests-response pairs and ends in either EAP-Failure, indicating that authentication is not (yet) successful, or in EAP-Success.

3.2.1. Initial Exchange

Upon receiving the EAP-Response/Identity from the peer, if either the peer or the server is in the Unregistered (0) state and the other is in one of the ephemeral states (0..2), the server chooses the Initial Exchange.

The Initial Exchange comprises two EAP-NOOB request-response pairs, one for version, cryptosuite and parameter negotiation and the other for the ECDHE key exchange. The first EAP-NOOB request (Type=1) from the server contains a newly allocated PeerId for the peer and an optional Realm. The server allocates a new PeerId in the Initial Exchange regardless of any old PeerId in the username part of the

received NAI. The server also sends in the request a list of the protocol versions (Vers) and cryptosuites (Cryptosuites) it supports, an indicator of the OOB channel directions it supports (Dirs), and a ServerInfo object. The peer chooses one of the versions and cryptosuites. The peer sends a response (Type=1) with the selected protocol version (Verp), the received PeerId, the selected cryptosuite (Cryptosuitep), an indicator of the OOB channel directions selected by the peer (Dirp), and a PeerInfo object. In the second EAP-NOOB request and response (Type=2), the server and peer exchange the public components of their ECDHE keys and nonces (PKs,Ns,PKp,Np). The ECDHE keys MUST be based on the negotiated cryptosuite. The Initial Exchange ends with EAP-Failure from the server because the authentication cannot yet be completed.

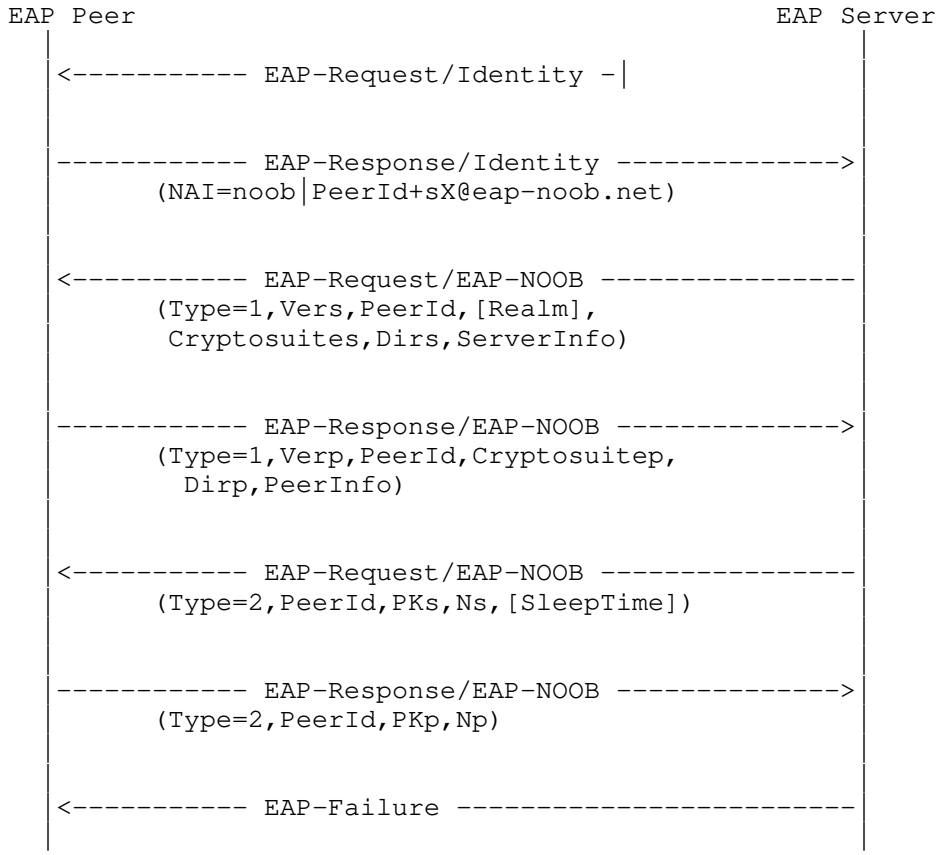

Figure 2: Initial Exchange

At the conclusion of the Initial Exchange, both the server and the peer move to the Waiting for OOB (1) state.

3.2.2. OOB Step

The OOB Step, labeled as OOB Output and OOB Input in Figure 1, takes place after the Initial Exchange. Depending on the direction negotiated, the peer or the server outputs the OOB message shown in Figure 3 or Figure 4. The data fields are the PeerId, the secret nonce Noob, and the cryptographic fingerprint Hoob. The contents of the data fields are defined in Section 3.3.2. The OOB message is delivered to the other endpoint via a user-assisted OOB channel.

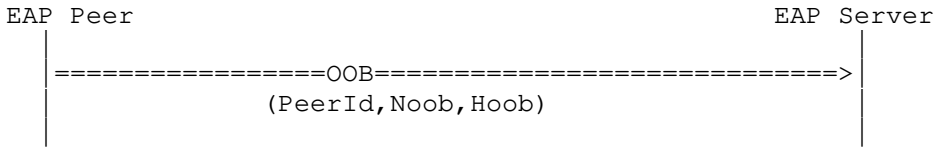

Figure 3: OOB Step, from peer to EAP server

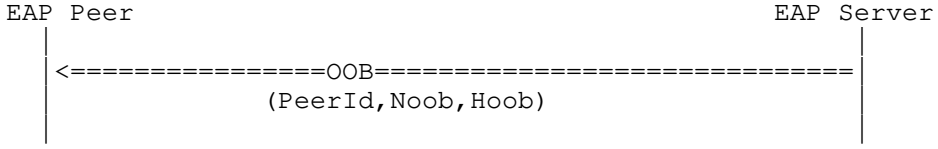

Figure 4: OOB Step, from EAP server to peer

The receiver of the OOB message MUST compare the received value of the fingerprint Hoob with a value that it computes locally. If the values are equal, the receiver moves to the OOB Received (2) state. Otherwise, the receiver MUST reject OOB message. For usability reasons, the receiver SHOULD indicate the acceptance or rejection of the OOB message to the user. The receiver SHOULD reject invalid OOB messages without changing its state, until an application-specific number of invalid messages (OobRetries) has been reached, after which the receiver SHOULD consider it an error and go back to the Unregistered (0) state.

The server or peer MAY send multiple OOB messages with different Noob values while in the Waiting for OOB (1) state. The OOB sender SHOULD remember the Noob values until they expire and accept any one of them in the following Completion Exchange. The Noob values sent by the

server expire after an application-dependent timeout (NoobTimeout), and the server MUST NOT accept Noob values older than that in the Completion Exchange. The RECOMMENDED value for NoobTimeout is 3600 seconds if there are no application-specific reasons for making it shorter or longer. The Noob values sent by the peer expire as defined in Section 3.2.4.

The OOB receiver does not accept further OOB messages after it has accepted one and moved to the OOB Received (2) state. However, the receiver MAY buffer redundant OOB messages in case OOB message expiry or similar error detected in the Completion Exchange causes it to return to the Waiting for OOB (1) state. It is RECOMMENED that the OOB receiver notifies the user about redundant OOB messages, but it MAY also discard them silently.

The sender will typically generate a new Noob, and therefore a new OOB message, at constant intervals (NoobInterval). The RECOMMENDED interval is NoobInterval = NoobTimeout / 2, so that the two latest values are always accepted. However, the timing of the Noob generation may also be based on user interaction or on implementation considerations.

Even though not recommended (see Section 3.3), this specification allows both directions to be negotiated (Dirp=3) for the OOB channel. In that case, both sides SHOULD output the OOB message, and it is up to the user to deliver one of them.

The details of the OOB channel implementation including the message encoding are defined by the application. Appendix E gives an example of how the OOB message can be encoded as a URL that may be embedded in a QR code and NFC tag.

3.2.3. Completion Exchange

After the Initial Exchange, if both the server and the peer support the peer-to-server direction for the OOB channel, the peer SHOULD initiate the EAP-NOOB method again after an applications-specific waiting time in order to probe for completion of the OOB Step. Also, if both sides support the server-to-peer direction of the OOB exchange and the peer receives the OOB message, it SHOULD initiate the EAP-NOOB method immediately. Once the server receives the EAP-Response/Identity, if one of the server and peer is in the OOB Received (2) state and the other is either in the Waiting for OOB (1) or OOB Received (2) state, the OOB Step has taken place and the server SHOULD continue with the Completion Exchange.

The Completion Exchange comprises one or two EAP-NOOB request-response pairs. If the peer is in the Waiting for OOB (1) state, the

OOB message has been sent in the peer-to-server direction. In that case, only one request-response pair (Type=4) takes place. In the request, the server sends the NoobId value, which the peer uses to identify the exact OOB message received by the server. On the other hand, if the peer is in the OOB Received (2) state, the direction of the OOB message is from server to peer. In that case, two request-response pairs (Type=8 and Type=4) are needed. With the first request, the server discovers NoobId, which identifies the exact OOB message received by the peer. The server returns the same NoobId to the peer in the latter request.

In the last and sometimes only request-response pair (Type=4) of the Completion Exchange, the server and peer exchange message authentication codes. Both sides MUST compute the keys Kms and Kmp as defined in Section 3.5 and the message authentication codes MACs and MACp as defined in Section 3.3.2. Both sides MUST compare the received message authentication code with a locally computed value. If the peer finds that it has received the correct value of MACs and the server finds that it has received the correct value of MACp, the Completion Exchange ends in EAP-Success. Otherwise, the endpoint where the comparison fails indicates this with an error message (error code 4001, see Section 3.6.1) and the Completion Exchange ends in EAP-Failure.

After successful Completion Exchange, both the server and the peer move to the Registered (4) state. They also derive the output key material and store the persistent association state as defined in Section 3.4 and Section 3.5.

It is possible that the OOB message expires before it is received. In that case, the sender of the OOB message no longer recognizes the NoobId that it receives in the Completion Exchange. Another reason why the OOB sender might not recognize the NoobId is if the received OOB message was spoofed and contained an attacker-generated Noob value. The recipient of an unrecognized NoobId indicates this with an error message (error code 1006, see Section 3.6.1) and the Completion Exchange ends in EAP-Failure. The recipient of the error message 1006 moves back to the Waiting for OOB (1) state. This state transition is shown as OOB Reject in Figure 1 (even though it really is a specific type of failed Completion Exchange). The sender of the error message, on the other hand, stays in its previous state.

Although it is not expected to occur in practice, poor user interface design could lead to two OOB messages delivered simultaneously, one from the peer to the server and the other from the server to the peer. The server detects this event by observing that both the server and peer are in the OOB Received state (2). In that case, as

a tiebreaker, the server MUST behave as if only the server-to-peer message was delivered.

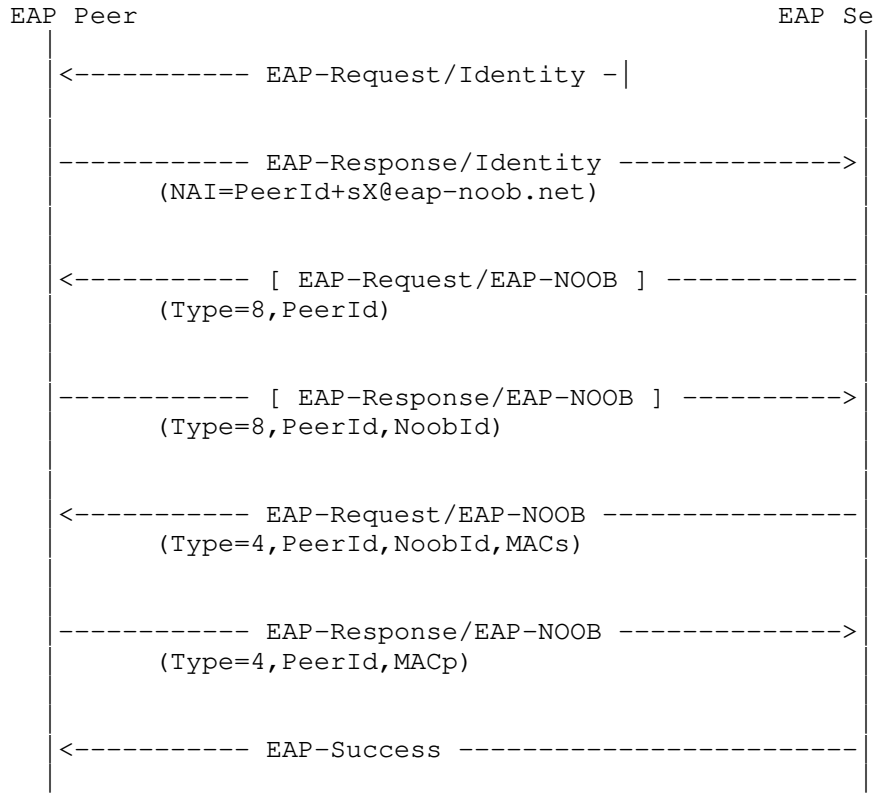

Figure 5: Completion Exchange

3.2.4. Waiting Exchange

As explained in Section 3.2.3, the peer SHOULD probe the server for completion of the OOB Step. If both the server and peer states are Waiting for OOB (1), the server will continue with the Waiting Exchange (Type=3). The main purpose of this exchange is to indicate to the peer that the server has not yet received a peer-to-server OOB message.

In order to limit the rate at which peers probe the server, the server MAY send to the peer either in the Initial Exchange or Waiting Exchange a minimum time to wait before probing the server again. A peer that has not received an OOB message MUST wait at least the

server-specified minimum waiting time in seconds (SleepTime) before initiating EAP again with the same server. The peer uses the latest SleepTime value that it has received in or after the Initial Exchange. If the server has not sent any SleepTime value, the peer SHOULD wait for an application-specified minimum time.

After the Waiting Exchange, the peer MUST discard (from its local ephemeral storage) Noob values that it has sent to the server in OOB messages that are older than the application-defined timeout NoobTimeout (see Section 3.2.2). The peer SHOULD discard such expired Noob values even if the probing failed, e.g. because of failure to connect to the EAP server or incorrect MAC. The timeout of peer-generated Noob values is defined like this in order to allow the peer to probe the server once after it has waited for the server-specified SleepTime.

If the server and peer have negotiated to use only the server-to-peer direction for the OOB channel (Dirp=2), the peer SHOULD nevertheless probe the server. The purpose of this is to keep the server informed about the peers that are still waiting for OOB messages. The server MAY set SleepTime to a high number (3600) to prevent the peer from probing the server frequently.

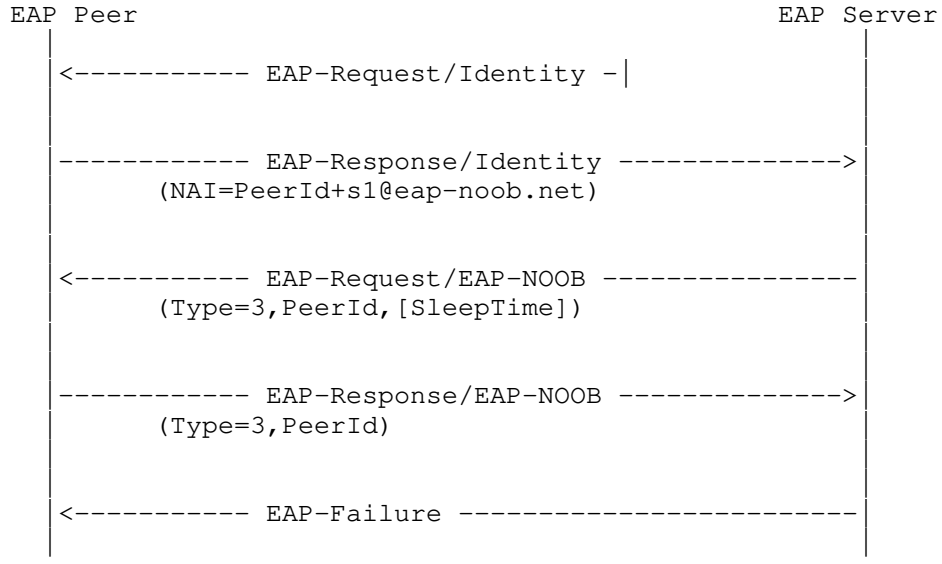

Figure 6: Waiting Exchange

3.3. Protocol data fields

This section defines the various identifiers and data fields used in the EAP-NOOB protocol.

3.3.1. Peer identifier, realm and NAI

The server allocates a new peer identifier (PeerId) for the peer in the Initial Exchange. The peer identifier MUST follow the syntax of the utf8-username specified in [RFC7542]; however, it MUST NOT exceed 60 bytes in length and MUST NOT contain the character '+'. The server MUST generate the identifiers in such a way that they do not repeat and cannot be guessed by the peer or third parties beforehand. One way to generate the identifiers is to choose a random 16-byte identifier and to base64url encode it without padding [RFC4648] into a 22-character string. Another way to generate the identifiers is to choose a random 22-character alphanumeric string. It should be noted that longer the peer identifiers result in longer OOB messages.

When the peer is in one of the states 1..2, the peer MUST use the PeerId that the server assigned to it in the latest Initial Exchange. When the peer is in one of the persistent states 3..4, it MUST use the PeerId from its persistent EAP-NOOB association. When the peer is in the Unregistered (0) state, it MUST use the default value "noob" as its PeerId.

The server MAY also assign a realm (Realm) to the peer by in Initial Exchange or Reconnect Exchange. The realm value MUST follow the syntax of the utf8-realm specified in [RFC7542]. When the peer is in one of the states 1..2, the peer MUST use the Realm that the server assigned to it the latest Initial Exchange, if one was assigned. When the peer is in one of the persistent states 3..4, it MUST use the Realm from its persistent EAP-NOOB association, if one is stored in the association. When the peer is in the Unregistered (0) state, or when the peer is in one of the other states 1..4 and the server has not assigned it a realm, the peer SHOULD use the default realm "eap-noob.net". However, the user or application MAY provide a different default realm to the peer.

To compose its NAI [RFC7542], the peer uses the PeerId in the user part and the Realm as the realm part. When no server-assigned PeerId or Realm is available, the default value is used instead. In the Unregistered (0) state, the peer must create the NAI as the concatenation of the PeerId, the symbol "@", and the Realm. In the other states 1..4, the peer MUST create the NAI as the concatenation of the PeerId, the string "+s", a single digit indicating the state of the peer, and the Realm.

Note that a new peer typically sends the NAI "noob@eap-noob.net" in its first EAP-Response/Identity message, and this default NAI is always acceptable when the peer is in the Unregistered (0) state. The purpose of the state indicator "+sX" is to inform the server about the peer state without the cost of an additional round trip. The server uses this information together with the server state to decide on the type of the exchange and, thus, the type of the next EAP-Request. The lack of a state indicator means that that peer is in state 0.

The purpose of the server-assigned realm is to enable more flexible routing of the EAP sessions over the AAA infrastructure, including roaming scenarios (see Appendix D). The possibility to configure a different default realm further enables registration of new devices while roaming. It also enables manufacturers to set up their own AAA servers for bootstrapping of new peer devices. Moreover, some Authenticators or AAA servers use the assigned Realm to determine peer-specific connection parameters, such as isolating the peer to a specific VLAN.

The peer's PeerId and Realm are ephemeral until a successful Completion Exchange takes place. Thereafter, the values become parts of the persistent EAP-NOOB association, until the user resets the peer and the server or until a new Realm is assigned in the Reconnect Exchange. Resetting the server means deleting the association for the peer from the server database.

3.3.2. Message data fields

Table 1 defines the data fields in the protocol messages. The in-band messages are formatted as JSON objects [RFC8259] in UTF-8 encoding. The JSON member names are in the left-hand column of the table.

Data field	Description
Vers,Verp	EAP-NOOB protocol versions supported by the EAP server, and the protocol version chosen by the peer. Vers is a JSON array of unsigned integers, and Verp is an unsigned integer. Currently, the only defined values are "[1]" and "1", respectively.
PeerId	Peer identifier as defined in Section 3.3.1.
Realm	Peer identifier as defined in Section 3.3.1.

Peer State "+sX"	Peer state indicator as defined in Section 3.3.1.
Type	EAP-NOOB message type. The type is an integer in the range 0..8. EAP-NOOB requests and the corresponding responses share the same type value.
PKs, PKp	The public components of the ECDHE keys of the server and peer. PKs and PKp are sent in the JSON Web Key (JWK) format [RFC7517]. Detailed format of the JWK object is defined by the cryptosuite.
Cryptosuites, Cryptosuitep	The identifiers of cryptosuites supported by the server and of the cryptosuite selected by the peer. The server-supported cryptosuites in Cryptosuites are formatted as a JSON array of the identifier integers. The server MUST send an nonempty array with no repeating elements, ordered by decreasing priority. The peer MUST respond with exactly one suite in the Cryptosuitep value, formatted as an identifier integer. The registration of cryptosuites is specified in Section 4.1.
Dirs, Dirp	The OOB channel directions supported by the server and ones selected by the peer. The possible values are 1=peer-to-server, 2 =server-to-peer, 3=both directions. Endpoints that are general-purpose computers or online services SHOULD support both directions. IoT devices with a limited user interface will mostly support only one direction. If the negotiated value is 3, the user may be presented with two OOB messages, one for each direction, even though the user needs to deliver only one of them. Since this can be confusing to the user, it is RECOMMENDED that the peer selects Dirp value 1 or 2. The EAP-NOOB protocol itself is designed to cope also with selected value 3, in which case it uses the first delivered OOB message. In the unlikely case of simultaneously delivered OOB messages, the protocol prioritizes the server-to-peer direction.
Dir	The actual direction of the OOB message (1

	=peer-to-server, 2=server-to-peer). This value is not sent over any communication channel but it is included in the computation of the cryptographic fingerprint Hoob.
Ns, Np	32-byte nonces for the Initial Exchange.
ServerInfo	This field contains information about the server to be passed from the EAP method to the application layer in the peer. The information is specific to the application or to the OOB channel and it is encoded as a JSON object of at most 500 bytes. It could include, for example, the access-network name and server name or a Uniform Resource Locator (URL) [RFC4266] or some other information that helps the user to deliver the OOB message to the server through the out-of-band channel.
PeerInfo	This field contains information about the peer to be passed from the EAP method to the application layer in the server. The information is specific to the application or to the OOB channel and it is encoded as a JSON object of at most 500 bytes. It could include, for example, the peer brand, model and serial number, which help the user to distinguish between devices and to deliver the OOB message to the correct peer through the out-of-band channel.
SleepTime	The number of seconds for which peer MUST NOT start a new execution of the EAP-NOOB method with the authenticator, unless the peer receives the OOB message or the peer is reset by the user. The server can use this field to limit the rate at which peers probe it. SleepTime is an unsigned integer in the range 0..3600.
Noob	16-byte secret nonce sent through the OOB channel and used for the session key derivation. The endpoint that received the OOB message uses this secret in the Completion Exchange to authenticate the exchanged key to the endpoint that sent the OOB message.
Hoob	16-byte cryptographic fingerprint (i.e. hash

	value) computed from all the parameters exchanged in the Initial Exchange and in the OOB message. Receiving this fingerprint over the OOB channel guarantees the integrity of the key exchange and parameter negotiation. Hence, it authenticates the exchanged key to the endpoint that receives the OOB message.
NoobId	16-byte identifier for the OOB message, computed with a one-way function from the nonce Noob.
MACs, MACp	Message authentication codes for mutual authentication, key confirmation, and integrity check on the exchanged information. The input to the HMAC is defined below, and the key for the HMAC is defined in Section 3.5.
Ns2, Np2	32-byte Nonces for the Reconnect Exchange.
KeyingMode	Interger indicating the key derivation method. 0 in the Completion Exchange, and 1..3 in the Reconnect Exchange.
PKs2, PKp2	The public components of the ECDHE keys of the server and peer for the Reconnect Exchange. PKp2 and PKs2 are sent in the JSON Web Key (JWK) format [RFC7517]. Detailed format of the JWK object is defined by the cryptosuite.
MACs2, MACp2	Message authentication codes for mutual authentication, key confirmation, and integrity check on the Reconnect Exchange. The input to the HMAC is defined below, and the key for the HMAC is defined in Section 3.5.

Table 1: Message data fields

The nonces in the in-band messages (Ns, Np, Ns2, Np2) are 32-byte fresh random byte strings, and the secret nonce Noob is a 16-byte fresh random byte string. All the nonces are generated by the endpoint that sends the message.

The fingerprint Hoob and the identifier NoobId are computed with the hash function specified in the negotiated cryptosuite and truncated

to the 16 leftmost bytes of the output. The message authentication codes (MACs, MACp, MACs2, MACp2) are computed with the HMAC function [RFC2104] based on the same cryptographic hash function and truncated to the 32 leftmost bytes of the output.

The inputs to the hash function for computing the fingerprint Hoob and to the HMAC for computing MACs, MACp, MACs2 and MACp2 are JSON arrays containing a fixed number (16) of elements. The array elements MUST be copied to the array verbatim from the sent and received in-band messages. When the element is a JSON object, its members MUST NOT be reordered or re-encoded, and whitespace MUST NOT be added anywhere in the JSON structure. Implementers should check that their JSON library copies the elements as UTF-8 strings and does not modify then in any way.

The inputs for computing the fingerprint and message authentication codes are the following:

```
Hoob = H(Dir,Vers,Verp,PeerId,Cryptosuites,Dirs,ServerInfo,Cryptos
uitep,Dirp,[Realm],PeerInfo,0,PKs,Ns,PKp,Np,Noob).

NoobId = H("NoobId",Noob).

MACs = HMAC(Kms; 2,Vers,Verp,PeerId,Cryptosuites,Dirs,ServerInfo,C
ryptosuitep,Dirp,[Realm],PeerInfo,0,PKs,Ns,PKp,Np,Noob).

MACp = HMAC(Kmp; 1,Vers,Verp,PeerId,Cryptosuites,Dirs,ServerInfo,C
ryptosuitep,Dirp,[Realm],PeerInfo,0,PKs,Ns,PKp,Np,Noob).

MACs2 = HMAC(Kms2; 2,Vers,Verp,PeerId,Cryptosuites,"",[ServerInfo]
,Cryptosuitep,"",[Realm],[PeerInfo],KeyingMode,[PKs2],Ns2,[PKp2],N
p2,"")

MACp2 = HMAC(Kmp2; 1,Vers,Verp,PeerId,Cryptosuites,"",[ServerInfo]
,Cryptosuitep,"",[Realm],[PeerInfo],KeyingMode,[PKs2],Ns2,[PKp2],N
p2,"")
```

Missing input values are represented by empty strings "" in the array. The values indicated with "" above are always empty strings. Realm is included in the computation of Macs and Macp if it was sent or received in the preceding Initial Exchange. Each of the values in brackets for the computation of Macs2 and Macp2 MUST be included if it was sent or received in the same Reconnect Exchange; otherwise the value is replaced by an empty string "".

The parameter Dir indicates the direction in which the OOB message containing the Noob value is being sent (1=peer-to-server, 2=server-to-peer). This field in needed to prevent the user from accidentally

delivering the OOB message back to its originator in the rare cases where both OOB directions have been negotiated. The keys for the HMACs are defined in Section 3.5.

The nonces (Ns, Np, Ns2, Np2, Noob), hash value (NoobId) and message authentication codes (MACs, MACp, MACs2, MACp2) in the in-band messages and when used as inputs to the hash and message authentication code MUST be base64url encoded [RFC4648]. The values Noob and Hoob in the OOB channel MAY also be base64url encoded, if that is appropriate for the application and the OOB channel. All base64url encoding is done without padding. The base64url encoded values will naturally consume more space than the number of bytes specified above (22-character string for a 16-byte nonce and 43-character string for a 32-byte nonce or message authentication code). In the key derivation in Section 3.5, on the other hand, the unencoded nonces (raw bytes) are used as input to the key derivation function.

The ServerInfo and PeerInfo are JSON objects with UTF-8 encoding. The length of either encoded object as a byte array MUST NOT exceed 500 bytes. The format and semantics of these objects MUST be defined by the application that uses the EAP-NOOB method.

3.4. Fast reconnect and rekeying

EAP-NOOB implements Fast Reconnect ([RFC3748], section 7.2.1) that avoids repeated use of the user-assisted OOB channel.

The rekeying and the Reconnect Exchange may be needed for several reasons. New EAP output values MSK and EMSK may be needed because of mobility or timeout of session keys. Software or hardware failure or user action may also cause the authenticator, EAP server or peer to lose its non-persistent state data. The failure would typically be detected by the peer or authenticator when session keys no longer are accepted by the other endpoint. Change in the supported cryptosuites in the EAP server or peer may also cause the need for a new key exchange. When the EAP server or peer detects any one of these events, it MUST change from the Registered to Reconnecting state. These state transitions are labeled Mobility/Timeout/Failure in Figure 1. The EAP-NOOB method will then perform the Reconnect Exchange next time when EAP is triggered.

3.4.1. Persistent EAP-NOOB association

To enable rekeying, the EAP server and peer store the session state in persistent memory after a successful Completion Exchange. This state data, called "persistent EAP-NOOB association", MUST include at least the data fields shown in Table 2. They are used for

identifying and authenticating the peer in the Reconnect Exchange. When a persistent EAP-NOOB association exists, the EAP server and peer are in the Registered state (4) or Reconnecting state (3), as shown in Figure 1.

Data field	Value	Type
PeerId	Peer identifier allocated by server	UTF-8 string (typically 22 bytes)
Verp	Negotiated protocol version	integer
Cryptosuitep	Negotiated cryptosuite	integer
CryptosuitepPrev (peer only)	Previous cryptosuite	integer
Realm	Optional realm assigned by server (default value is "eap-noob.net")	UTF-8 string
Kz	Persistent key material	32 bytes
KzPrev (peer only)	Previous Kz value	32 bytes

Table 2: Persistent EAP-NOOB association

3.4.2. Reconnect Exchange

The server chooses the Reconnect Exchange when peer is in the Reconnecting (3) state and the server itself is in the Registered (4) or Reconnecting (3) state. The peer MUST NOT initiate EAP-NOOB when the peer is in Registered state.

The Reconnect Exchange comprises three EAP-NOOB request-response pairs, one for cryptosuite and parameter negotiation, another for the nonce and ECDHE key exchange, and the last one for exchanging message authentication codes. In the first request and response (Type=5) the server and peer negotiate a protocol version and cryptosuite in the same way as in the Initial Exchange. The server SHOULD NOT offer and the peer MUST NOT accept protocol versions or cryptosuites that it knows to be weaker than the one currently in the Cryptosuitep field of the persistent EAP-NOOB association. The server SHOULD NOT needlessly change the cryptosuites it offers to the same peer because

peer devices may have limited ability to update their persistent storage. However, if the peer has different values in the Cryptosuitep and CryptosuitepPrev fields, it SHOULD also accept offers that are not weaker than CryptosuitepPrev. Note that Cryptosuitep and CryptosuitePrev from the persistent EAP-NOOB association are only used to support the negotiation as described above; all actual cryptograhic operations use the negotiated cryptosuite. The request and response (Type=5) MAY additionally contain PeerInfo and ServerInfo objects.

The server then determines the KeyingMode (defined in Section 3.5) based on changes in the negotiated cryptosuite and whether it desires to achieve forward secrecy or not. The server SHOULD only select KeyingMode 3 when the negotiated cryptosuite differs from the Cryptosuitep in the server's persistent EAP-NOOB association, although it is technically possible to select this values without changing the cryptosuite. In the second request and response (Type=6), the server informs the peer about the KeyingMode, and the server and peer exchange nonces (Ns2, Np2). When KeyingMode is 2 or 3 (rekeying with ECDHE), they also exchange public components of ECDHE keys (PKs2, PKp2). The server ECDHE key MUST be fresh, i.e. not previously used with the same peer, and the peer ECDHE key SHOULD be fresh, i.e. not previously used.

In the third and final request and response (Type=7), the server and peer exchange message authentication codes. Both sides MUST compute the keys Kms2 and Kmp2 as defined in Section 3.5 and the message authentication codes MACs2 and MACp2 as defined in Section 3.3.2. Both sides MUST compare the received message authentication code with a locally computed value.

The rules by which the peer compares the received MACs2 are non-trivial because, in addition to authenticating the current exchange, MACs2 may confirm the success or failure of a recent cryptosuite upgrade. The peer processes the final request (Type=7) as follows:

1. The peer first compares the received MACs2 value with one it computed using the Kz stored in the persistent EAP-NOOB association. If the received and computed values match, the peer deletes any data stored in the CryptosuitepPrev and KzPrev fields of the persistent EAP-NOOB association. It does this because the received MACs2 confirms that the peer and server share the same Cryptosuitep and Kz, and any previous values must no longer be accepted.

2. If, on the other hand, the peer finds that the received MACs2 value does not match the one it computed locally with Kz, the peer checks whether the KzPrev field in the persistent EAP-NOOB

association stores a key. If it does, the peer repeats the key derivation (Section 3.5) and local MACs2 computation (Section 3.3.2) using KzPrev in place of Kz. If this second computed MACs2 matches the received value, it indicates synchronization failure caused by the loss of the last response (Type=7) in a previously attempted cryptosuite upgrade. In this case, the peer rolls back that upgrade by overwriting Cryptosuitep with CryptosuitepPrev and Kz with KzPrev in the persistent EAP-NOOB association. It also clears the CryptosuitepPrev and KzPrev fields.

3. If the received MACs2 matched one of the locally computed values, the peer proceeds to send the final response (Type=7). The peer also moves to the Registered (4) state. When KeyingMode is 1 or 2, the peer stops here. When KeyingMode is 3, the peer also updates the persistent EAP-NOOB association with the negotiated Cryptosuitep and the newly-derived Kz value. To prepare for possible synchronization failure caused by the loss of the final response (Type=7) during cryptosuite upgrade, the peer copies the old Cryptosuitep and Kz values in the persistent EAP-NOOB association to the CryptosuitepPrev and KzPrev fields.

4. Finally, if the peer finds that the received MACs2 does not match either of the two values that it computed locally (or one if no KzPrev was stored), the peer sends an error message (error code 4001, see Section 3.6.1), which causes the the Reconnect Exchange to end in EAP-Failure.

The server rules for processing the final message are simpler than the peer rules because the server does not store previous keys and it never rolls back a cryptosuite upgrade. Upon receiving the final response (Type=7), the server compares the received value of MACp2 with one it computes locally. If the values match, the Reconnect Exchange ends in EAP-Success. When KeyingMode is 3, the server also updates Cryptosuitep and Kz in the persistent EAP-NOOB association. On the other hand, if the server finds that the values do not match, it sends an error message (error code 4001), and the Reconnect Exchange ends in EAP-Failure.

The endpoints MAY send updated Realm, ServerInfo and PeerInfo objects in the Reconnect Exchange. When there is no update to the values, they SHOULD omit this information from the messages. If the Realm was sent, each side updates Realm in the persistent EAP-NOOB association when moving to the Registered (4) state.

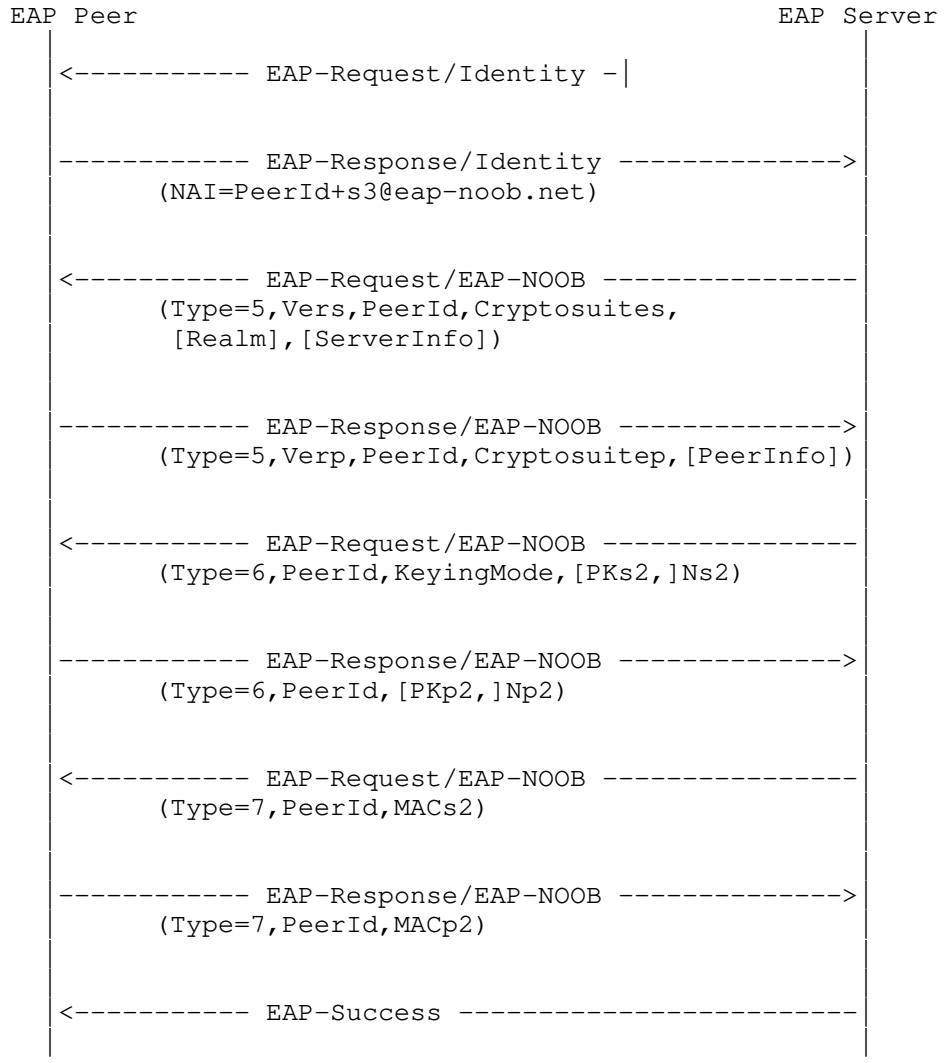

Figure 7: Reconnect Exchange

3.4.3. User reset

As shown in the association state machine in Figure 1, the only specified way for the association to return from the Registered state (4) to the Unregistered state (0) is through user-initiated reset. After the reset, a new OOB message will be needed to establish a new association between the EAP server and peer. Typical situations in which the user reset is required are when the other side has

accidentally lost the persistent EAP-NOOB association data, or when the peer device is decommissioned.

The server could detect that the peer is in the Registered or Reconnecting state but the server itself is in one of the ephemeral states 0..2 (including situations where the server does not recognize the PeerId). In this case, effort should be made to recover the persistent server state, for example, from a backup storage - especially if many peer devices are similarly affected. If that is not possible, the EAP server SHOULD log the error or notify an administrator. The only way to continue from such a situation is by having the user reset the peer device.

On the other hand, if the peer is in any of the ephemeral states 0..2, including the Unregistered state, the server will treat the peer as a new peer device and allocate a new PeerId to it. The PeerInfo can be used by the administrator as a clue to which physical device has lost its state. However, there is no secure way of matching the "new" peer with the old PeerId without repeating the OOB Step. This situation will be resolved when the user performs the OOB Step and, thus, identifies the physical peer device. The server user interface SHOULD support situations where the "new" peer is actually a previously registered peer that has been reset by a user or has otherwise lost the persistent EAP-NOOB association data and needs to be merged with the old peer data in the server.

3.5. Key derivation

EAP-NOOB derives the EAP output values MSK and EMSK and other secret keying material from the output of an Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) algorithm following the NIST specification [NIST-DH]. In NIST terminology, we use a C(2, 0, ECC CDH) scheme, i.e. two ephemeral keys and no static keys. In the Initial and Reconnect Exchanges, the server and peer compute the ECDHE shared secret Z as defined in section 6.1.2.2 of the NIST specification [NIST-DH]. In the Completion and Reconnect Exchanges, the server and peer compute the secret keying material from Z with the single-step key derivation function (KDF) defined in section 5.8.1 of the NIST specification. The hash function H for KDF is taken from the negotiated cryptosuite.

KeyingMode	Description
0	Completion Exchange (always with ECDHE)
1	Reconnect Exchange, rekeying without ECDHE
2	Reconnect Exchange, rekeying with ECHDE, no change in cryptosuite
3	Reconnect Exchange, rekeying with ECDHE, new cryptosuite negotiated

Table 3: Keying modes

The key derivation has three different modes (KeyingMode), which are specified in Table 3. Table 4 defines the inputs to KDF in each KeyingMode.

In the Completion Exchange (KeyingMode=0), the input Z comes from the preceding Initial exchange. KDF takes some additional inputs (OtherInfo), for which we use the concatenation format defined in section 5.8.1.2.1 of the NIST specification [NIST-DH]. OtherInfo consists of the AlgorithmId, PartyUInfo, PartyVInfo, and SuppPrivInfo fields. The first three fields are fixed-length bit strings, and SuppPrivInfo is a variable-length string with a one-byte Datalength counter. AlgorithmId is the fixed-length 8-byte ASCII string "EAP-NOOB". The other input values are the server's and peer's nonces. In the Completion Exchange, the inputs also include the secret nonce Noob from the OOB message.

In the simplest form of the Reconnect Exchange (KeyingMode=1), fresh nonces are exchanged but no ECDHE keys are sent. In this case, input Z to the KDF is replaced with the shared key Kz from the persistent EAP-NOOB association. The result is rekeying without the computational cost of the ECDHE exchange, but also without forward secrecy.

When forward secrecy is desired in the Reconnect Exchange (KeyingMode=2 or KeyingMode=3), both nonces and ECDHE keys are exchanged. Input Z is the fresh shared secret from the ECDHE exchange with PKs2 and PKp2. The inputs also include the shared secret Kz from the persistent EAP-NOOB association.

KeyingMode	KDF input field	Value	Length (bytes)
0 Completion	Z	ECDHE shared secret from PKs and PKp	variable
	AlgorithmId	"EAP-NOOB"	8
	PartyUInfo	Np	32
	PartyVInfo	Ns	32
	SuppPubInfo	(not allowed)	
	SuppPrivInfo	Noob	16
1 Reconnect, rekeying without ECDHE	Z	Kz	32
	AlgorithmId	"EAP-NOOB"	8
	PartyUInfo	Np2	32
	PartyVInfo	Ns2	32
	SuppPubInfo	(not allowed)	
	SuppPrivInfo	(null)	0
2 or 3 Reconnect, rekeying, with ECDHE, same or new cryptosuite	Z	ECDHE shared secret from PKs2 and PKp2	variable
	AlgorithmId	"EAP-NOOB"	8
	PartyUInfo	Np2	32
	PartyVInfo	Ns2	32
	SuppPubInfo	(not allowed)	
	SuppPrivInfo	Kz	32

Table 4: Key derivation input

Table 5 defines how the output bytes of KDF are used. In addition to the EAP output values MSK and EMSK, the server and peer derive another shared secret key AMSK, which MAY be used for application-layer security. Further output bytes are used internally by EAP-NOOB for the message authentication keys (Kms,Kmp,Kms2,Kmp2).

The Completion Exchange (KeyingMode=0) produces the shared secret Kz, which the server and peer store in the persistent EAP-NOOB association. When a new cryptosuite is negotiated in the Reconnect Exchange (KeyingMode=3), it similarly produces a new Kz. In that case, the server and peer update both the cryptosuite and Kz in the persistent EAP-NOOB association. Additionally, the peer stores the previous cryptosuite and Kz values in the CryptosuitepPrev and KzPrev fields of the persistent EAP-NOOB association.

KeyingMode	KDF output bytes	Used as	Length (bytes)
0 Completion	0..63	MSK	64
	64..127	EMSK	64
	128..191	AMSK	64
	192..223	MethodId	32
	224..255	Kms	32
	256..287	Kmp	32
	288..319	Kz	32
1 or 2 Reconnect, rekeying without ECDHE, or with ECDHE and unchanged cryptosuite	0..63	MSK	64
	64..127	EMSK	64
	128..191	AMSK	64
	192..223	MethodId	32
	224..255	Kms2	32
	256..287	Kmp2	32
3 Reconnect, rekeying with ECDHE, new cryptosuite	0..63	MSK	64
	64..127	EMSK	64
	128..191	AMSK	64
	192..223	MethodId	32
	224..255	Kms2	32
	256..287	Kmp2	32
	288..319	Kz	32

Table 5: Key derivation output

Finally, every EAP method must export a Server-Id, Peer-Id and Session-Id [RFC5247]. In EAP-NOOB, the exported Peer-Id is the PeerId which the server has assigned to the peer. The exported Server-Id is a zero-length string (i.e. null string) because EAP-NOOB neither knows nor assigns any server identifier. The exported Session-Id is created by concatenating the Type-Code xxx (TBA) with the MethodId, which is obtained from the KDF output as shown in Table 5.

3.6. Error handling

Various error conditions in EAP-NOOB are handled by sending an error notification message (Type=0) instead of the expected next EAP request or response message. Both the EAP server and the peer may send the error notification, as shown in Figure 8 and Figure 9. After sending or receiving an error notification, the server MUST send an EAP-Failure (as required by [RFC3748] section 4.2). The

notification MAY contain an ErrorInfo field, which is a UTF-8 encoded text string with a maximum length of 500 bytes. It is used for sending descriptive information about the error for logging and debugging purposes.

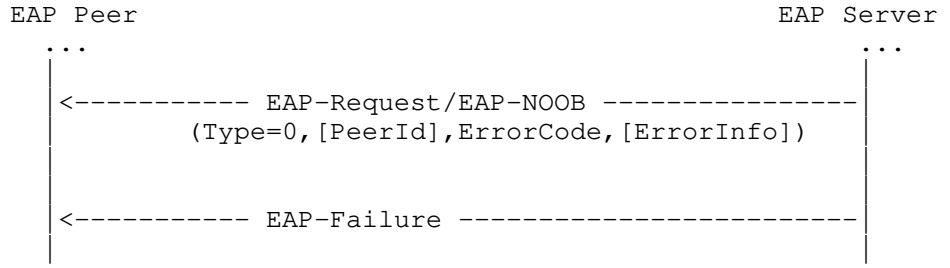

Figure 8: Error notification from server to peer

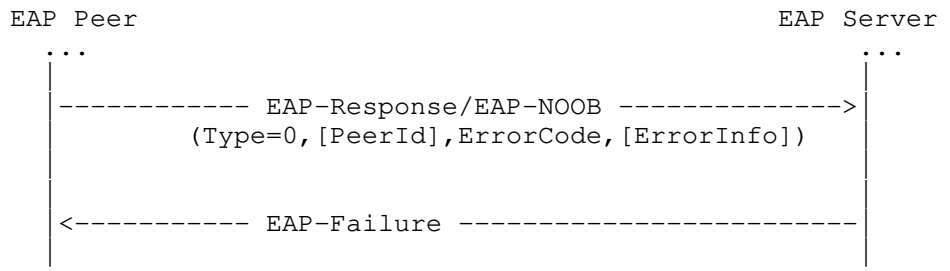

Figure 9: Error notification from peer to server

After the exchange fails due to an error notification, the server and peer set the association state as follows. In the Initial Exchange, both the sender and recipient of the error notification MUST set the association state to the Unregistered (0) state. In the Waiting and Completion Exchanges, each side MUST remain in its old state as if the failed exchange did not take place, with the exception that the recipient of error code 1006 processes it as specified in Section 3.2.3. In the Reconnect Exchange, both sides MUST set the association state to the Reconnecting (3) state.

Errors that occur in the OOB channel are not explicitly notified in-band.

3.6.1. Invalid messages

If the NAI structure is invalid, the server SHOULD send the error code 1001 to the peer. The recipient of an EAP-NOOB request or response SHOULD send the following error codes back to the sender: 1002 if it cannot parse the message as a JSON object or the top-level JSON object has missing or unrecognized members; 1003 if a data field has an invalid value, such as an integer out of range, and there is no more specific error code available; 1004 if the received message type was unexpected; 1005 if the PeerId has an unexpected value; 1006 if the NoobId is not recognized; and 1007 if the ECDHE key is invalid.

3.6.2. Unwanted peer

The preferred way for the EAP server to rate limit EAP-NOOB connections from a peer is to use the SleepTime parameter in the Waiting Exchange. However, if the EAP server receives repeated EAP-NOOB connections from a peer which apparently should not connect to this server, the server MAY indicate that the connections are unwanted by sending the error code 2001. After receiving this error message, the peer MAY refrain from reconnecting to the same EAP server and, if possible, both the EAP server and peer SHOULD indicate this error condition to the user. However, in order to avoid persistent denial of service, the peer is not required to stop entirely from reconnecting to the server.

3.6.3. State mismatch

In the states indicated by "-" in Figure 10 in Appendix A, user action is required to reset the association state or to recover it, for example, from backup storage. In those cases, the server sends the error code 2002 to the peer. If possible, both the EAP server and peer SHOULD indicate this error condition to the user.

3.6.4. Negotiation failure

If there is no matching protocol version, the peer sends the error code 3001 to the server. If there is no matching cryptosuite, the peer sends the error code 3002 to the server. If there is no matching OOB direction, the peer sends the error code 3003 to the server.

In practice, there is no way of recovering from these errors without software or hardware changes. If possible, both the EAP server and peer SHOULD indicate these error conditions to the user. In particular, user action is needed for resetting the association from a persistent state to the Unregistered (0) state.

3.6.5. Cryptographic verification failure

If the receiver of the OOB message detects an unrecognized PeerId or incorrect fingerprint (Hoob) in the OOB message, the receiver MUST remain in the Waiting for OOB state (1) as if no OOB message was received. The receiver SHOULD indicate the failure to accept the OOB message to the user.

Note that if the OOB message was delivered from the server to the peer and the peer does not recognize the PeerId, the likely cause is that the user has unintentionally delivered the OOB message to the wrong destination. If possible, the peer SHOULD indicate this to the user; however, the peer device may not have the capability for many different error indications to the user and it MAY use the same indication as in the case of an incorrect fingerprint.

The rationale for the above is that the invalid OOB message could have been presented to the receiver by mistake or intentionally by a malicious party and, thus, it should be ignored in the hope that the honest user will soon deliver a correct OOB message.

If the EAP server or peer detects an incorrect message authentication code (MACs, MACp, MACs2, MACp2), it sends the error code 4001 to the other side. As specified in the beginning of Section 3.6, the failed Completion Exchange will not result in server or peer state changes while error in the Reconnect Exchange will put both sides to the Reconnecting (3) state and thus lead to another reconnect attempt.

The rationale for this is that the invalid cryptographic message may have been spoofed by a malicious party and, thus, it should be ignored. In particular, a spoofed message on the in-band channel should not force the honest user to perform the OOB Step again. In practice, however, the error may be caused by other failures, such as a software bug. For this reason, the EAP server MAY limit the rate of peer connections with SleepTime after the above error. Also, there MUST be a way for the user to reset the EAP server and peer to the Unregistered state (0), so that the OOB Step can be repeated.

3.6.6. Application-specific failure

Applications MAY define new error messages for failures that are specific to the application or to one type of OOB channel. They MAY also use the generic application-specific error code 5001, or the error codes 5002 and 5003, which have been reserved for indicating invalid data in the ServerInfo and PeerInfo fields, respectively. Additionally, anticipating OOB channels that make use of a URL, the error code 5003 has been reserved for indicating invalid server URL.

4. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to the EAP-NOOB protocol, in accordance with [RFC8126].

The EAP Method Type number for EAP-NOOB needs to be assigned.

This memo also requires IANA to create new registries as defined in the following subsections.

4.1. Cryptosuites

Cryptosuites are identified by an integer. Each cryptosuite MUST specify an ECDHE curve for the key exchange, encoding of the ECDHE public key as a JWK object, and a cryptographic hash function for the fingerprint and MAC computation and key derivation. The hash value output by the cryptographic hash function MUST be at least 32 bytes in length. The following suites are defined by EAP-NOOB:

Cryptosuite	Algorithms
1	ECDHE curve Curve25519 [RFC7748], public-key format [RFC7518] Section 6.2.1, hash function SHA-256 [RFC6234]

Table 6: EAP-NOOB cryptosuites

An example of Cryptosuite 1 public-key encoded as a JWK object is given below (line breaks are for readability only).

```
"jwk":{"kty":"EC","crv":"Curve25519","x":"3p7bfXt9wbTTW2HC7OQ1Nz-
DQ8hbeGdNrfx-FG-IK08"}
```

Assignment of new values for new cryptosuites MUST be done through IANA with "Specification Required" and "IESG Approval" as defined in [RFC8126].

4.2. Message Types

EAP-NOOB Request and Response pairs are identified by an integer Message Type. The following Message Types are defined by EAP-NOOB:

Message Type	Used in Exchange	Purpose
1	Initial	Version, cryptosuite and parameter negotiation
2	Initial	Exchange of ECDHE keys and nonces
3	Waiting	Indication to peer that the server has not yet received an OOB message
4	Completion	Authentication and key confirmation with MAC
5	Reconnect	Version, cryptosuite, and parameter negotiation
6	Reconnect	Exchange of ECDHE keys and nonces
7	Reconnect	Authentication and key confirmation with MAC
8	Completion	NoobId discovery
0	Error	Error notification

Table 7: EAP-NOOB

Assignment of new values for new Message Types MUST be done through IANA with "Expert Review" as defined in [RFC8126].

4.3. Error codes

The error codes defined by EAP-NOOB are listed in Table 8.

Error code	Purpose
1001	Invalid NAI
1002	Invalid message structure
1003	Invalid data
1004	Unexpected message type
1005	Unexpected peer identifier
1006	Unrecognized OOB message identifier
1007	Invalid ECDHE key
2001	Unwanted peer
2002	State mismatch, user action required
3001	No mutually supported protocol version
3002	No mutually supported cryptosuite
3003	No mutually supported OOB direction
4001	MAC verification failure
5001	Application-specific error
5002	Invalid server info
5003	Invalid server URL
5004	Invalid peer info
6001-6999	Private and experimental use

Table 8: EAP-NOOB error codes

Assignment of new error codes MUST be done through IANA with "Specification Required" and "IESG Approval" as defined in [RFC8126], with the exception of the range 6001-6999, which is reserved for "Private Use" and "Experimental Use".

4.4. Domain name reservation considerations

"eap-noob.net" should be registered as a special-use domain. The considerations required by [RFC6761] for registering this special use domain name are as follows:

- Users: Non-admin users are not expected to encounter this name or recognize it as special. AAA administrators may need to recognize the name.

- Application Software: Application software is not expected to recognize this domain name as special.

- Name Resolution APIs and Libraries: Name resolution APIs and libraries are not expected to recognize this domain name as special.

- Caching DNS Servers: Caching servers are not expected to recognize this domain name as special.

- Authoritative DNS Servers: Authoritative DNS servers MUST respond to queries for eap-noob.net with NXDOMAIN.

- DNS Server Operators: Except for the authoritative DNS server, there are no special requirements for the operators.

- DNS Registries/Registrars: There are no special requirements for DNS registrars.

5. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

5.1. Implementation with wpa_supplicant and hostapd

- Responsible Organization: Aalto University

- Location: <https://github.com/tuomaura/eap-noob>

- Coverage: This implementation includes all of the features described in the current specification. The implementation supports two dimensional QR codes and NFC as example out-of-band (OOB) channels

- Level of Maturity: Alpha

- Version compatibility: Version 03 of the draft implemented

- Licensing: BSD

- Contact Information: Tuomas Aura, tuomas.aura@aalto.fi

5.2. Protocol modeling

The current EAP-NOOB specification has been modeled with the mCRL2 formal specification language [mcrl2]. The model <https://github.com/tuomaura/eap-noob/tree/master/protocolmodel/mcrl2> was used mainly for simulating the protocol behavior and for verifying basic safety and liveness properties as part of the specification process. For example, the model was used to verify the correctness of the tiebreaking mechanism when two OOB messages are received simultaneously, one in each direction. The mCRL2 model is not intended for verifying security of the authenticated key-exchange; a different modeling method is needed for that. It was, however, used to verify that a man-in-the-middle attacker cannot cause persistent failure by spoofing a finite number of messages.

6. Security considerations

EAP-NOOB is an authentication and key derivation protocol and, thus, security considerations can be found in most sections of this specification. In the following, we explain the protocol design and highlight some other special considerations.

6.1. Authentication principle

EAP-NOOB establishes a shared secret with an authenticated ECDHE key exchange. The mutual authentication in EAP-NOOB is based on two separate features, both conveyed in the OOB message. The first authentication feature is the secret nonce Noob. The peer and server use this secret in the Completion Exchange to mutually authenticate the session key previously created with ECDHE. The message authentication codes computed with the secret nonce Noob are alone sufficient for authenticating the key exchange. The second authentication feature is the integrity-protecting fingerprint Hoob. Its purpose is to prevent impersonation and man-in-the-middle attacks even in situations where the attacker is able to eavesdrop the OOB channel and the nonce Noob is compromised. In some human-assisted OOB channels, such as sound burst or user-transferred URL, it may be easier to detect tampering than eavesdropping of the OOB message, and such applications benefit from the second authentication feature.

The additional security provided by the cryptographic fingerprint Hoob is somewhat intricate to understand. The endpoint that receives the OOB message uses Hoob to verify the integrity of the ECDHE exchange. Thus, the OOB receiver can detect impersonation and man-in-the-middle attacks on the in-band channel. The other endpoint, however, is not equally protected because the OOB message and fingerprint are sent only in one direction. Some protection to the OOB sender is afforded by the fact that the user may notice the

failure of the association at the OOB receiver and therefore reset the OOB sender. Other device-pairing protocols have solved similar situations by requiring the user to confirm to the OOB sender that the association was accepted by the OOB receiver, e.g. by pressing an "accept" button on the sender side. Applications MAY implement EAP-NOOB in this way. Nevertheless, since EAP-NOOB was designed to work with strictly one-directional OOB communication and the fingerprint is only the second authentication feature, the EAP-NOOB specification does not mandate such explicit confirmation to the OOB sender.

To summarize, EAP-NOOB uses the combined protection of the secret nonce Noob and the cryptographic fingerprint Hoob, both conveyed in the OOB message. The secret nonce Noob alone is sufficient for mutual authentication, unless the attacker can eavesdrop it from the OOB channel. Even if an attacker is able to eavesdrop the secret nonce Noob, it nevertheless cannot perform a full man-in-the-middle attack on the in-band channel because the mismatching fingerprint would alert the OOB receiver, which would reject the OOB message. That attacker that eavesdropped the secret nonce can impersonate the OOB receiver to the OOB sender. In this case, the association will appear to be complete only on the OOB sender side, and such situations have to be resolved by the user by resetting the sender to the initial state.

The expected use cases for EAP-NOOB are ones where it replaces a user-entered access credentials in IoT appliances. In wireless network access without EAP, the user-entered credential is often a passphrase that is shared by all the network stations. The advantage of an EAP-based solution, including EAP-NOOB, is that it establishes a different master secret for each peer device, which makes the system more resilient against device compromise than if there were a common master secret. Additionally, it is possible to revoke the security association for an individual device on the server side.

Forward secrecy in EAP-NOOB is optional. The Reconnect Exchange in EAP-NOOB provides forward secrecy only if both the server and peer send their fresh ECDHE keys. This allows both the server and the peer to limit the frequency of the costly computation that is required for forward secrecy. The server MAY adjust the frequency of its attempts at ECDHE rekeying based on what it knows about the peer's computational capabilities.

The users delivering the OOB messages will often authenticate themselves to the EAP server, e.g. by logging into a secure web page. In this case, the server can reliably associate the peer device with the user account. Applications that make use of EAP-NOOB can use this information for configuring the initial owner of the freshly-registered device.

6.2. Identifying correct endpoints

One remaining threat against EAP-NOOB is that the attacker first impersonates the peer on the in-band channel in the Initial Exchange and, during the OOB step, tricks the user to deliver the OOB message to or from the wrong peer device. The server will now be associated with that wrong peer. Similarly, the attacker can try to trick the user to deliver the OOB message to the wrong server, so that the peer device becomes associated with the wrong server. This reliance on user in identifying the correct endpoints is an inherent property of user-assisted out-of-band authentication.

One mechanism that can mitigate user mistakes is certification of peer devices. The certificate can convey to the server authentic identifiers and attributes of the peer device. Compared to a fully certificate-based authentication, however, EAP-NOOB can be used without trusted third parties and does not require the user to know any identifier of the peer device; physical access to the device is sufficient.

The user could accidentally deliver the OOB message to a wrong peer device in addition to the correct one. Such accidents in EAP-NOOB will not enable the wrong device to compute the master key of the correct device or to eavesdrop the communication protected by that key. This is because the wrong device would need to have performed a man-in-the middle attack on the in-band Initial Exchange before the accident occurred. In comparison, simpler solutions where the master key is transferred to the device via the OOB channel are vulnerable to opportunistic attacks if the user mistakenly delivers the master key to more than one device.

6.3. Trusted path issues

Another remaining threat is spoofed user input or output on the peer device. When the user is delivering the OOB message to or from a peer device, a trusted path between the user and the peer device is needed. That is, the user must communicate directly with an authentic operating system and EAP-NOOB implementation in the peer device and not with a spoofed user interface. Otherwise, a Registered device that is under the control of the attacker could emulate the behavior of an Unregistered device. The secure path can be implemented, for example, by indicating the peer device's status (Unregistered or Registered/Reconnecting) when the devices is powered up or by requiring the user to press a reset button to trigger the Initial Exchange.

6.4. Peer identifiers and attributes

The PeerId value in the protocol is a server-allocated identifier for its association with the peer and SHOULD NOT be shown to the user because its value is initially ephemeral. Since the PeerId is allocated by the server and the scope of the identifier is the single server, the so-called identifier squatting attacks, where a malicious peer could reserve another peer's identifier, are not possible in EAP-NOOB. The server SHOULD assign a random or pseudo-random PeerId to each new peer. It SHOULD NOT select the PeerId based on any peer characteristics that it may know, such as the peer's link-layer network address.

User reset or failure in the OOB Step can cause the peer to perform many Initial Exchanges with the server and to allocate many PeerIds and to store the ephemeral protocol state for them. The peer will typically only remember the latest one. EAP-NOOB leaves it to the implementation to decide when to delete these ephemeral associations. There is no security reason to delete them early, and the server does not have any way to verify that the peers are actually the same one. Thus, it is safest to store the ephemeral states for at least one day. If the OOB messages are sent only in the server-to-peer direction, the server SHOULD NOT delete the ephemeral state before all the related Noob values have expired.

After completion of EAP-NOOB, the server may store the PeerInfo data, and the user may use it to identify the peer and its properties, such as the make and model or serial number. A compromised peer could lie in the PeerInfo that it sends to the server. If the server stores any information about the peer, it is important that this information is approved by the user during or after the OOB Step. Without verification by the user or authentication with vendor certificates on the application level, the PeerInfo is not authenticated information and should not be relied on.

One possible use for the PeerInfo field is EAP channel binding ([RFC3748] Section 7.15). That is, the PeerInfo may include data items that bind the EAP-NOOB association and exported keys to properties of the authenticator or the access link, such as the SSID and BSSID of the wireless network (see Appendix C).

6.5. Identity protection

The PeerInfo field contains identifiers and other information about the peer device (see Appendix C), and the peer sends this information in plaintext to the EAP server before the server authentication in EAP-NOOB has been completed. While the information refers to the peer device and not directly to the user, it may be better for user

privacy to avoid sending unnecessary information. In the Reconnect Exchange, the optional PeerInfo SHOULD be omitted unless some critical data has changed.

Peer devices that randomize their layer-2 address to prevent tracking can do this whenever the user resets the EAP-NOOB association. During the lifetime of the association, the PeerId is a unique identifier that can be used to track the peer in the access network. Later versions of this specification may consider updating the PeerId at each Reconnect Exchange. In that case, it is necessary to consider how the authenticator and access-network administrators can recognize and blacklist misbehaving peer devices and how to avoid loss of synchronization between the server and the peer if messages are lost during the identifier update.

6.6. Downgrading threats

The fingerprint Hoob protects all the information exchanged in the Initial Exchange, including the cryptosuite negotiation. The message authentication codes MACs and MACp also protect the same information. The message authentication codes MACs2 and MACp2 protect information exchanged during key renegotiation in the Reconnect Exchange. This prevents downgrading attacks to weaker cryptosuites as long as the possible attacks take more time than the maximum time allowed for the EAP-NOOB completion. This is typically the case for recently discovered cryptanalytic attacks.

As an additional precaution, the EAP server and peer SHOULD check for downgrading attacks in the Reconnect Exchange. As long as the server or peer saves any information about the other endpoint, it MUST also remember the previously negotiated cryptosuite and MUST NOT accept renegotiation of any cryptosuite that is known to be weaker than the previous one, such as a deprecated cryptosuite.

Integrity of the direction negotiation cannot be verified in the same way as the integrity of the cryptosuite negotiation. That is, if the OOB channel used in an application is critically insecure in one direction, a man-in-the-middle attacker could modify the negotiation messages and thereby cause that direction to be used. Applications that support OOB messages in both directions SHOULD therefore ensure that the OOB channel has sufficiently strong security in both directions. While this is a theoretical vulnerability, it could arise in practice if EAP-NOOB is deployed in unexpected applications. However, most devices acting as the peer are likely to support only one direction of exchange, in which case interfering with the direction negotiation can only prevent the completion of the protocol.

The long-term shared key material Kz in the persistent EAP-NOOB association is established with an ECDHE key exchange when the peer and server are first associated. It is a weaker secret than a manually configured random shared key because advances in cryptanalysis against the used ECDHE curve could eventually enable the attacker to recover Kz. EAP-NOOB protects against such attacks by allowing cryptosuite upgrades in the Reconnect Exchange and by updating shared key material Kz whenever the cryptosuite is upgraded. We do not expect the cryptosuite upgrades to be frequent, but if one becomes necessary, the upgrade can be made without manual resetting and reassociation of the peer devices.

6.7. Recovery from loss of last message

EAP methods do not receive notification of an EAP-Success message from the parent EAP state machine [RFC4137] after successful authentication. As stated in [RFC3748], EAP-Success messages are not protected, may be lost, and are not re-transmitted. Therefore, in the Completion Exchange (shown in Figure 5), after the EAP peer sends the last EAP-Response (Type=4) containing the MACp, it exports the keying material such as MSK, EMSK and Session-Id to the parent EAP state machine. A loss of this message in the absence of an attacker would not have any consequences as EAP Retransmission Behavior defined in Section 4.3 of [RFC3748] suggests 3-5 retransmissions.

However, a determined attacker can drop the final EAP-Response message (and all subsequent retransmissions) leaving the peer in registered state (4) while the server remains in state 1 or 2 (depending if the OOB message direction was server-to-peer or peer-to-server).

An attacker can try to use a similar strategy for leaving the peer and the server in discordant states during the Reconnect Exchange. In this scenario, after the EAP peer successfully verifies the server MACs2 received, it sends the last EAP-Response (Type=7) containing the MACp2. The peer would move state 4 and export all the keying material to the parent EAP state machine. However, an attacker can drop the last EAP-Response (and all subsequent retransmissions) leaving the server in state 3.

6.8. Other denial-of-service threats

To be written.

6.9. EAP security claims

EAP security claims are defined in section 7.2.1 of [RFC3748]. The security claims for EAP-NOOB are listed in Table 9.

Security property	EAP-NOOB claim
Authentication mechanism	ECDHE key exchange with out-of-band authentication
Protected cryptosuite negotiation	yes
Mutual authentication	yes
Integrity protection	yes
Replay protection	yes
Key derivation	yes
Key strength	The specified cryptosuites provide key strength of at least 128 bits.
Dictionary attack protection	yes
Fast reconnect	yes
Cryptographic binding	not applicable
Session independence	yes
Fragmentation	no
Channel binding	yes (The ServerInfo and PeerInfo can be used to convey integrity-protected channel properties such as network SSID or peer MAC address.)

Table 9: EAP security claims

7. References

7.1. Normative references

[NIST-DH] Barker, E., Chen, L., Roginsky, A., and M. Smid, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", NIST Special Publication 800-56A Revision 2 , May 2013, <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[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>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, DOI 10.17487/RFC5247, August 2008, <https://www.rfc-editor.org/info/rfc5247>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <https://www.rfc-editor.org/info/rfc6761>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/info/rfc7517>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <https://www.rfc-editor.org/info/rfc7542>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

7.2. Informative references

[BluetoothPairing] Bluetooth, SIG, "Simple pairing whitepaper", Technical report , 2007.

[EUI-48] Institute of Electrical and Electronics Engineers, "802-2014 IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture.", IEEE Standard 802-2014. , June 2014.

[IEEE-802.1X] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Port-Based Network Access Control", IEEE Standard 802.1X-2004. , December 2004.

[mcrl2] Groote, J. and M. Mousavi, "Modeling and analysis of communicating systems", The MIT press , 2014, <https://mitpress.mit.edu/books/modeling-and-analysis-communicating-systems>.

[RFC2904] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., and D. Spence, "AAA Authorization Framework", RFC 2904, DOI 10.17487/RFC2904, August 2000, <https://www.rfc-editor.org/info/rfc2904>.

[RFC4137] Vollbrecht, J., Eronen, P., Petroni, N., and Y. Ohba, "State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator", RFC 4137, DOI 10.17487/RFC4137, August 2005, <https://www.rfc-editor.org/info/rfc4137>.

[RFC4266] Hoffman, P., "The gopher URI Scheme", RFC 4266, DOI 10.17487/RFC4266, November 2005, <https://www.rfc-editor.org/info/rfc4266>.

[RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, DOI 10.17487/RFC5216, March 2008, <https://www.rfc-editor.org/info/rfc5216>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[Sethi14] Sethi, M., Oat, E., Di Francesco, M., and T. Aura, "Secure Bootstrapping of Cloud-Managed Ubiquitous Displays", Proceedings of ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2014), pp. 739-750, Seattle, USA , September 2014, <http://dx.doi.org/10.1145/2632048.2632049>.

Appendix A. Exchanges and events per state

Figure 10 shows how the EAP server chooses the exchange type depending on the server and peer states. In the state combinations marked with hyphen "-", there is no possible exchange and user action is required to make progress. Note that peer state 4 is omitted from the table because the peer never connects to the server when the peer is in that state. The table also indicates the handling of errors in each exchange. A notable detail is that the recipient of error code 1006 moves to state 1.

peer states	exchange chosen by server	next peer and server states
server state: Unregistered (0)		
0..2	Initial Exchange	both 1 (0 on error)
3	-	no change, notify user
server state: Waiting for OOB (1)		
0	Initial Exchange	both 1 (0 on error)
1	Waiting Exchange	both 1 (no change on error)
2	Completion Exchange	both 4 (A)
3	Completion Exchange	both 4
server state: OOB Received (2)		
0	Initial Exchange	both 1 (0 on error)
1	Completion Exchange	both 4 (B)
2	Completion Exchange	both 4 (A)
3	Completion Exchange	both 4
server state: Reconnecting (3) or Registered (4)		
0..2	-	no change, notify user
3	Reconnect Exchange	both 4 (3 on error)

(A) peer to 1 on error 1006, no other changes on error
(B) server to 1 on error 1006, no other changes on error

Figure 10: How server chooses the exchange type

Figure 11 lists the local events that can take place in the server or peer. Both the server and peer output and accept OOB messages in

association state 1, leading the receiver to state 2. Communication errors and timeouts in states 0..2 lead back to state 0, while similar errors in states 3..4 lead to state 3. Application request for rekeying (e.g. to refresh session keys or to upgrade cryptosuite) also takes the association from state 3..4 to state 3. User can always reset the association state to 0. Recovering association data, e.g. from a backup, leads to state 3.

server/ peer state	possible local events on server and peer	next state
1	OOB Output*	1
1	OOB Input*	2 (1 on error)
0..2	Timeout/network failure	0
3..4	Timeout/network failure	3
3..4	Rekeying request	3
0..4	User resets peer state	0
0..4	Association state recovery	3

Figure 11: Local events on server and peer

Appendix B. Application-specific parameters

Table 10 lists OOB channel parameters that need to be specified in each application that makes use of EAP-NOOB. The list is not exhaustive and is included for the convenience of implementors only.

Parameter	Description
OobDirs	Allowed directions of the OOB channel
OobMessageEncoding	How the OOB message data fields are encoded for the OOB channel
SleepTimeDefault	Default minimum time in seconds that the peer should sleep before the next Waiting Exchange
OobRetries	Number of received OOB messages with invalid Hoob after which the receiver moves to Unregistered (0) state
NoobTimeout	How many seconds the sender of the OOB message remembers the sent Noob value. The RECOMMENDED value is 3600 seconds.
ServerInfoMembers	Required members in ServerInfo
PeerInfoMembers	Required members in PeerInfo

Table 10: OOB channel characteristics

Appendix C. ServerInfo and PeerInfo contents

The ServerInfo and PeerInfo fields in the Initial Exchange and Reconnect Exchange enable the server and peer, respectively, send information about themselves to the other endpoint. They contain JSON objects whose structure may be specified separately for each application and each type of OOB channel. ServerInfo and PeerInfo MAY contain auxiliary data needed for the OOB channel messaging and for EAP channel binding. Table 11 lists some suggested data fields for ServerInfo.

Data field	Description
ServerName	String that may be used to aid human identification of the server.
ServerURL	Prefix string when the OOB message is formatted as URL, as suggested in Appendix E.
SSIDList	List of wireless network identifier (SSID) strings used for roaming support, as suggested in Appendix D. The SSIDs are UTF-8 encoded.
Base64SSIDList	List of wireless network identifier (SSID) strings used for roaming support, as suggested in Appendix D. The SSIDs are base64url encoded. Peer SHOULD send at most one of the fields SSIDList and Base64SSIDList in PeerInfo, and the server SHOULD ignore SSIDList if Base64SSIDList is included.

Table 11: Suggested ServerInfo data fields

PeerInfo typically contains auxiliary information for identifying and managing peers on the application level at the server end. Table 12 lists some suggested data fields for PeerInfo.

Data field	Description
PeerName	String that may be used to aid human identification of the peer.
Manufacturer	Manufacturer or brand string.
Model	Manufacturer-specified model string.
SerialNumber	Manufacturer-assigned serial number.
MACAddress	Peer link-layer identifier (EUI-48) in the 12-digit base-16 form [EUI-48]. The string MAY include additional colon ':' or dash '-' characters that MUST be ignored by the server.
SSID	Wireless network SSID for channel binding. The SSID is a UTF-8 string.
Base64SSID	Wireless network SSID for channel binding. The SSID is base64url encoded. Peer SHOULD send at most one of the fields SSID and Base64SSID in PeerInfo, and the server SHOULD ignore SSID if Base64SSID is included.
BSSID	Wireless network BSSID (EUI-48) in the 12-digit base-16 form [EUI-48]. The string MAY include additional colon ':' or dash '-' characters that MUST be ignored by the server.

Table 12: Suggested PeerInfo data fields

Appendix D. EAP-NOOB roaming

AAA architectures [RFC2904] allow for roaming of network-connected appliances that are authenticated over EAP. While the peer is roaming in a visited network, authentication still takes place between the peer and an authentication server at its home network. EAP-NOOB supports such roaming by assigning a Realm to the peer. After the Realm has been assigned, the peer's NAI enables the visited network to route the EAP session to the peer's home AAA server.

A peer device that is new or has gone through a hard reset should be connected first to the home network and establish an EAP-NOOB association with its home AAA server before it is able to roam.

After that, it can perform the Reconnect Exchange from the visited network.

Alternatively, the device may provide some method for the user to configure the Realm of the home network. In that case, the EAP-NOOB association can be created while roaming. The device will use the user-assigned Realm in the Initial Exchange, which enables the EAP messages to be routed correctly to the home AAA server.

While roaming, the device needs to identify the networks where the EAP-NOOB association can be used to gain network access. For 802.11 access networks, the server MAY send a list of SSIDs in the ServerInfo JSON object in a member called SSIDList or Base64SSIDList. This member contains a JSON array where each element is an SSID that is base64url encoded without padding. If present, the peer MAY use this list as a hint to determine the networks where the EAP-NOOB association can be used for access authorization, in addition to the access network where the Initial Exchange took place.

Appendix E. OOB message as URL

While EAP-NOOB does not mandate any particular OOB communication channel, typical OOB channels include graphical displays and emulated NFC tags. In the peer-to-server direction, it may be convenient to encode the OOB message as a URL, which is then encoded as a QR code for displays and printers or as an NDEF record for NFC tags. A user can then simply scan the QR code or NFC tag and open the URL, which causes the OOB message to be delivered to the authentication server. The URL MUST specify the https protocol i.e. secure connection to the server, so that the man-in-the-middle attacker cannot read or modify the OOB message.

The ServerInfo in this case includes a JSON member called ServerUrl of the following format with maximum length of 60 characters:

https://<host>[:<port>]/[<path>]

To this, the peer appends the OOB message fields (PeerId, Noob, Hoob) as a query string. PeerId is provided to the peer by the server and might be a 22-character string. The peer base64url encodes, without padding, the 16-byte values Noob and Hoob into 22-character strings. The query parameters MAY be in any order. The resulting URL is of the following format:

https://<host>[:<port>]/[<path>]?P=<PeerId>&N=<Noob>&H=<Hoob>

The following is an example of a well-formed URL encoding the OOB message (without line breaks):

https://example.com/Noob?P=ZrD7qkczNoHGbGcN2bN0&N=rMinS0-F4EfCU8D9ljx
X_A&H=QvnMp4UGxuQVFaXPW_14UW

Appendix F. Example messages

The message examples in this section are generated with Curve25519 ECDHE test vectors specified in section 6.1 of [RFC7748] (server=Alice, peer=Bob). The direction of the OOB channel negotiated is 1 (peer-to-server). The JSON messages are as follows (line breaks are for readability only).

====== Initial Exchange ======

Identity response:
noob@eap-noob.net

EAP request (type 1):
{"Type":1,"Vers":[1],"PeerId":"07KRU6OgqX0HIeRFldnbSW","Realm":"no
ob.example.com","Cryptosuites":[1],"Dirs":3,"ServerInfo":{"Name":"
Example","Url":"https://noob.example.com/sendOOB"}}

EAP response (type 1):
{"Type":1,"Verp":1,"PeerId":"07KRU6OgqX0HIeRFldnbSW","Cryptosuitep
":1,"Dirp":1,"PeerInfo":{"Make":"Acme","Type":"None","Serial":"DU-
9999","SSID":"Noob1","BSSID":"6c:19:8f:83:c2:80"}}

EAP request (type 2):
{"Type":2,"PeerId":"07KRU6OgqX0HIeRFldnbSW","PKs":{"kty":"EC","crv
":"Curve25519","x":"hSDwCYkwp1R0i33ctD73Wg2_Og0mOBr066SpjqqbTmo"},
"Ns":"PYO7NVd9Af3BxEri1MI6hL8Ck49YxwCjSRPqlC1SPbw","SleepTime":60}

EAP response (type 2):
{"Type":2,"PeerId":"07KRU6OgqX0HIeRFldnbSW","PKp":{"kty":"EC","crv
":"Curve25519","x":"3p7bfXt9wbTTW2HC7OQ1Nz-DQ8hbeGdNrfx-FG-
IK08"},"Np":"HIvB6g0n2btpxEcU7YXnWB-451ED6L6veQQd6ugiPFU"}

====== Waiting Exchange ======

Identity response:
07KRU6OgqX0HIeRFldnbSW+s1@noob.example.com

EAP request (type 3):
{"Type":3,"PeerId":"07KRU6OgqX0HIeRFldnbSW","SleepTime":60}

EAP response (type 3):
{"Type":3,"PeerId":"07KRU6OgqX0HIeRFldnbSW"}

====== OOB Step ======

```
   Identity response:
      P=07KRU6OgqX0HIeRFldnbSW&N=x3JlolaPciK4Wa6XlMJxtQ&H=faqWz68trUrBTK
      AnioZMQA

   ====== Completion Exchange ======

   Identity response:
      07KRU6OgqX0HIeRFldnbSW+s2@noob.example.com

   EAP request (type 8):
      {"Type":8,"PeerId":"07KRU6OgqX0HIeRFldnbSW"}

   EAP response (type 8):
      {"Type":8,"PeerId":"07KRU6OgqX0HIeRFldnbSW","NoobId":"U0OHwYGCS4nE
      kzk2TPIE6g"}

   EAP request (type 4):
      {"Type":4,"PeerId":"07KRU6OgqX0HIeRFldnbSW","NoobId":"U0OHwYGCS4nE
      kzk2TPIE6g","MACs":"Y5NfKQkZTbRW3sEFhWy0Bv0ic2wsMnaA6xGqtUmQqmc"}

   EAP response (type 4):
      {"Type":4,"PeerId":"07KRU6OgqX0HIeRFldnbSW","MACp":"ddY225rN3lYzo7
      qZNPStbVO1HRdNnTx0Rit6_8xEh7A"}

   ====== Reconnect Exchange ======

   Identity response:
      07KRU6OgqX0HIeRFldnbSW+s3@noob.example.com

   EAP request (type 5):
      {"Type":5,"Vers":[1],"PeerId":"07KRU6OgqX0HIeRFldnbSW","Cryptosuit
      es":[1],"Realm":"noob.example.com","ServerInfo":{"Name":"Example",
      "Url":"https://noob.example.com/sendOOB"}}

   EAP response (type 5):
      {"Type":5,"Verp":1,"PeerId":"07KRU6OgqX0HIeRFldnbSW","Cryptosuitep
      ":1,"PeerInfo":{"Make":"Acme","Type":"None","Serial":"DU-
      9999","SSID":"Noob1","BSSID":"6c:19:8f:83:c2:80"}}

   EAP request (type 6):
      {"Type":6,"PeerId":"07KRU6OgqX0HIeRFldnbSW","PKs2":{"kty":"EC","cr
      v":"Curve25519","x":"hSDwCYkwp1R0i33ctD73Wg2_Og0mOBr066SpjqqbTmo"}
      ,"Ns2":"RDLahHBlIgnmL_F_xcynrHurLPkCsrp3G3B_S82WUF4"}

   EAP response (type 6):
      {"Type":6,"PeerId":"07KRU6OgqX0HIeRFldnbSW","PKp2":{"kty":"EC","cr
      v":"Curve25519","x":"3p7bfXt9wbTTW2HC7OQ1Nz-DQ8hbeGdNrfx-FG-
      IK08"},"Np2":"jN0_V4P0JoTqwI9VHHQKd9ozUh7tQdc9ABd-j6oTy_4"}
```

EAP request (type 7):
{"Type":7,"PeerId":"07KRU6OgqX0HIeRFldnbSW","MACs2":"_pXDF4-7uBKXKqVKKB6U-GP9EDnGCNOMdkyfEQp_iwA"}

EAP response (type 7):
{"Type":7,"PeerId":"07KRU6OgqX0HIeRFldnbSW","MACp2":"qSUH4zA0VzMqU2O1U-JJTqwGRXGB8i3bggasYL6o1uU"}

Appendix G. TODO list

- Write security considerations on persistent denial-of-service conditions and avoiding them. Explain how the Completion Exchange is protected against DoS by dropped messages.

Appendix H. Version history

- Version 01:
  - Fixed Reconnection Exchange.
  - URL examples.
  - Message examples.
  - Improved state transition (event) tables.
- Version 02:
  - Reworked the rekeying and key derivation.
  - Increased internal key lengths and in-band nonce and MAC lengths to 32 bytes.
  - Less data in the persistent EAP-NOOB association.
  - Updated reference [NIST-DH] to Revision 2 (2013).
  - Shorter suggested PeerId format.
  - Optimized the example of encoding OOB message as URL.
  - NoobId in Completion Exchange to differentiate between multiple valid Noob values.
  - List of application-specific parameters in appendix.
  - Clarified the equivalence of Unregistered state and no state.

  * Peer SHOULD probe the server regardless of the OOB channel direction.
  * Added new error messages.
  * Realm is part of the persistent association and can be updated.
  * Clarified error handling.
  * Updated message examples.
  * Explained roaming in appendix.
  * More accurate definition of timeout for the Noob nonce.
  * Additions to security considerations.

- Version 03:
  * Clarified reasons for going to Reconnecting state.
  * Included Verp in persistent state.
  * Added appendix on suggested ServerInfo and PeerInfo fields.
  * Exporting PeerId and SessionId.
  * Explicitly specified next state after OOB Step.
  * Clarified the processing of an expired OOB message and unrecognized NoobId.
  * Enabled protocol version upgrade in Reconnect Exchange.
  * Explained handling of redundant received OOB messages.
  * Clarified where raw and base64url encoded values are used.
  * Cryptosuite must specify the detailed format of the JWK object.
  * Base64url encoding in JSON strings is done without padding.
  * Simplified explanation of PeerId, Realm and NAI.
  * Added error codes for private and experimental use.
  * Updated the security considerations.

- Version 04:
  - Recovery from synchronization failure due to lost last response.

Appendix I. Acknowledgments

Aleksi Peltonen modeled the protocol specification with the mCRL2 formal specification language. Shiva Prasad TP and Raghavendra MS implemented parts of the protocol with wpa_supplicant and hostapd. Their inputs helped us in improving the specification.

The authors would also like to thank Rhys Smith and Josh Howlett for providing valuable feedback as well as new use cases and requirements for the protocol. Thanks to Darshak Thakore, Stefan Winter and Hannes Tschofenig for interesting discussions and arguments in this problem space.

Authors' Addresses

Tuomas Aura
Aalto University
Aalto 00076
Finland

EMail: tuomas.aura@aalto.fi

Mohit Sethi
Ericsson
Jorvas 02420
Finland

EMail: mohit@piuha.net

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: November 28, 2018

TCI

May 27, 2018

Certificate Limitation Policy
draft-belyavskiy-certificate-limitation-policy-06

Abstract

The document provides a specification of the application-level trust model. Being provided at the application level, the limitations of trust can be distributed separately using cryptographically protected format instead of hardcoding the checks into the application itself.

1. Introduction

Binary trust model standardized as a set of trusted anchors and CRLs/ OCSP services does not cover all corner cases in the modern crypto world. There is a need in more differentiated limitations. Some of them are suggested [1] by Google when it limits the usage of Symantec's certificates. The CRL profile does not fit the purpose of such limitations. The CRLs are issued by the same CAs that are subject to be limited.

Currently the set of CAs trusted by OS or browsers can be used for the validation purposes. In case when a large enough CA becomes untrusted, it cannot be deleted from the storage of trusted CAs because it may cause error of validation of many certificates. The measures usually taken in such cases usually include application-level limitation of certificates lifetimes, refusing to accept EV-certificates in other way than DV, requirements to use Certificate Transparency, etc.

This document suggests a cryptographically signed format dubbed Certificate Limitation Profile (CLP) designed for description of such limitations. This format can be used by applications that use system-wide set of trust anchors for validating purposes or by applications with own wide enough set of trusted anchors in case when the trust anchor for the entity found misbehaving cannot be revoked.

Currently the only way to provide such limitations is hard coding them in application itself. Using of CLPs does not allow to completely avoid hard coding but allows to hard code only the minimal set of rarely changing data:

the fact that application uses CLP

the certificate to verify the signature under the CLP file

minimal date of the CLP to be used for the current version of application.

It will be possible to move the checks for the limitations to the external cryptographical libraries, such as OpenSSL, instead of checking them at the application level.

2. Certificate Limitations Profile

A proposed syntax and overall structure of CLP is very similar to the one defined for CRLs [2].

```
CertificateList  ::=  SEQUENCE  {
     tbsCertList          TBSCertList,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }

TBSCertList  ::=  SEQUENCE  {
     version                 Version,
     signature               AlgorithmIdentifier,
     issuer                  Name,
     thisUpdate              Time,
     nextUpdate              Time,
     limitedCertificates     SEQUENCE OF SEQUENCE  {
          userCertificate         CertificateSerialNumber,
          certificateIssuer       Name,
          limitationDate          Time,
          limitationPropagation   Enum,
          fingerprint SEQUENCE {
              fingerprintAlgorithm AlgorithmIdentifier,
              fingerprintValue     OCTET STRING
                                 } OPTIONAL,
          limitations            Limitations,
                                 } OPTIONAL,
                               };

Limitations  ::=  SEQUENCE SIZE (1..MAX) OF Limitation

Limitation  ::=  SEQUENCE  {
     limitID      OBJECT IDENTIFIER,
     LimitationValue   OCTET STRING
                 -- contains the DER encoding of an ASN.1 value
                 -- corresponding to the limitation type
                 -- identified by limitID
     }
```

The ASN.1 format of particular limitations is provided in the corresponding sections. Anywhere below the Time in ASN.1 notation is treated according to RFC 5280:

```
Time ::= CHOICE {
     utcTime        UTCTime,
     generalTime    GeneralizedTime }
```

2.1. CLP fields

CLP has general structure similar to CRLs. The upper-level fields are:

TBSCertList - the sequence of individual certificates to be limited;

signatureAlgorithm - the OID of the signature algorithm used for signature;

signatureValue - the bit string representing signature of the TBSCertList.

2.2. CLP signature

The key used for signing the CLP files should have a special Key Usage value and/or an Extended Key Usage value.

2.3. CLP entry fields

Each entry in list contains the following fields:

The issuer of the certificate with limited trust.

The serial of the certificate with limited trust.

The fingerprint of the certificate with limited trust (optional).

limitationPropagation. This field indicates whether limitations are applied to the certificate itself, to all of its descendants in the chain of trust, or both.

and a subset of the following limitations:

issuedNotAfter - do not trust the certs issued after the specified date

trustNotAfter - do not trust the certs after the specified date

validityPeriod, days - take minimal value from "native" validity period and specified in the limitation file

ignoredX509Extensions - list of X.509 extensions of limited certificate that MUST be ignored for the specified certificate (e.g. EV-indicating extensions)

requiredX509extensions - list of X.509 extensions that MUST be present in the certificate to be trusted.

requiredNativeChecking - list of the CA-provided checks that MUST be applied

applicationNameConstraints - list of domains allowed to be issued by this certificate

excludedIssueIntermediatory - disallow issuing of the Intermediatory certificates

The limitations are identified by OIDs

2.3.1. Limitations

2.3.1.1. issuedNotAfter

When this limitation is present, any certificate matching the entry and issued after the specified date MUST NOT be trusted

The issuedNotAfter limitation is identified by OID TBA.

```
issuedNotAfter ::= SEQUENCE {
    IssuedNotAfter    Time
}
```

2.3.1.2. trustNotAfter

When this limitation is present, any certificate matching the entry MUST NOT be trusted after the specified date.

The trustNotAfter limitation is identified by OID TBA.

```
trustNotAfter ::= SEQUENCE {
    TrustNotAfter    Time
}
```

2.3.1.3. validityPeriod

When this limitation is present, no certificate matching the entry should be treated as valid after specified period from its validFrom.

The validityPeriod is measured in days.

The validityPeriod limitation is identified by OID TBA.

```
validityPeriod ::= SEQUENCE {
    Days INTEGER
}
```

2.3.1.4. ignoredX509Extensions

When this limitation is present, the extensions listed in this element should be ignored for the matching certificate.

The ignoredX509Extensions limitation is identified by OID TBA.

```
ignoredX509Extensions :: = SEQUENCE SIZE (1..MAX) OF ExtenID
ExtenID :: = OBJECT IDENTIFIER
```

2.3.1.5. requiredX509extensions

When this limitation is present, the extensions listed in this element should be present for the matching certificate.

The requiredX509extensions limitation is identified by OID TBA.

```
requiredX509extensions :: = SEQUENCE SIZE (1..MAX) OF ExtenID
ExtenID :: = OBJECT IDENTIFIER
```

2.3.1.6. requiredNativeChecking

When this limitation is present, it specifies that the certificates issued by this CA SHOULD be checked against CRL and/or OCSP, depending on contents of the extension.

The requiredNativeChecking limitation is identified by OID TBA.

```
requiredNativeChecking ::= SEQUENCE {
    RequiredCRLChecking BOOLEAN,
    RequiredOCSPChecking BOOLEAN
}
```

2.3.1.7. applicationNameConstraints

This limitation are applied like Name Constraints [3] limitation specified in RFC 5280.

This section implies 2 variants of checks:

- The list of names that are allowed for the CA to issue certificates for
- The list of names that are forbidden for the CA to issue certificates for

The applicationNameConstraints limitation is specified according to RFC 5280, 4.2.1.10 and reuses OID specified in RFC 5280.

```
id-ce-nameConstraints OBJECT IDENTIFIER ::=  { id-ce 30 }

NameConstraints ::= SEQUENCE {
     permittedSubtrees       [0]     GeneralSubtrees OPTIONAL,
     excludedSubtrees        [1]     GeneralSubtrees OPTIONAL }

GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree

GeneralSubtree ::= SEQUENCE {
     base                    GeneralName,
     minimum         [0]     BaseDistance DEFAULT 0,
     maximum         [1]     BaseDistance OPTIONAL }

BaseDistance ::= INTEGER (0..MAX)
```

2.3.1.7.1. excludedIssueIntermediatory

When this limitation is present, the intermediate certificates issued by this CA MUST NOT be trusted.

The excludedIssueIntermediatory limitation is identified by OID TBA.

3. Verification of CLP

The verification of CLP SHOULD be performed by the application. The application should check whether the provided CLP matches the internal requirements and is correclty signed by the specified key.

4. Verification with CLP

In case of using CLP the checks enforced by CLP should be applied after the other checks.

The limitation provided by CLP MUST NOT extend the trustworthy of the checked certificate.

The limitations are applied after cryptographic validation of the certificate and during building its chain of trust. If the certificate or any of its ascendants in the chain of trust matches any record in the CLP, the limitations are applied from the ascendant to descendants. The issuedNotAfter and trustNotAfter limitations are applied to find out the actual validity periods for the any certificate in the chain of trust. If the CLP prescribes to have a particular extension(s) and the certificate does not have it, the certificate MUST NOT be trusted.

Application MAY use more than one CLPs (e.g. app-wide, set of system-wide, user-defined). When multiple CLPs are in use, the limitations are applied simultaneously.

In case when more than one chain of trust are valid for a certificate, if any of this chains is valid after applying the limitations, the certificate MUST be treated as valid.

5. ASN.1 notation

TBD

6. Security considerations

In case when an application uses CLP, it is recommended to specify the minimal date of issuing of the CLP document somewhere in code. It allows to avoid an attack of CLP rollback when the stale version of CLP is used.

It is recommended to distribute CLPs using the channels that are used for distribution of the applications themselves to avoid possible DoS consequences.

If application checks for fresh CLPs, it SHOULD check that nextUpdate field in a fresh one is newer than in the current one. The application MAY accept a CLP with nextUpdate in past. If an application is failing to get updates, then it can continue to run with what it has.

6.1. Unsigned CLP

In case of trusted environment signing CLP can be reluctant. If CLP is delivered via application bundle, it can be verified together with other application data. But it makes sense to separate trust to the source of the content from trust to the content itself. On the other hand it is not a problem to create a local CLP signed by a locally created key.

7. IANA considerations

TBD

8. Acknoledgements

Special thanks to Rich Salz, Igor Ustinov, Vasily Dolmatov, Stanislav Smyishlyaev, Patrik Faeltstroem, Alexander Venedioukhin, Artem Chuprina, Viktor Dukhovni.

9. References

The current version of the document is available on GitHub https://github.com/beldmit/clp

10. References

10.1. URIs

[1] https://groups.google.com/a/chromium.org/forum/#!msg/blink-dev/eUAKwjihhBs/rpxMXjZHCQAJ

[2] https://tools.ietf.org/html/rfc5280#section-5

[3] https://tools.ietf.org/html/rfc5280#section-4.2.1.10

Author’s Address

Dmitry Belyavskiy
Technical Centre of Internet
8 Marta str., 1 bld. 12
Moscow 127083
RU

Email: beldmit@gmail.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: May 29, 2019

TCI

November 25, 2018

Certificate Limitation Policy
draft-belyavskiy-certificate-limitation-policy-07

Abstract

The document provides a specification of the application-level trust model. Being provided at the application level, the limitations of trust can be distributed separately using cryptographically protected format instead of hardcoding the checks into the application itself.

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 May 29, 2019.

1. Introduction

Binary trust model standardized as a set of trusted anchors and CRLs/ OCSP services does not cover all corner cases in the modern crypto world. There is a need in more differentiated limitations. Some of them are suggested [1] by Google when it limits the usage of Symantec's certificates. The CRL profile does not fit the purpose of such limitations. The CRLs are issued by the same CAs that are subject to be limited.

Currently the set of CAs trusted by OS or browsers can be used for the validation purposes. In case when a large enough CA becomes untrusted, it cannot be deleted from the storage of trusted CAs because it may cause error of validation of many certificates. The measures usually taken in such cases usually include application-level limitation of certificates lifetimes, refusing to accept EV-certificates in other way than DV, requirements to use Certificate Transparency, etc.

This document suggests a cryptographically signed format dubbed Certificate Limitation Profile (CLP) designed for description of such limitations. This format can be used by applications that use system-wide set of trust anchors for validating purposes or by applications with own wide enough set of trusted anchors in case when the trust anchor for the entity found misbehaving cannot be revoked.

Currently the only way to provide such limitations is hard coding them in application itself. Using of CLPs does not allow to completely avoid hard coding but allows to hard code only the minimal set of rarely changing data:

the fact that application uses CLP

the certificate to verify the signature under the CLP file

minimal date of the CLP to be used for the current version of application.

It will be possible to move the checks for the limitations to the external cryptographical libraries, such as OpenSSL, instead of checking them at the application level.

2. Certificate Limitations Profile

A proposed syntax and overall structure of CLP is very similar to the one defined for CRLs [2].

```
CertificateList  ::=  SEQUENCE  {
     tbsCertList          TBSCertList,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING  }

TBSCertList  ::=  SEQUENCE  {
     version                 Version,
     signature               AlgorithmIdentifier,
     issuer                  Name,
     thisUpdate              Time,
     nextUpdate              Time,
     limitedCertificates     SEQUENCE OF SEQUENCE  {
          userCertificate         CertificateSerialNumber,
          certificateIssuer       Name,
          limitationDate          Time,
          limitationPropagation   Enum,
          fingerprint SEQUENCE {
              fingerprintAlgorithm AlgorithmIdentifier,
              fingerprintValue     OCTET STRING
                                 } OPTIONAL,
          limitations          Limitations,
                             } OPTIONAL,
                           };

Limitations  ::=  SEQUENCE SIZE (1..MAX) OF Limitation

Limitation  ::=  SEQUENCE  {
     limitID      OBJECT IDENTIFIER,
     LimitationValue   OCTET STRING
                 -- contains the DER encoding of an ASN.1 value
                 -- corresponding to the limitation type
                 -- identified by limitID
     }
```

The ASN.1 format of particular limitations is provided in the corresponding sections. Anywhere below the Time in ASN.1 notation is treated according to RFC 5280:

```
Time ::= CHOICE {
     utcTime        UTCTime,
     generalTime    GeneralizedTime }
```

2.1. CLP fields

CLP has general structure similar to CRLs. The upper-level fields are:

TBSCertList – the sequence of individual certificates to be limited;

signatureAlgorithm – the OID of the signature algorithm used for signature;

signatureValue – the bit string representing signature of the TBSCertList.

2.2. CLP signature

The key used for signing the CLP files should have a special Key Usage value and/or an Extended Key Usage value.

2.3. CLP entry fields

Each entry in list contains the following fields:

The issuer of the certificate with limited trust.

The serial of the certificate with limited trust.

The fingerprint of the certificate with limited trust (optional).

and a subset of the following limitations:

issuedNotAfter – do not trust the certs issued after the specified date

trustNotAfter – do not trust the certs after the specified date

validityPeriod, days – take minimal value from "native" validity period and specified in the limitation file

ignoredX509Extensions – list of X.509 extensions of limited certificate that MUST be ignored for the specified certificate (e.g. EV-indicating extensions)

requiredX509extensions – list of X.509 extensions that MUST be present in the certificate to be trusted.

requiredNativeChecking – list of the CA-provided checks that MUST be applied

applicationNameConstraints – list of domains allowed to be issued by this certificate

excludedIssueIntermediatory - disallow issuing of the Intermediatory certificates

The limitations are identified by OIDs

2.3.1. Limitations

2.3.1.1. issuedNotAfter

When this limitation is present, any certificate matching the entry and issued after the specified date MUST NOT be trusted

The issuedNotAfter limitation is identified by OID TBA.

```
issuedNotAfter ::= SEQUENCE {
    IssuedNotAfter    Time
}
```

2.3.1.2. trustNotAfter

When this limitation is present, any certificate matching the entry MUST NOT be trusted after the specified date.

The trustNotAfter limitation is identified by OID TBA.

```
trustNotAfter ::= SEQUENCE {
    TrustNotAfter    Time
}
```

2.3.1.3. validityPeriod

When this limitation is present, no certificate matching the entry should be treated as valid after specified period from its validFrom.

The validityPeriod is measured in days.

The validityPeriod limitation is identified by OID TBA.

```
validityPeriod ::= SEQUENCE {
    Days INTEGER
}
```

2.3.1.4. ignoredX509Extensions

When this limitation is present, the extensions listed in this element should be ignored for the matching certificate.

The ignoredX509Extensions limitation is identified by OID TBA.

```
ignoredX509Extensions :: = SEQUENCE SIZE (1..MAX) OF ExtenID
ExtenID :: = OBJECT IDENTIFIER
```

2.3.1.5. requiredX509extensions

When this limitation is present, the extensions listed in this element should be present for the matching certificate.

The requiredX509extensions limitation is identified by OID TBA.

```
requiredX509extensions :: = SEQUENCE SIZE (1..MAX) OF ExtenID
ExtenID :: = OBJECT IDENTIFIER
```

2.3.1.6. requiredNativeChecking

When this limitation is present, it specifies that the certificates issued by this CA SHOULD be checked against CRL and/or OCSP, depending on contents of the extension.

The requiredNativeChecking limitation is identified by OID TBA.

```
requiredNativeChecking ::= SEQUENCE {
    RequiredCRLChecking BOOLEAN,
    RequiredOCSPChecking BOOLEAN
}
```

2.3.1.7. applicationNameConstraints

This limitation are applied like Name Constraints [3] limitation specified in RFC 5280.

This section implies 2 variants of checks:

- The list of names that are allowed for the CA to issue certificates for

- The list of names that are forbidden for the CA to issue certificates for

The applicationNameConstraints limitation is specified according to RFC 5280, 4.2.1.10 and reuses OID specified in RFC 5280.

```
id-ce-nameConstraints OBJECT IDENTIFIER ::=  { id-ce 30 }

NameConstraints ::= SEQUENCE {
     permittedSubtrees       [0]     GeneralSubtrees OPTIONAL,
     excludedSubtrees        [1]     GeneralSubtrees OPTIONAL }

GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree

GeneralSubtree ::= SEQUENCE {
     base                    GeneralName,
     minimum         [0]     BaseDistance DEFAULT 0,
     maximum         [1]     BaseDistance OPTIONAL }

BaseDistance ::= INTEGER (0..MAX)
```

2.3.1.7.1. excludedIssueIntermediatory

When this limitation is present, the intermediate certificates issued by this CA MUST NOT be trusted.

The excludedIssueIntermediatory limitation is identified by OID TBA.

3. Verification of CLP

The verification of CLP SHOULD be performed by the application. The application should check whether the provided CLP matches the internal requirements and is correclty signed by the specified key.

4. Verification with CLP

In case of using CLP the checks enforced by CLP should be applied after the other checks.

The limitation provided by CLP MUST NOT extend the trustworthy of the checked certificate.

The limitations are applied after cryptographic validation of the certificate and during building its chain of trust. If the certificate or any of its ascendants in the chain of trust matches any record in the CLP, the limitations are applied from the ascendant to descendants. The issuedNotAfter and trustNotAfter limitations are applied to find out the actual validity periods for the any certificate in the chain of trust. If the CLP prescribes to have a particular extension(s) and the certificate does not have it, the certificate MUST NOT be trusted.

Application MAY use more than one CLPs (e.g. app-wide, set of system-wide, user-defined). When multiple CLPs are in use, the limitations are applied simultaneously.

In case when more than one chain of trust are valid for a certificate, if any of this chains is valid after applying the limitations, the certificate MUST be treated as valid.

5. ASN.1 notation

TBD

6. Real-world considerations

6.1. Expected sources and consumers of CLPs

Public CLPs can be created and distributed by such parties as OS vendors, browser vendors and other parties treated as worth trusting.

Usage of CLPs is reasonable for applications establishing TLS connections with unpredictable sets of peers. The main examples of such applications are web-browsers and MTAs.

6.2. Size limitation

To avoid uncontrolled growth of CLPs, the limitations are applied to root and intermediate CA certificates.

7. Security considerations

In case when an application uses CLP, it is recommended to specify the minimal date of issuing of the CLP document somewhere in code. It allows to avoid an attack of CLP rollback when the stale version of CLP is used.

It is recommended to distribute CLPs using the channels that are used for distribution of the applications themselves to avoid possible DoS consequences.

If application checks for fresh CLPs, it SHOULD check that nextUpdate field in a fresh one is newer than in the current one. The application MAY accept a CLP with nextUpdate in past. If an application is failing to get updates, then it can continue to run with what it has.

7.1. Unsigned CLP

In case of trusted environment signing CLP can be reluctant. If CLP is delivered via application bundle, it can be verified together with other application data. But it makes sense to separate trust to the source of the content from trust to the content itself. On the other hand it is not a problem to create a local CLP signed by a locally created key.

8. IANA considerations

TBD

9. Acknoledgements

Special thanks to Rich Salz, Igor Ustinov, Vasily Dolmatov, Stanislav Smyishlyaev, Patrik Faeltstroem, Alexander Venedioukhin, Artem Chuprina, Viktor Dukhovni.

10. References

The current version of the document is available on GitHub https://github.com/beldmit/clp

11. References

11.1. URIs

[1] https://groups.google.com/a/chromium.org/forum/#!msg/blink-dev/eUAKwjihhBs/rpxMXjZHCQAJ

[2] https://tools.ietf.org/html/rfc5280#section-5

[3] https://tools.ietf.org/html/rfc5280#section-4.2.1.10

Author's Address

Dmitry Belyavskiy
Technical Centre of Internet
8 Marta str., 1 bld. 12
Moscow 127083
RU

Email: beldmit@gmail.com

ACME Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2019

T. Fiebig
TU Delft
K. Borgolte
Princeton University
October 21, 2018

Extended Security Considerations for the Automatic Certificate Management Environment (ESecACME)
draft-fiebig-acme-esecacme-00

Abstract

By now, most Public Key Infrastructure X.509 (PKIX) certificates are issued via the ACME protocol. Recently, several attacks against domain validation (DV) have been published, including IP-use-after-free, (forced) on-path attacks, and attacks on protocols used for validation. In general, these attacks can be mitigated by (selectively) requirering additional challenges, e.g., DNS validation, proof of prior-key-ownership, or in severe cases even extended validation (EV) instead of DV. This document provides a list of critical cases and describes which mitigations can be used to reduce the threat of issuing a certificate to an unauthorized party.

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 April 24, 2019.

Copyright Notice

Table of Contents

1. Introduction

By now, most Public Key Infrastructure X.509 (PKIX) certificates are issued via the ACME protocol. The automated nature of ACME and its heavy use of Domain Validation (DV) make it susceptible to a variety of attacks. These include IP-use-after-free [CSTRIFE], (forced) on-path attacks [BAMBOO], and attacks on protocols used for validation [DVP], e.g., DNS. In general, these attacks can be mitigated by (selectively) requirering additional challenges, e.g., DNS validation, proof of prior-key-ownership, or in severe cases even extended validation (EV) instead of DV.

This document provides a list of potential attacks and how they can be detected. In addition, it describes which mitigations can be used to reduce the threat of issuing a certificate to an unauthorized party in case a potential attack has been detected. This section also holds information on how these mitigations may impact the usability of CAs using ACME to issue certificates.

2. Attacks

In this section we describe common attacks against DV, how they can be detected, and which additional verification methods should be used in case they are detected.

2.1. IP/Resource-use-after-free

IP- and Resource-use-after-free attacks occur if a domain owner points a DNS record to a resource, which they later vacate without deleting the DNS record. The resource, usually in cloud scenarios, can then be allocated by another party.

For example, one might run a service for www.example.com on a virtual machine hosted with a cloud provider. One then points the AAAA record of www.example.com to the IPv6 address of that virtual machine, 2001:DB8:1234:1234::1. However, when the operator discontinues the service, they do not delete this DNS record, leading to a stale record. If another client of the cloud provider now allocates a virtual machine, and receives the same IPv6 address, they can proof ownership of www.example.com to an ACME compliant CA. These observations similarly hold for DNS records pointing to legacy IPv4 resources (A records), mail servers in case of email verification using the ACME protocol (MX records), http and https delegations (SRV records), and DNS delegations if DNSSEC is not being used (NS records).

2.1.1. Detection

This attack type is difficult to detect from the CAs site, without operators taking precautions themselves, which we describe in the following section. Heuristics CAs could use depend on the availability of cooperation from operators, or require proof of prior key ownership.

Ideally, operators will use TLSA records to pin the TLS public key for a name, allowing a CA to match the TLSA record to the key for which a certificate is requested.

If a DNS challenge is used, failed DNSSEC validation may point at a resource-use-after-free attack.

A heuristic which does not require prior cooperation by operators is using Certificate Transparency (CT) logs to identify prior certificate issuances. Furthermore, CAA records could be used to limit the number of CT logs which have to be searched by the ACME compliant CA. Furthermore, if the CA with which a certificate has been requested is also the only CA allowed in the CAA, it could check the ACME account ID of prior requests vs. the one used in the current request.

2.1.2. Defense

On a mismatch between the TLSA public key and the public key used in the request, the CA must deny the requested certificate. In case of pre-existing certificates, or a mismatch in the ACME account ID, the operator should use an additional validation technique. If DNSSEC is being used, the DNS challenge is an option. Given that NS and MX records may also suffer from resource-use-after-free attacks, unauthenticated DNS and email challenges are not an option.

Due to the usability implications of the available defense options a CA may opt to only perform mitigation on high-risk resources, e.g., known cloud operators and operators with a high customer churn.

2.2. (Forced)-on-path Attacks

If an attacker can perform a Monkey-in-the-Middle (MitM) attack by controlling a part of the network path between the CA and the resource used for validation, they can also impersonate an operator and illegitimately obtain a certificate for a domain. Attackers may force this on-path situation, e.g., using BGP shorter-prefix attacks [BAMBOO].

2.2.1. Detection

To detect on-path attacks, CAs should validate challenges from multiple vantage points. For this purpose, the CA should operate a geographically and topologically distributed system for verification. This system should contain at least one validator per IP region (AfriNIC, APNIC, ARIN, LACNIC, RIPE). Similarly, a CA may monitor the BGP prefix from which it received a request with a service similar to https://bgpmon.net [1]. Note that, depending how close the attacker is to the victim, no path without malicious activity may remain, generalizing the detection issue to that outlined for resource-use-after-free attacks.

2.2.2. Defense

The same defense options as for resource-use-after-free attacks apply.

2.3. DNS Cache Poisoning Attacks

Paper just appeared; will be included in the next version of this draft.

2.3.1. Detection

2.3.2. Defense

3. Summary Indicators for Additional Validation

In this section, we summarize indicators for using an extended validation machanism.

3.1. High-Resource-Reuse Source / Cloud Provider

If the validation target for a challenge (A/AAAA/NS/MX) is located in a network with a high resources churn, e.g., a cloud or hosting provider, or a residential ISP, extended validation requirements should be considered.

3.2. Multi-Vantagepoint Validation Mismatch

If at least one of an CAs validation notes does not match the results of the other nodes, the CA must consider the requested domain to be under attack, necessitating either DNSSEC signed DNS validation, proof of prior-key-ownership or EV.

3.3. BGP monitoring

If any prefix for either the A, AAAA, MX, or NS records (or intermediate names and CNAMEs) is considered to be under a BGP MitM attack by a service similar to https://bgpmon.net [2], the CA must consider the requested domain to be under attack, necessitating either DNSSEC signed DNS validation, proof of prior-key-ownership or EV.

3.4. DNS Fragmentation

Paper just appeared; will be included in the next version of this draft.

3.5. Failed DNSSEC Validation

If DNSSEC validation for a domain for which a certificate is requested fails, the CA must consider the domain to be under attack, necessitating either proof of prior-key-ownership or EV.

3.6. Recent Domain Transfer

If a domain has been transfered during the last 72 hours, the CA should consider the domains ownership-state as insufficiently defined, and reuire either proof of prior-key-ownership or EV.

3.7. High-Risk Domains

If a domain is a high-risk domain, CAs should only offer DNSSEC signed DNS validation, proof of prior-key-ownership DV, or EV. Domains are high risk domains if they are part of the Alexa top 10,000, belong to a CA, a software or hardware vendor, or a payment provider.

4. Additional Validation Options

If one or multiple of the indicators above are detected by a CA, it can employ one of the following additional validation options.

4.1. Proof of Prior Key Ownership

If a CA detects an attack, it can require the requesting party to proof that it has access to the private key for a previously issued certificate. This can be done implicitly, by requirering DV over HTTPS, using a validating certificate, or, explicitly, by using a dedicated ACME-challenge.

4.1.1. Limitations

This option has several operational challenges. An operator's infrastructure may not be design in a way that preserves prior private keys, for example in large container setups. Similarly, the prior key might have been lost due to data-loss, or because the systems holding it have been discontinued. Similarly, prior certificates may have expired.

Furthermore, an attacker may have obtained a prior private key by compromising a system, or by having had legitimate authority over the domain before.

4.2. Additional Use of a DNS Challenge

If the CA detects an attack on one validation, e.g., web based DV, it may use ACME-DNS instead.

4.2.1. Limitations

This challenge does not provide full resillience against all attacks. It however increases the effort an adversary has to put into an attack significantly.

4.3. Additional Use of an Email Challenge

If the CA detects an attack on one validation, e.g., web based DV, it may use ACME-EMAIL instead.

4.3.1. Limitations

This challenge does not provide full resillience against all attacks. It however increases the effort an adversary has to put into an attack significantly.

4.3.2. Limitations

4.4. Out-of-Band and offline validation

If a party is unable to proof prior-key-ownership, and any of the attack indicators outlined before is detected by the CA, the CA should perform a traditional extended validation, requesting appropriate documentation from the requesting party.

4.4.1. Limitations

EV is a manual process which prevents ACME from being used. It is significantly more costly and smaller CAs may be unable to provide the necessary infrastructure to support EV.

5. IANA Considerations

There are no IANA considerations.

6. Security Considerations

This document itself serves as a summary of additional security considerations. Operators of CAs should carefully follow the recommendations made in this document to prevent issuing certificates to unauthorized parties.

7. Acknowledgements

8. References

8.1. Normative References

[BAMBOO] Mittal, P., "Bamboozling Certificate Authorities with BGP", August 2018, <https://www.usenix.org/conference/usenixsecurity18/presentation/birge-lee>.

[CSTRIFE] Vigna, G., "Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates", February 2018, <http://dx.doi.org/10.14722/ndss.2018.23327>.

[DVP] Waidner, M., "https://www.usenix.org/conference/usenixsecurity18/presentation/birge-lee", n.d..

8.2. URIs

[1] https://bgpmon.net

[2] https://bgpmon.net

Authors' Addresses

Tobias Fiebig
TU Delft

Email: t.fiebig@tudelft.nl

Kevin Borgolte
Princeton University

Email: kevinbo@iseclab.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 16, 2019

D. Van Geest
ISARA Corporation
S. Fluhrer
Cisco Systems

October 13, 2018

Algorithm Identifiers for HSS and XMSS for Use in the Internet X.509 Public Key Infrastructure draft-vangeest-x509-hash-sigs-01

Abstract

This document specifies algorithm identifiers and ASN.1 encoding formats for the Hierarchical Signature System (HSS), eXtended Merkle Signature Scheme (XMSS), and XMSS^MT, a multi-tree variant of XMSS. This specification applies to the Internet X.509 Public Key infrastructure (PKI) when digital signatures are used to sign certificates and certificate revocation lists (CRLs).

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 April 16, 2019.

Copyright Notice

Table of Contents

1. Introduction

The Hierarchical Signature System (HSS) is described in [I-D.mcgrew-hash-sigs].

The eXtended Merkle Signature Scheme (XMSS), and its multi-tree variant XMSS^MT, are described in [RFC8391].

These signature algorithms are based on well-studied Hash Based Signature (HBS) schemes, which can withstand known attacks using quantum computers. They combine Merkle Trees with One Time Signature (OTS) schemes in order to create signature systems which can sign a large but limited number of messages per private key. The private keys are stateful; a key's state must be updated and persisted after signing to prevent reuse of OTS keys. If an OTS key is reused, cryptographic security is not guaranteed for that key.

Due to the statefulness of the private key and the limited number of signatures that can be created, these signature algorithms might not be appropriate for use in interactive protocols. While the right selection of algorithm parameters would allow a private key to sign a

virtually unbounded number of messages (e.g. 2^60), this is at the cost of a larger signature size and longer signing time. Since these algorithms are already known to be secure against quantum attacks, and because roots of trust are generally long-lived and can take longer to be deployed than end-entity certificates, these signature algorithms are more appropriate to be used in root and subordinate CA certificates. They are also appropriate in non-interactive contexts such as code signing.

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. Subject Public Key Algorithms

Certificates conforming to [RFC5280] can convey a public key for any public key algorithm. The certificate indicates the algorithm through an algorithm identifier. An algorithm identifier consists of an OID and optional parameters.

In this document, we define two new OIDs for identifying the different hash-based signature algorithms. A third OID is defined in [I-D.ietf-lamps-cms-hash-sig] and repeated here for convenience. For all of the OIDs, the parameters MUST be absent.

2.1. HSS Public Keys

The object identifier and public key algorithm identifier for HSS is defined in [I-D.ietf-lamps-cms-hash-sig]. The definitions are repeated here for reference.

The object identifier for an HSS public key is id-alg-hss-lms-hashsig:

```
id-alg-hss-lms-hashsig  OBJECT IDENTIFIER ::= { iso(1)
   member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
   smime(16) alg(3) 17 }
```

Note that the id-alg-hss-lms-hashsig algorithm identifier is also referred to as id-alg-mts-hashsig. This synonym is based on the terminology used in an early draft of the document that became [I-D.mcgrew-hash-sigs].

The HSS public key’s properties are defined as follows:

```
pk-HSS-LMS-HashSig PUBLIC-KEY ::= {
   IDENTIFIER id-alg-hss-lms-hashsig
   KEY HSS-LMS-HashSig-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

HSS-LMS-HashSig-PublicKey ::= OCTET STRING
```

[I-D.ietf-lamps-cms-hash-sig] contains more information on the contents and format of an HSS public key.

2.2. XMSS Public Keys

The object identifier for an XMSS public key is id-xmss:

```
id-xmss  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmss(13) 0 }
```

The XMSS public key’s properties are defined as follows:

```
pk-xmss PUBLIC-KEY ::= {
   IDENTIFIER id-xmss
   KEY XMSS-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

XMSS-PublicKey ::= OCTET STRING
```

The format of an XMSS public key is is formally defined using XDR [RFC4506] and is defined in Appendix B.3 of [RFC8391]. In particular, the first 4 bytes represents the big-ending encoding of the XMSS algorithm type.

2.3. XMSS^MT Public Keys

The object identifier for an XMSS^MT public key is id-xmssmt:

```
id-xmssmt  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmssmt(14) 0 }
```

The XMSS^MT public key’s properties are defined as follows:

```
pk-xmssmt PUBLIC-KEY ::= {
   IDENTIFIER id-xmssmt
   KEY XMSSMT-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

XMSSMT-PublicKey ::= OCTET STRING
```

The format of an XMSS^MT public key is is formally defined using XDR [RFC4506] and is defined in Appendix C.3 of [RFC8391]. In particular, the first 4 bytes represents the big-ending encoding of the XMSS^MT algorithm type.

3. Key Usage Bits

The intended application for the key is indicated in the keyUsage certificate extension.

If the keyUsage extension is present in an end-entity certificate that indicates id-xmss or id-xmssmt in SubjectPublicKeyInfo, then the keyUsage extension MUST contain one or both of the following values:

nonRepudiation; and
digitalSignature.

If the keyUsage extension is present in a certification authority certificate that indicates id-xmss or id-xmssmt, then the keyUsage extension MUST contain one or more of the following values:

nonRepudiation;
digitalSignature;
keyCertSign; and
cRLSign.

[I-D.ietf-lamps-cms-hash-sig] defines the key usage for id-alg-hss-lms-hashsig, which is the same as for the keys above.

4. Signature Algorithms

Certificates and CRLs conforming to [RFC5280] may be signed with any public key signature algorithm. The certificate or CRL indicates the algorithm through an algorithm identifier which appears in the signatureAlgorithm field within the Certificate or CertificateList. This algorithm identifier is an OID and has optionally associated parameters. This section identifies algorithm identifiers that MUST be used in the signatureAlgorithm field in a Certificate or CertificateList.

Signature algorithms are always used in conjunction with a one-way hash function.

This section identifies OIDs for HSS, XMSS, and XMSS^MT. When these algorithm identifiers appear in the algorithm field as an AlgorithmIdentifier, the encoding MUST omit the parameters field. That is, the AlgorithmIdentifier SHALL be a SEQUENCE of one component, one of the OIDs defined below.

The data to be signed (e.g., the one-way hash function output value) is directly signed by the hash-based signature algorithms without any additional formatting necessary. The signature values is a large OCTET STRING. This signature value is then ASN.1 encoded as a BIT STRING and included in the Certificate or CertificateList in the signature field.

4.1. HSS Signature Algorithm

The ASN.1 OIDs used to specify that an HSS signature was generated on a SHA-256 or SHA-512 hash of an object are, respectively:

```
hss-with-SHA256  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) hss(12) 2 }

hss-with-SHA512  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) hss(12) 1 }
```

[I-D.ietf-lamps-cms-hash-sig] contains more information on the contents and format of an HSS signature.

4.2. XMSS Signature Algorithm

The ASN.1 OIDs used to specify that an XMSS signature was generated on a SHA-256 or SHA-512 hash of an object are, respectively:

```
xmss-with-SHA256  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmss(13) 2 }

xmss-with-SHA512  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmss(13) 1 }
```

The format of an XMSS signature is is formally defined using XDR [RFC4506] and is defined in Appendix B.2 of [RFC8391].

4.3. XMSS^MT Signature Algorithm

The ASN.1 OIDs used to specify that an XMSS^MT signature was generated on a SHA-256 or SHA-512 hash of an object are, respectively:

```
xmssmt-with-SHA256  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmssmt(14) 2 }

xmssmt-with-SHA512  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmssmt(14) 1 }
```

The format of an XMSS^MT signature is is formally defined using XDR [RFC4506] and is defined in Appendix C.2 of [RFC8391].

5. ASN.1 Module

For reference purposes, the ASN.1 syntax is presented as an ASN.1 module here.

```
-- ASN.1 Module

Hashsigs-pkix-0 -- TBD - IANA assigned module OID

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS
  PUBLIC-KEY
  FROM AlgorithmInformation-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0)
    id-mod-algorithmInformation-02(58)}
;
```

```
   --
   -- HSS Signatures
   --

   -- HSS Object Identifiers

   hss-with-SHA256  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) hss(12) 2 }

   hss-with-SHA512  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) hss(12) 1 }

   -- HSS Signature Algorithms

   sa-hssWithSHA256 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER hss-with-SHA256
      PARAMS ARE absent
      HASHES { mda-sha256 }
      PUBLIC-KEYS { pk-HSS-LMS-HashSig }
      SMIME-CAPS { IDENTIFIED BY hss-with-SHA256 } }

   sa-hssWithSHA512 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER hss-with-SHA512
      PARAMS ARE absent
      HASHES { mda-sha512 }
      PUBLIC-KEYS { pk-HSS-LMS-HashSig }
      SMIME-CAPS { IDENTIFIED BY hss-with-SHA512 } }

   -- Note that pk-HSS-LMS-HashSig is defined in
   -- AlgorithmInformation-2009

   --
   -- XMSS Keys and Signatures
   --

   -- XMSS Object Identifiers

   id-xmss  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmss(13) 0 }
```

```
   xmss-with-SHA256  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmss(13) 2 }

   xmss-with-SHA512  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmss(13) 1 }

   -- XMSS Signature Algorithms and Public Key

   sa-xmssWithSHA256 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER xmss-with-SHA256
      PARAMS ARE absent
      HASHES { mda-sha256 }
      PUBLIC-KEYS { pk-xmss }
      SMIME-CAPS { IDENTIFIED BY xmss-with-SHA256 } }

   sa-xmssWithSHA512 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER xmss-with-SHA512
      PARAMS ARE absent
      HASHES { mda-sha512 }
      PUBLIC-KEYS { pk-xmss }
      SMIME-CAPS { IDENTIFIED BY xmss-with-SHA512 } }

   pk-xmss PUBLIC-KEY ::= {
      IDENTIFIER id-xmss
      KEY XMSS-PublicKey
      PARAMS ARE absent
      CERT-KEY-USAGE
         { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

   XMSS-PublicKey ::= OCTET STRING

   --
   -- XMSS^MT Keys and Signatures
   --

   -- XMSS^MT Object Identifiers

   id-xmssmt  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmssmt(14) 0 }
```

```
   xmssmt-with-SHA256  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmssmt(14) 2 }

   xmssmt-with-SHA512  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmssmt(14) 1 }

   -- XMSS^MT Signature Algorithms and Public Key

   sa-xmssmtWithSHA256 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER xmssmt-with-SHA256
      PARAMS ARE absent
      HASHES { mda-sha256 }
      PUBLIC-KEYS { pk-xmssmt }
      SMIME-CAPS { IDENTIFIED BY xmssmt-with-SHA256 } }

   sa-xmssmtWithSHA512 SIGNATURE-ALGORITHM ::= {
      IDENTIFIER xmssmt-with-SHA512
      PARAMS ARE absent
      HASHES { mda-sha512 }
      PUBLIC-KEYS { pk-xmssmt }
      SMIME-CAPS { IDENTIFIED BY xmssmt-with-SHA512 } }

   pk-xmssmt PUBLIC-KEY ::= {
      IDENTIFIER id-xmssmt
      KEY XMSSMT-PublicKey
      PARAMS ARE absent
      CERT-KEY-USAGE
         { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

   XMSSMT-PublicKey ::= OCTET STRING

   END
```

6. Security Considerations

6.1. Algorithm Security Considerations

The cryptographic security of the signatures generated by the algorithms mentioned in this document depends only on the hash algorithms used within the signature algorithms and the pre-hash algorithm used to create an X.509 certificate’s message digest. Grover’s algorithm [Grover96] is a quantum search algorithm which gives a quadratic improvement in search time to brute-force pre-image attacks. The results of [BBBV97] show that this improvement is

optimal, however [Fluhrer17] notes that Grover's algorithm doesn't parallelize well. Thus, given a bounded amount of time to perform the attack and using a conservative estimate of the performance of a real quantum computer, the pre-image quantum security of SHA-256 is closer to 190 bits. All parameter sets for the signature algorithms in this document currently use SHA-256 internally and thus have at least 128 bits of quantum pre-image resistance, or 190 bits using the security assumptions in [Fluhrer17].

[Zhandry15] shows that hash collisions can be found using an algorithm with a lower bound on the number of oracle queries on the order of 2^(n/3) on the number of bits, however [DJB09] demonstrates that the quantum memory requirements would be much greater. Therefore a pre-hash using SHA-256 would have at least 128 bits of quantum collision-resistance as well as the pre-image resistance mentioned in the previous paragraph.

Given the quantum collision and pre-image resistance of SHA-256 estimated above, the algorithm identifiers hss-with-SHA256, xmss-with-SHA256 and xmssmt-with-SHA256 defined in this document provide 128 bits or more of quantum security. This is believed to be secure enough to protect X.509 certificates for well beyond any reasonable certificate lifetime, although the SHA-512 variants could be used if there are any doubts.

The algorithm identifiers hss-with-SHA512, xmss-with-SHA512 and xmssmt-with-SHA512 are defined in order to provide 256 bits of classical security (256 bits of brute-force pre-image resistance with the signature algorithms' SHA-256 and 256 bits of birthday attack collision resistance with the SHA-512 pre-hash).

6.2. Implementation Security Considerations

Implementations must protect the private keys. Compromise of the private keys may result in the ability to forge signatures. Along with the private key, the implementation must keep track of which leaf nodes in the tree have been used. Loss of integrity of this tracking data can cause an one-time key to be used more than once. As a result, when a private key and the tracking data are stored on non-volatile media or stored in a virtual machine environment, care must be taken to preserve confidentiality and integrity.

The generation of private keys relies on random numbers. The use of inadequate pseudo-random number generators (PRNGs) to generate these values can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys, searching the resulting small set of possibilities, rather than brute force searching the whole key space. The generation of quality

random numbers is difficult. [RFC4086] offers important guidance in this area.

The generation of hash-based signatures also depends on random numbers. While the consequences of an inadequate pseudo-random number generator (PRNGs) to generate these values is much less severe than the generation of private keys, the guidance in [RFC4086] remains important.

7. Acknowledgements

Thanks for Russ Housley for the helpful suggestions.

This document uses a lot of text from similar documents ([RFC3279] and [RFC8410]) as well as [I-D.ietf-lamps-cms-hash-sig]. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

8. IANA Considerations

IANA is requested to assign a module OID from the "SMI for PKIX Module Identifier" registry for the ASN.1 module in Section 5.

9. References

9.1. Normative References

[I-D.ietf-lamps-cms-hash-sig]
Housley, R., "Use of the HSS/LMS Hash-based Signature Algorithm in the Cryptographic Message Syntax (CMS)", draft-ietf-lamps-cms-hash-sig-01 (work in progress), September 2018.

[I-D.mcgrew-hash-sigs]
McGrew, D., Curcio, M., and S. Fluhrer, "Hash-Based Signatures", draft-mcgrew-hash-sigs-13 (work in progress), September 2018.

[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>.

[RFC4506] Eisler, M., Ed., "XDR: External Data Representation Standard", STD 67, RFC 4506, DOI 10.17487/RFC4506, May 2006, <https://www.rfc-editor.org/info/rfc4506>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC8391] Huelsing, A., Butin, D., Gazdag, S., Rijneveld, J., and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC 8391, DOI 10.17487/RFC8391, May 2018, <https://www.rfc-editor.org/info/rfc8391>.

9.2. Informative References

[BBBV97] Bennett, C., Bernstein, E., Brassard, G., and U. Vazirani, "Strengths and weaknesses of quantum computing", SIAM J. Comput. 26(5), 1510-1523, 1997.

[DJB09] Bernstein, D., "Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete?", SHARCS 9, p. 105, 2009.

[Fluhrer17] Fluhrer, S., "Reassessing Grover’s Algorithm", Cryptology ePrint Archive Report 2017/811, August 2017, <https://eprint.iacr.org/2017/811.pdf>.

[Grover96] Grover, L., "A fast quantum mechanical algorithm for database search", 28th ACM Symposium on the Theory of Computing p. 212, 1996.

[RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, DOI 10.17487/RFC3279, April 2002, <https://www.rfc-editor.org/info/rfc3279>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC8410] Josefsson, S. and J. Schaad, "Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the Internet X.509 Public Key Infrastructure", RFC 8410, DOI 10.17487/RFC8410, August 2018, <https://www.rfc-editor.org/info/rfc8410>.

[RFC8411] Schaad, J. and R. Andrews, "IANA Registration for the Cryptographic Algorithm Object Identifier Range", RFC 8411, DOI 10.17487/RFC8411, August 2018, <https://www.rfc-editor.org/info/rfc8411>.

[Zhandry15] Zhandry, M., "A note on the quantum collision and set equality problems", Quantum Information & Computation 15, 7-8, 557-567, May 2015.

Authors' Addresses

Daniel Van Geest
ISARA Corporation
560 Westmount Rd N
Waterloo, Ontario N2L 0A9
Canada

Email: daniel.vangeest@isara.com

Scott Fluhrer
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sfluhrer@cisco.com

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

D. Van Geest
ISARA Corporation
S. Fluhrer
Cisco Systems

March 11, 2019

Algorithm Identifiers for HSS and XMSS for Use in the Internet X.509 Public Key Infrastructure draft-vangeest-x509-hash-sigs-03

Abstract

This document specifies algorithm identifiers and ASN.1 encoding formats for the Hierarchical Signature System (HSS), eXtended Merkle Signature Scheme (XMSS), and XMSS^MT, a multi-tree variant of XMSS. This specification applies to the Internet X.509 Public Key infrastructure (PKI) when digital signatures are used to sign certificates and certificate revocation lists (CRLs).

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.

Copyright Notice

Table of Contents

1. Introduction

The Hierarchical Signature System (HSS) is described in [I-D.mcgrew-hash-sigs].

The eXtended Merkle Signature Scheme (XMSS), and its multi-tree variant XMSS^MT, are described in [RFC8391].

These signature algorithms are based on well-studied Hash Based Signature (HBS) schemes, which can withstand known attacks using quantum computers. They combine Merkle Trees with One Time Signature (OTS) schemes in order to create signature systems which can sign a large but limited number of messages per private key. The private keys are stateful; a key's state must be updated and persisted after signing to prevent reuse of OTS keys. If an OTS key is reused, cryptographic security is not guaranteed for that key.

Due to the statefulness of the private key and the limited number of signatures that can be created, these signature algorithms might not be appropriate for use in interactive protocols. While the right selection of algorithm parameters would allow a private key to sign a

virtually unbounded number of messages (e.g. 2^60), this is at the cost of a larger signature size and longer signing time. Since these algorithms are already known to be secure against quantum attacks, and because roots of trust are generally long-lived and can take longer to be deployed than end-entity certificates, these signature algorithms are more appropriate to be used in root and subordinate CA certificates. They are also appropriate in non-interactive contexts such as code signing. In particular, there are multi-party IoT ecosystems where publicly trusted code signing certificates are useful.

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. Subject Public Key Algorithms

Certificates conforming to [RFC5280] can convey a public key for any public key algorithm. The certificate indicates the algorithm through an algorithm identifier. An algorithm identifier consists of an OID and optional parameters.

In this document, we define new OIDs for identifying the different hash-based signature algorithms. An additional OID is defined in [I-D.ietf-lamps-cms-hash-sig] and repeated here for convenience. For all of the OIDs, the parameters MUST be absent.

2.1. HSS Public Keys

The object identifier and public key algorithm identifier for HSS is defined in [I-D.ietf-lamps-cms-hash-sig]. The definitions are repeated here for reference.

The object identifier for an HSS public key is id-alg-hss-lms-hashsig:

```
id-alg-hss-lms-hashsig  OBJECT IDENTIFIER ::= { iso(1)
   member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
   smime(16) alg(3) 17 }
```

Note that the id-alg-hss-lms-hashsig algorithm identifier is also referred to as id-alg-mts-hashsig. This synonym is based on the terminology used in an early draft of the document that became [I-D.mcgrew-hash-sigs].

The HSS public key's properties are defined as follows:

```
pk-HSS-LMS-HashSig PUBLIC-KEY ::= {
   IDENTIFIER id-alg-hss-lms-hashsig
   KEY HSS-LMS-HashSig-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

HSS-LMS-HashSig-PublicKey ::= OCTET STRING
```

[I-D.ietf-lamps-cms-hash-sig] contains more information on the contents and format of an HSS public key.

2.2. XMSS Public Keys

The object identifier for an XMSS public key is id-alg-xmss:

```
id-alg-xmss  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmss(13) 0 }
```

The XMSS public key's properties are defined as follows:

```
pk-XMSS PUBLIC-KEY ::= {
   IDENTIFIER id-alg-xmss
   KEY XMSS-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

XMSS-PublicKey ::= OCTET STRING
```

The format of an XMSS public key is is formally defined using XDR [RFC4506] and is defined in Appendix B.3 of [RFC8391]. In particular, the first 4 bytes represents the big-ending encoding of the XMSS algorithm type.

2.3. XMSS^MT Public Keys

The object identifier for an XMSS^MT public key is id-alg-xmssmt:

```
id-alg-xmssmt  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmssmt(14) 0 }
```

The XMSS^MT public key's properties are defined as follows:

```
pk-XMSSMT PUBLIC-KEY ::= {
   IDENTIFIER id-alg-xmssmt
   KEY XMSSMT-PublicKey
   PARAMS ARE absent
   CERT-KEY-USAGE
      { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

XMSSMT-PublicKey ::= OCTET STRING
```

The format of an XMSS^MT public key is is formally defined using XDR [RFC4506] and is defined in Appendix C.3 of [RFC8391]. In particular, the first 4 bytes represents the big-ending encoding of the XMSS^MT algorithm type.

3. Key Usage Bits

The intended application for the key is indicated in the keyUsage certificate extension.

If the keyUsage extension is present in an end-entity certificate that indicates id-alg-xmss or id-alg-xmssmt in SubjectPublicKeyInfo, then the keyUsage extension MUST contain one or both of the following values:

nonRepudiation; and
digitalSignature.

If the keyUsage extension is present in a certification authority certificate that indicates id-alg-xmss or id-alg-xmssmt, then the keyUsage extension MUST contain one or more of the following values:

nonRepudiation;
digitalSignature;
keyCertSign; and
cRLSign.

[I-D.ietf-lamps-cms-hash-sig] defines the key usage for id-alg-hss-lms-hashsig, which is the same as for the keys above.

4. Signature Algorithms

This section identifies OIDs for signing using HSS, XMSS, and XMSS^MT. When these algorithm identifiers appear in the algorithm field as an AlgorithmIdentifier, the encoding MUST omit the parameters field. That is, the AlgorithmIdentifier SHALL be a SEQUENCE of one component, one of the OIDs defined below.

The data to be signed is prepared for signing. For the algorithms used in this document, the data is signed directly by the signature algorithm, the data is not hashed before processing. Then, a private key operation is performed to generate the signature value. For HSS, the signature value is described in section 3.3 of [I-D.mcgrew-hash-sigs]. For XMSS and XMSS^MT the signature values are described in sections B.2 and C.2 of [RFC8391] respectively. The octet string representing the signature is encoded directly in the BIT STRING without adding any additional ASN.1 wrapping. For the Certificate and CertificateList structures, the signature value is wrapped in the "signatureValue" BIT STRING field.

4.1. HSS Signature Algorithm

The HSS public key OID is also used to specify that an HSS signature was generated on the full message, i.e. the message was not hashed before being processed by the HSS signature algorithm.

```
id-alg-hss-lms-hashsig OBJECT IDENTIFIER ::= { iso(1)
    member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
    smime(16) alg(3) 17 }
```

[I-D.ietf-lamps-cms-hash-sig] contains more information on the contents and format of an HSS signature.

4.2. XMSS Signature Algorithm

The XMSS public key OID is also used to specify that an XMSS signature was generated on the full message, i.e. the message was not hashed before being processed by the XMSS signature algorithm.

```
id-alg-xmss  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmss(13) 0 }
```

The format of an XMSS signature is is formally defined using XDR [RFC4506] and is defined in Appendix B.2 of [RFC8391].

4.3. XMSS^MT Signature Algorithm

The XMSS^MT public key OID is also used to specify that an XMSS^MT signature was generated on the full message, i.e. the message was not hashed before being processed by the XMSS^MT signature algorithm.

```
   id-alg-xmssmt  OBJECT IDENTIFIER ::= { itu-t(0)
      identified-organization(4) etsi(0) reserved(127)
      etsi-identified-organization(0) isara(15) algorithms(1)
      asymmetric(1) xmssmt(14) 0 }
```

The format of an XMSS^MT signature is is formally defined using XDR [RFC4506] and is defined in Appendix C.2 of [RFC8391].

5. ASN.1 Module

For reference purposes, the ASN.1 syntax is presented as an ASN.1 module here.

```
-- ASN.1 Module

Hashsigs-pkix-0 -- TBD - IANA assigned module OID

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS
  PUBLIC-KEY, SIGNATURE-ALGORITHM
  FROM AlgorithmInformation-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0)
    id-mod-algorithmInformation-02(58)}
;

-- Object Identifiers

--
-- id-alg-hss-lms-hashsig is defined in [ietf-lamps-cms-hash-sig]
--
-- id-alg-hss-lms-hashsig OBJECT IDENTIFIER ::= { iso(1)
--     member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
--     smime(16) alg(3) 17 }

id-alg-xmss  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmss(13) 0 }

id-alg-xmssmt  OBJECT IDENTIFIER ::= { itu-t(0)
   identified-organization(4) etsi(0) reserved(127)
   etsi-identified-organization(0) isara(15) algorithms(1)
   asymmetric(1) xmssmt(14) 0 }
```

```
   -- Signature Algorithms and Public Keys

   --
   -- sa-HSS-LMS-HashSig is defined in [ietf-lamps-cms-hash-sig]
   --
   -- sa-HSS-LMS-HashSig SIGNATURE-ALGORITHM ::= {
   --     IDENTIFIER id-alg-hss-lms-hashsig
   --     PARAMS ARE absent
   --     PUBLIC-KEYS { pk-HSS-LMS-HashSig }
   --     SMIME-CAPS { IDENTIFIED BY id-alg-hss-lms-hashsig } }

   --
   -- pk-HSS-LMS-HashSig is defined in [ietf-lamps-cms-hash-sig]
   --
   -- pk-HSS-LMS-HashSig PUBLIC-KEY ::= {
   --    IDENTIFIER id-alg-hss-lms-hashsig
   --    KEY HSS-LMS-HashSig-PublicKey
   --    PARAMS ARE absent
   --    CERT-KEY-USAGE
   --       { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }
   --
   -- HSS-LMS-HashSig-PublicKey ::= OCTET STRING

   sa-XMSS SIGNATURE-ALGORITHM ::= {
       IDENTIFIER id-alg-xmss
       PARAMS ARE absent
       PUBLIC-KEYS { pk-XMSS }
       SMIME-CAPS { IDENTIFIED BY id-alg-xmss } }

   pk-XMSS PUBLIC-KEY ::= {
      IDENTIFIER id-alg-xmss
      KEY XMSS-PublicKey
      PARAMS ARE absent
      CERT-KEY-USAGE
         { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

   XMSS-PublicKey ::= OCTET STRING

   sa-XMSSMT SIGNATURE-ALGORITHM ::= {
       IDENTIFIER id-alg-xmssmt
       PARAMS ARE absent
       PUBLIC-KEYS { pk-XMSSMT }
       SMIME-CAPS { IDENTIFIED BY id-alg-xmssmt } }
```

```
   pk-XMSSMT PUBLIC-KEY ::= {
      IDENTIFIER id-alg-xmssmt
      KEY XMSSMT-PublicKey
      PARAMS ARE absent
      CERT-KEY-USAGE
         { digitalSignature, nonRepudiation, keyCertSign, cRLSign } }

   XMSSMT-PublicKey ::= OCTET STRING

   END
```

6. Security Considerations

6.1. Algorithm Security Considerations

The cryptographic security of the signatures generated by the algorithms mentioned in this document depends only on the hash algorithms used within the signature algorithms and the pre-hash algorithm used to create an X.509 certificate's message digest. Grover's algorithm [Grover96] is a quantum search algorithm which gives a quadratic improvement in search time to brute-force pre-image attacks. The results of [BBBV97] show that this improvement is optimal, however [Fluhrer17] notes that Grover's algorithm doesn't parallelize well. Thus, given a bounded amount of time to perform the attack and using a conservative estimate of the performance of a real quantum computer, the pre-image quantum security of SHA-256 is closer to 190 bits. All parameter sets for the signature algorithms in this document currently use SHA-256 internally and thus have at least 128 bits of quantum pre-image resistance, or 190 bits using the security assumptions in [Fluhrer17].

[Zhandry15] shows that hash collisions can be found using an algorithm with a lower bound on the number of oracle queries on the order of 2^(n/3) on the number of bits, however [DJB09] demonstrates that the quantum memory requirements would be much greater. Therefore a parameter set using SHA-256 would have at least 128 bits of quantum collision-resistance as well as the pre-image resistance mentioned in the previous paragraph.

Given the quantum collision and pre-image resistance of SHA-256 estimated above, the current parameter sets used by id-alg-hss-lms-hashsig, id-alg-xmss and id-alg-xmssmt provide 128 bits or more of quantum security. This is believed to be secure enough to protect X.509 certificates for well beyond any reasonable certificate lifetime.

6.2. Implementation Security Considerations

Implementations MUST protect the private keys. Compromise of the private keys may result in the ability to forge signatures. Along with the private key, the implementation MUST keep track of which leaf nodes in the tree have been used. Loss of integrity of this tracking data can cause a one-time key to be used more than once. As a result, when a private key and the tracking data are stored on non-volatile media or stored in a virtual machine environment, care must be taken to preserve confidentiality and integrity.

The generation of private keys relies on random numbers. The use of inadequate pseudo-random number generators (PRNGs) to generate these values can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys, searching the resulting small set of possibilities, rather than brute force searching the whole key space. The generation of quality random numbers is difficult. [RFC4086] offers important guidance in this area.

The generation of hash-based signatures also depends on random numbers. While the consequences of an inadequate pseudo-random number generator (PRNGs) to generate these values is much less severe than the generation of private keys, the guidance in [RFC4086] remains important.

7. Acknowledgements

Thanks for Russ Housley for the helpful suggestions.

This document uses a lot of text from similar documents ([RFC3279] and [RFC8410]) as well as [I-D.ietf-lamps-cms-hash-sig]. Thanks go to the authors of those documents. "Copying always makes things easier and less error prone" - [RFC8411].

8. IANA Considerations

IANA is requested to assign a module OID from the "SMI for PKIX Module Identifier" registry for the ASN.1 module in Section 5.

9. References

9.1. Normative References

[I-D.ietf-lamps-cms-hash-sig]
Housley, R., "Use of the HSS/LMS Hash-based Signature Algorithm in the Cryptographic Message Syntax (CMS)", draft-ietf-lamps-cms-hash-sig-07 (work in progress), March 2019.

[I-D.mcgrew-hash-sigs]
McGrew, D., Curcio, M., and S. Fluhrer, "Hash-Based Signatures", draft-mcgrew-hash-sigs-15 (work in progress), January 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>.

[RFC4506] Eisler, M., Ed., "XDR: External Data Representation Standard", STD 67, RFC 4506, DOI 10.17487/RFC4506, May 2006, <https://www.rfc-editor.org/info/rfc4506>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC8391] Huelsing, A., Butin, D., Gazdag, S., Rijneveld, J., and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC 8391, DOI 10.17487/RFC8391, May 2018, <https://www.rfc-editor.org/info/rfc8391>.

9.2. Informative References

[BBBV97] Bennett, C., Bernstein, E., Brassard, G., and U. Vazirani, "Strengths and weaknesses of quantum computing", SIAM J. Comput. 26(5), 1510-1523, 1997.

[DJB09] Bernstein, D., "Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete?", SHARCS 9, p. 105, 2009.

[Fluhrer17]
Fluhrer, S., "Reassessing Grover's Algorithm", Cryptology ePrint Archive Report 2017/811, August 2017, <https://eprint.iacr.org/2017/811.pdf>.

[Grover96]
Grover, L., "A fast quantum mechanical algorithm for database search", 28th ACM Symposium on the Theory of Computing p. 212, 1996.

[RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, DOI 10.17487/RFC3279, April 2002, <https://www.rfc-editor.org/info/rfc3279>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC8410] Josefsson, S. and J. Schaad, "Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the Internet X.509 Public Key Infrastructure", RFC 8410, DOI 10.17487/RFC8410, August 2018, <https://www.rfc-editor.org/info/rfc8410>.

[RFC8411] Schaad, J. and R. Andrews, "IANA Registration for the Cryptographic Algorithm Object Identifier Range", RFC 8411, DOI 10.17487/RFC8411, August 2018, <https://www.rfc-editor.org/info/rfc8411>.

[Zhandry15]
Zhandry, M., "A note on the quantum collision and set equality problems", Quantum Information & Computation 15, 7-8, 557-567, May 2015.

Authors' Addresses

Daniel Van Geest
ISARA Corporation
560 Westmount Rd N
Waterloo, Ontario N2L 0A9
Canada

Email: daniel.vangeest@isara.com

Scott Fluhrer
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sfluhrer@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 26, 2019

M. Sato
M. Shimaoka
SECOM IS Lab.
H. Nakajima, Ed.
Mercari R4D
July 25, 2018

General Security Considerations for Crypto Assets Custodians
draft-vcgtf-crypto-assets-security-considerations-02

Abstract

This document discusses technical and operational risks of crypto assets custodian and its security controls to avoid the unintended transactions for its customers.

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 January 26, 2019.

Copyright Notice

Table of Contents

1. Introduction

This document gives guidance as to what security measure should the crypto assets custodians consider and implement to protect the asset of its customers. The management of the secret key for the crypto assets especially has different aspects than other types of information systems and requires special attention.

This document reports especially on the appropriate management of the secret key by the crypto assets custodians to avoid the unintended transactions for its customers.

2. Scope of this document

This document discusses the threat, risk, and controls on the followings:

- Online system of crypto assets custodian that provides the custodian service to its customer (consumers and trade partners);
- Assets information (including the private key of the crypto assets) that the online system of a crypto assets custodian manages;
- Social impact that can arise from the discrepancy in the security measures that are implemented in the online system of a crypto assets custodian.

This document is applicable to the crypto assets custodians that manages the private key that corresponds to the crypto assets. It includes the organizations that outsources the key management to another organization. In such a case, the certain recommendations applies to those outsourcers.

3. 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.

4. Terminology

Terms used in this document are defined in [I-D.nakajima-crypto-asset-terminology]

5. Basic description of a model online system of a crypto assets custodian

5.1. General

In this section, a model online system of a crypto assets custodian that is used to explain the concepts and provisions in this document are explained.

5.2. A basic model of online system of a crypto assets custodian and its functional components

Followings are the basic model of a crypto assets custodian that this document deals with.

https://raw.githubusercontent.com/VCGTF/draft-crypto-assets-security-considerations/master/CryptoAssetCustodiansSystemModeling.svg

Figure 1: Basic Model of Crypto Assets Custodian

Figure 1 Basic Model of Crypto Assets Custodian

- Customer Interface
  Provides screen and input functions such as login process, account management (deposit/withdrawal instruction etc.) and trade instruction for the customers(users). Web application, API, etc.

- Customer Authentication Function
  Performs user authentication process for login to the crypto assets custodians.

- Customer Credential Database
  Manages required IDs for login and verification information related to user authentication process (f.g password verification info.).

- Customer Assets Management Function
  A group of functions to manage customer accounts. Receive instructions for deposit or withdrawal (outgoing coins) and perform processing according to the user instructions. Refer or update asset data.

- Blockchain Node
  Connects to another blockchain nodes to retrieve blockchain data.

- Incoming transaction management Function
  Checks transaction stored in blockchain and confirm whether incoming coins are involved in the specified addresses.

- Order processing function
  A group of functions that receives sales instructions from customers and performs processing related to trading of crypto assets. Refers and updates asset data based on asset data.

- Assets Database
  Manages holdings of fiat currencies and crypto assets. It does not include the private keys for signing transactions. Managed separately from the assets of the custodian for each customer.

- Transaction Singing Function

  * Transaction Generator
    Generates transactions to be sent to the blockchain based on instructions from the customer asset management system or the exchange management system.

  * Transaction Broadcaster
    Sends the signed transaction to the blockchain. Connects to nodes of the another nodes on the blockchain.

  * Transaction Signing Function
    Generates digital signatures based on the instructed transaction contents and the private signature key (with IDs and addresses).

  * Address Management
    Manages public keys with related to the private signature keys, or addresses (such as values calculated from the public keys).

  * Private Signature Key Management Function
    Manages the signing keys of the crypto assets (the keys used for the transaction signing). Sometimes it is separated into the cold-wallet as security sountermeasure. "Signature key generator" creates signature keys. The generated keys are

registered in the signature key management unit, and the public keys and addresses are registered in the address management units.

- Exchange Operation Modules
  A group of functions for custodians' administrators. Based on operations from administrators, instructs generation of generating new signature keys or transfer crypto assets.

- Operator Authentication Function
  Authenticates the administrator users.

- Operator Audit Database
  Manages verification data related to the authentication processes of the administrators.

We defined each functional element to distinguish functions logically, and do not show the actual arrangement on the actual system. For example, in our actual system, address management unit may be managed by an integrated database. Also, there are implementations with multiple functions packaged together. For example, each functional element of the transaction signature system may be integrated with the customer property management system, or the transaction signature system may be operating as another system.

When using existing implementations such as bitcoin wallet, bitcoin wallet is thought to provide the functions of transaction signature system as just one implementation as a whole. It is also conceivable that some functions are provided by a remote subcontractor as in a form in which the function of the transaction signature system is provided by an remote server.

5.3. The flow leading to the sending of the transaction

5.4. Types of keys that are used for signature and encryption

5.4.1. Type of keys

Types	Description
Signature Key	A private key for signing transactions (asymmetric key cryptography)
Verification Key	A public key for verification of transactions (asymmetric key cryptography Recipient address of transactions are unique value calculated from verification key
Encryption/decryption key for signature key	Secret key to keep confidentiality of signature key (symmetric key cryptography)
Master Seed	A seed to generate a signature key in decisional wallet

5.4.2. Flow for the key generation and the key usage

https://raw.githubusercontent.com/VCGTF/draft-crypto-assets-security-considerations/master/SignaureKeyLifeCycle.svg

Figure 2: Lifecycle of signature key, verification key and encryption/decryption key for signature key

After a pair of a signature key and a verification key (hereafter "key pair") is generated, an address to receive transactions is generated from the verification key. By notifying a sender of crypto assets this address, the sender is able to transfer the asset to the address. When the recipient transfers the asset to the other address, the original recipient signs the transaction data which includes the transfer order. Inactive state of the signature key is the state such that the signature key is stored in confidential manner in the signature key management function of Figure 1. An example of inactivation is encryption by encryption/decryption key (e.g. pass phrase), that is, the signature key is encrypted. In contrary, activation is the process to make the key usable to sign, by decrypting the inactivated key. The activation is assumed to be executed in transaction signing function if Figure 1. Activation and inactivation may be executed in an implementation of wallet, when the wallet have both functions. The signature key is not needed after its generation until execution of signing to transaction. Thus, there is a way to manage the signature key in offline manner with storing the verification key and address online (see Section 9.2.2).

5.4.3. On the usage of multiple keys

In some crypto assets system, it is recommended not to use the same key pair twice, thus it produces multiple key pairs. This feature is for preventing trace and not relevant to the business efficiency of a crypto assets custodian. However, a crypto assets custodian should manage addresses for each customer. Thus it should manage multiple key pairs for the same crypto assets.

5.4.4. On the suspension of keys

Suspension of key usage is only an operation inside a crypto assets custodian. By definition of blockchain based crypto assets system, any user cannot cancel transaction once it is made. As another case, it is difficult to revoke signature key even after the suspension of key. For example, a customer accidentally operate some crypto assets for suspended address. In such case, the suspended signature key is needed to make an reimbursement. Thus, suspension of keys should be conducted with considering such cases.

5.5. On the characteristics of crypto assets on Blockchain and distributed ledger technologies

In this sub-clause, following items will described as characteristics of crypto assets over blockchain and distributed ledger technology in preparation for Section 8 and Section 9.

5.5.1. The importance of the private key used for signing

5.5.2. Diversity of implementations

5.5.2.1. On the cryptographic algorithms used by crypto assets

5.5.3. On the possibility of the forking of the Blockchain

5.5.3.1. Rollback by reorganization

5.5.3.2. The treatment of the forked crypto assets

5.5.4. Risk on the unapproved transactions

5.5.4.1. General

5.5.4.2. The handling of the unapproved transactions

5.5.4.3. Transaction failures caused by the vulnerabilities of the implementation or the specification of the crypto assets

6. Basic objectives for the security management of crypto assets custodians

At crypto assets custodians, it is mandatory to establish, conduct, maintain and continuously improve security management. As requirements in term os security management, items described in [ISO.27001:2013] are sufficiently considered according to the business process of each crypto assets custodian. Especially, following items should be carefully considered, because a crypto assets custodian retains customer’s asset and should deal with issues specific to crypto assets.

- On stakeholders (Related to [ISO.27001:2013] Clause 4)

It is needed to consider protection of customer’s assets, as well as division of responsibility with outsourcers including security of private key management for crypto asset, and mattes by which a crypto assets custodian may give social impacts like money laundering.

- On security policy (Related to [ISO.27001:2013] Clause 5)

A crypto assets custodian should define a security policy which includes security objectives and controls described in and after clause 7. Especially, it is recommended to disclose the security policy on the management of crypto assets to customers to facilitate self evaluation.

- Continuous risk evaluation and improvement (Related to [ISO.27002:2013] Clause 6, 8, 9 and 10)

A crypto assets custodian should watch security risks of crypto assets in addition to aligning the general security management framework, because the risks change and increase due to rapid development of related technology as described in clause 5.3.2. It is especially important to continuously evaluate risk and improve security objectives, policy and controls to keep effectiveness of security controls after starting their operations.

7. Approaches to basic security controls

A crypto assets custodian should decide security objectives and controls with considering all of following viewpoints.

- Countermeasure to threat as lost, theft, leak and abuse of customer’s assets data and private key for crypto assets

- Requirements for actual business
- Compliance to laws and rules
- Social responsibilities to prevent crimes in use of crypto assets like scam and money laundering

The crypto assets custodian conducts threat analysis, vulnerability evaluation, risk evaluation and defining security objectives and controls according to its actual business and system. Security objectives and controls should be decided with considering threats and risks specific to crypto assets described in clause 5, as well as general security objectives and controls described in [ISO.27002:2013].

Followings are items of security objectives and controls described in [ISO.27002:2013].

- Information Security Policies
- Organization of information security
- Human resource security
- Assets management
- Access control
- Cryptography
- Physical and environmental security
- Operations security
- Communications security
- System acquisition, development and maintenance
- Supplier relationships
- Information security incident management
- Information security aspects of business continuity management
- Compliance

Consideration of above items is mandatory. Next close describes specific items to be considered by a crypto assets custodian.

8. Risks of Crypto Assets Custodian

8.1. About this section

The main risks to be noted as a crypto assets custodian office are roughly classified into those related to the system of the crypto assets custodian office, those caused by external factors such as a block chain outside the control of the crypto assets custodian office, and risks leading to abuse such as criminal acts Enumerate. Risks related to the system of the crypto assets custodians are organized in terms of threats and factors, actors that pose a threat. Organize in terms of possible incidents in risks caused by external factors such as block chains and risks of misuse. There may also be risks inherent in systems and operations that are different for each operator. Each business operator needs to identify the risks to be dealt with while taking into consideration the risks shown in this section, taking into consideration different systems and operations for each business operator. After that, it is desirable to determine the priority of the control measures by evaluating the impact each risk can have on the business.

8.2. Risks related to the system of the crypto assets custodian

Here are the typical risks to the information assets held by the system of the crypto assets custodian. In the basic model of Section 5.2, pay attention to the signing secret key and the asset data as an information asset which is particularly important from the viewpoint of protecting customer assets. If the signing secret key and the surrounding environment are not secure, it is also possible for a malicious person to create an illegal transaction and send it to the node in the block chain. Once an illegal transaction is sent to the node and written to the block chain it is nearly impossible to cancel the transaction. Therefore, prior measures to prevent illegal transactions being created are particularly important. It is necessary to carefully evaluate the risks related to management of signing secret keys and illegal transaction creation and to consider appropriate safety measures. In addition, it is necessary to consider the loss of the signing secret key. When the signing secret key is lost, it becomes impossible to use the Crypto Assets stored in the address corresponding to the signing private key. As for the risk concerning the signing secret key, consider the secret key for signature and the environment surrounding it based on the basic model in Section 5.2 in Section 8.2.1. As for the asset data, since the contents of data, data format, management form, and details of processing vary from custodians to custodians, this document considers the model more abstracted. Common contents of asset data to be protected include the total amount of Crypto Assets and legal currency deposited by the customer at the custodians, the amount of

Crypto Assets and legal currency held by the custodians, customer account number and Crypto Assets An address and so on are conceivable. If such asset data is illegally rewritten by a malicious one, it will also cause damage to customers and impede the work of the custodians. The asset data is discussed in Section 8.2.2. In addition to protecting important information such as transaction signature private key and asset data, it is also necessary to consider risks such as system outage so that customers can smoothly control their own assets. Risks related to system outages are discussed in Section 8.2.3. In addition to the information and risks mentioned in this section, there are risks inherent to each system of the crypto assets custodian and risks in cooperation with external business operators. It is necessary to conduct a detailed risk assessment on the actual system of the crypto assets custodian.

8.2.1. Risks related to signing keys

In the crypto assets custodian, the role and risk of the signing key are extremely large. This is not only to enable the transfer of coins but also to disappear due to the anonymity of the Crypto Asset, the property that it is impossible to revocation the signing key against leakage / theft or roll back the transaction by. In this section, we show the risk of fraudulent use that could lead to the loss of the signing key, leakage / theft, and damage of value. It also shows supply chain risk etc. when introducing Wallet related to signature key.

8.2.1.1. Risk analysis of signing secret key

Risk analysis differs depending on the assumed threats, system configuration, threat modeling, and so on. In this section, we show a case study based on the following assumption as an example. Here, the threat concerning the signature secret key and the factors that can cause the threat are assumed as follows. In addition, we assumed the following as the actor giving input to the signing secret key based on Figure 1 in Section 5.

- Threats:
  - lost
  - leakage, theft
  - fraudulent use
- Factors of Threats:

    * mis-operations
    * Legitimate users' malice
    * Spoofing
    * Intrusions from outside
    * Unintended behaviors of implementations

- Actors:
    * exchange operation modules
    * Transaction Signing modules
    * customer asset management function implementation
    * incoming coin management function implementation

Factors of threats are organized as follows.

Mis-operation: An act that an authorized user (including an administrator) of the system accidentally operated by mistake. For example, it is incorrectly supposed that an operation to coin 100,000 yen is incorrectly dispatched for 1 million yen.

Legitimate users' malice: Acts performed by a legitimate user of the system (including administrator etc) with malicious intent. For example, theft or unauthorized use of the signature private key due to internal fraud. In this case, it is the purpose of identifying acts that can be factors, and the purpose and incentive of the act are not limited here.

Spoofing: An act other than an authorized user of the system impersonating a legitimate user (more accurately impersonating some kind of operation). For example, an internal burglar without administrator privilege accesses the system with administrator authority.

Intrusions from outside: an act of an outsider accessing the system maliciously in a manner other than spoofing. For example, by using malicious intrusion from the outside by exploiting the system's vulnerability, incorporating malware into the exchanging system via targeted e-mail to the custodians administrator or the like and generating a private signature private key (or transaction creation) from the outside, Allow remote control of etc.

Unintended behaviors of implementations: The system behaves unexpectedly by the designer or operator irrespective of the intention or malice of the operation. For example, a signature private key leaks due to a bug in the exchange management system.

Of these threat factors, theft and fraudulent use are regarded as threats that can only be caused by explicit malicious factors. As a result, the possible risks for signing key to be assumed are the following:

1. Threat by lost - Risk of Unauthorized operation (with legitimate path)

    * End-user’s malice

    * Operator’s malice in Custodian

    * Spoofing to end users

    * internal frauds (spoofing to operators) - Risk of Intrusion from the outside

    * Intrusion into the transaction signing modules

    * Intrusion into the incoming coin management function (implementation)

    * Intrusion into the customer asset management function (implementation)

    * Intrusion into the exchange operation modules - Risk of System Behaviors different from human operation

    * Unintended behaviors of the transaction signing modules

    * Unintended behaviors of the incoming coin management function (implementation)

    * Unintended behaviors of the customer asset management function (implementation)

    * Unintended behaviors of the exchange operation modules - Risk of mis-operation (by human error)

    * Mis-operation of end user

    * Mis-operation of operator

2. Threat by leakage - Risk of Unauthorized operation (with legitimate path)

   * End-user’s malice
   * Operator’s malice in Custodian
   * Spoofing to end users
   * internal frauds (spoofing to operators) - Risk of Intrusion from the outside
   * Intrusion into the transaction signing modules
   * Intrusion into the incoming coin management function (implementation)
   * Intrusion into the customer asset management function (implementation)
   * Intrusion into the exchange operation modules - Risk of System Behaviors different from human operation
   * Unintended behaviors of the transaction signing modules
   * Unintended behaviors of the incoming coin management function (implementation)
   * Unintended behaviors of the customer asset management function (implementation)
   * Unintended behaviors of the exchange operation modules - Risk of mis-operation (by human error)
   * Mis-operation of end user
   * Mis-operation of operator

3. Threat by theft - Risk of Unauthorized operation (with legitimate path)

   * End-user’s malice
   * Operator’s malice in Custodian
   * Spoofing to end users

* internal frauds (spoofing to operators) - Risk of Intrusion from the outside

* Intrusion into the transaction signing modules

* Intrusion into the incoming coin management function (implementation)

* Intrusion into the customer asset management function (implementation)

* Intrusion into the exchange operation modules

4. Threat by fraudulent use - Risk of Unauthorized operation (with legitimate path)

* End-user’s malice

* Operator’s malice in Custodian

* Spoofing to end users

* internal frauds (spoofing to operators) - Risk of Intrusion from the outside

* Intrusion into the transaction signing modules

* Intrusion into the incoming coin management function (implementation)

* Intrusion into the customer asset management function (implementation)

* Intrusion into the exchange operation modules

The following sections outline each risk, and their security controls are shown in Section 9.2.2.

8.2.1.2. Risk of loss of signing secret key

The disappearance risk shown in Table 8-2 is an enumeration of events having a possibility of causing disappearance, paying attention to the input (operation instruction) to the signature secret key. As a typical risk, loss of the secret key for signature due to erroneous operation by the administrator may be considered.

8.2.1.3. Leakage / theft risk of signing private key

Intentional operation by malicious intention is essential, but leakage can be caused by negligence without malice. For this reason, leakage risk and theft risk need to be sorted out.

The leakage risk shown in Table 8-2 lists events that have the possibility of causing leakage including negligence, paying attention to the input (operation instruction) to the signing secret key. Typically, an internal criminal, leakage risk due to unintentional behavior, illegal invasion, etc. can be considered. Likewise, the theft risk enumerates events that have the possibility of being woken up by some sort of malicious intention, paying attention to the input (operation instruction) to the signature secret key. Typically, there is a risk of leakage from internal criminals or unauthorized intrusion from the outside.

Both leaks and theft are similar in terms of leakage of sensitive information to the outside, and the control measures are common. This will be described later in Section 9.3.2.

8.2.1.4. Risk of unauthorized use of signing private key

The fraudulent use risk shown in Table 8-2 is an enumeration of events having a possibility of being caused by something with a malicious intention, paying attention to the input (operation instruction) to the signature secret key. Typically, there is a risk of illegal use due to illegal invasion or impersonation from the outside.

8.2.1.5. Other related risks

Supply chain risk of hardware wallet As a product having a secret key management function, there is a so-called hardware wallet. Many hardware wallets connect to the management terminal such as PC via USB and perform key management operation from the management terminal. FIPS 140-2, etc. as a security certification of products with key management function, etc. However, since most of cryptographic algorithms handled by Crypto Assets are not subject to certification, the security of hardware wallet for Crypto Assets Unfortunately it is inevitable that the third-party accreditation system on insurance is inadequate. For this reason, while there are products with sufficient safety in the hardware wallet that generally available, it is necessary to recognize that there are products with insufficient safety. Furthermore, even with products with certain safety, safety may be impaired by being crafted on the distribution route. For example, a hardware wallet that is loaded with malware in the middle of a sales channel, even if the purchaser newly generates

a signing secret key in the hardware wallet, the attacker can use the signature secret It becomes possible to restore the key.

8.2.2. Risks related to asset data

The asset data is data for managing assets such as Crypto Assets and statutory currency held by customers and custodians. The secret key of the transaction signature shall not be included (see Section 5.2). As mentioned above, since asset data varies from custodians to custodians, we consider this as a more abstracted model in this document. Since detailed threat analysis and risk assessment need to be performed on the asset data handled by the actual custodian system, only the way of thinking is shown here. The main threats of asset data may be illegal rewriting, loss, or leakage. The factors include mis-operation by the manager, malice of the righteous person, spoofing the legitimate person, malicious intent of the outsider, unintended behavior of the system. In the example of the basic model in Section 5.2, there is a route from the exchange management system, the customer property management system, and the incoming coin determination unit. Among the threats of asset data, the following examples are considered as incidents due to illegal rewriting. A customer asset management system referring to unauthorized asset data may create an illegal transaction and flow through the normal process to the block chain (Section 8.2.1.4). For example, it is possible to rewrite the holding amount recorded in the asset data, change the Crypto Assets transfer destination address, or the like. For example, by rewriting the list of addresses associated with the customer, illegal recombination of the amount held between customers or between customers and custodians within the asset data within the custodian will be made. As a transaction in the block chain the result is not reflected, assets that a customer or custodian should have had is lost. Risks related to asset data can be thought of as a problem similar to general financial and settlement systems, but transactions can be canceled if transactions are written to the block chain as a result of fraud against asset data It is necessary to consider it on the premise that there is no property.

8.2.3. Risks related to suspension of systems and operations

The system of the crypto assets custodian office is software, hardware, network, and the like constituting the crypto assets custodian. In addition, the term "operation" refers to an operation performed manually, such as operation monitoring of a switching center system, account opening, remittance instruction, wallet deposit / withdrawal. It is conceivable that the system and work may be stopped due to various factors. Risks related to suspension of systems and operations can be regarded as problems similar to general financial and payment systems. However, the fact that the crypto

assets custodian office is connected to the Internet all the time, not the dedicated communication network at all times, that it is operating 24 hours a day, 365 days, that many crypto assets custodians are built on the public cloud infrastructure, crypto assets custodian It is necessary to consider it based on the fact that the operating situation of the place has a large effect on the crypto assets price and it tends to be an object of attack.

8.2.3.1. Risks related to network congestion

Crypto assets custodians frequently receive denial of service attacks. As a target of a denial-of-service attack, publicly-released top page, API endpoint, etc. are common, but when an attacker knows the system configuration and a business system or operation monitoring system is placed on the Internet, A case of receiving a denial of service attack may be considered.

8.2.3.2. Risk of system outage

It is conceivable that the data center, the cloud infrastructure, etc. where the system is installed are stopped, and the system and operations are stopped. The system can be stopped due to various factors such as power outage due to natural disasters and disruption of communication, large scale failure due to mistake by cloud infrastructure operator, large scale system failure due to mistake of infrastructure operation, failure of software release.

8.2.3.3. Risks related to Operator

Even if the system itself is in operation, if operations monitoring and tasks of personnel responsible for the operation are hindered, the work will be suspended. For example, regular inspections of power facilities at operation bases, disruption of transportation means due to catastrophic changes and strikes, protest activities and rush of press reporters may hinder the entrance and exit of buildings and halt operations. In addition, when personnel are using the same transportation method or participating in the same event, there is a risk that many personnel can not operate due to the same accident such as traffic accident or food poisoning.

8.2.3.4. Regulatory risks

In countries where the crypto assets custodian office is stipulated, licensing system or registration system, business may be suspended due to business improvement order, business suspension order, deletion of registration etc etc.

8.3. Risks from external factors

Even if the systems and operations of the crypto assets custodian are appropriately operated, if the blockchain / network in which the crypto assets operates and the Internet infrastructure supporting the connection between the nodes are attacked, On the other hand, the service cannot be continued or the transaction can not be handled properly.

8.3.1. Risks related to the Internet infrastructure and authentication infrastructure

8.3.1.1. Internet routing and name resolution attacks

An attacker interferes with routing of the Internet such as route hijacking and domain name resolution (Domain Name Service), hindering the reachability to the custodians and guiding it to a fake custodian office or block chain It is possible to intentionally cause a branch by hindering synchronization. This method can be considered not only by malicious attackers but also by ISP etc. based on instructions from the government.

8.3.1.2. Attacks on Web PKI

Many crypto assets custodian offices provide services on the Web, and TLS and server certificates are used for user authenticity verification and encryption of sites. When a certificate authority issuing a server certificate is attacked, it may be possible to impersonate the site, or if the server certificate is revoked, the service cannot be provided.

8.3.1.3. Attack on Messaging

By intervening with an e-mail or other messaging system, an attacker can fraud and block SMS / MMS of e-mails and mobile phones used for interaction with users and delivery of one-time passwords. When a user’s message is fraudulent, it is possible to log in as a user and reset the password.

8.3.2. Troubles due to block chains of crypto assets

8.3.2.1. Crypto assets blockchain fork, split

8.3.2.2. Re-org of Blockchain by 51% attack and selfish mining

8.3.2.3. Compromise of hash function and cryptographic algorithm

8.3.2.4. Inadequate consensus algorithm

By misusing a bug in the agreement algorithm, fake transaction information is sent to a specific node and disguised as to whether or not to send money to the counter party. For example, in the case of the MtGOX case in 2014, double payment attacks abusing Transaction Malleability occurred frequently in piggybacking.

8.3.3. Risks arising from external reputation databases

8.3.3.1. Establishment of Bank Account Elimination and Freezing

As a part of AML / CFT, there are cases where banks refuse to open accounts related to the work of the crypto assets custodian office, the risk that the bank accounts will be frozen, such as receiving guidance from the regulatory authorities or accident is there. In the event that the account is frozen, the business of depositing and withdrawing a legal currency with the user as a crypto assets custodian office is stopped.

8.3.3.2. Crypto assets address

As a part of AML/CFT, when another crypto assets custodian trader remittles to another crypto assets address, there are cases where it is checked whether the remittance destination address does not correspond to a high risk transaction. In the case where the address of the hot wallet of the custodian office is registered as a problematic address, it is assumed that the custodian of the crypto assets with such a service can not be performed smoothly. Since it is common in many cases that criminal remit crypto assets stolen for disturbance to a known address, there is a risk that the address of the hot wallet of the crypto assets custodian office will be classified as a high risk customer by mistake.

8.3.3.3. Filtering and blocking for Web sites

There is a risk that the URL of the crypto assets custodian office is filtered by the network or blocked by the ISP so that the user can not access it. Also, if it is recognized as a malware distribution site or the like, there is a risk that it will not be displayed as a search result or it will be impossible to browse from the browser.

8.3.3.4. Email

As a measure against spam mails, most of mail servers provide mail rejection based on reputation and classification function of spam mails. If the e-mail delivered by the custodian is judged as spam, it may be impossible to contact the user.

8.3.3.5. Appraisal of a smartphone application

Depending on the platform, there are cases where handling of crypto assets by the application is restricted. If the examination of the smartphone application is dropped, the user may not be able to download the smartphone application for accessing the custodian and may be unable to use the service.

8.4. Financial Crime Risks

8.4.1. Risk of being abused in criminal acts

8.4.1.1. Identity Theft

8.4.1.2. Scams

8.4.1.3. Ransom

8.4.1.4. Utilization for transactions in the dark market

8.4.2. Anti Money Laundering

8.4.3. Counter Financing of Terrorism

9. Consideration on security controls at crypto assets custodian

9.1. General

In this clause, considerations from the ISMS viewpoint in implementing security controls to crypto assets custodians for risks described in clause 8 are described. They included issues caused by specific characteristics to crypto assets. This clause also describes relationship to [ISO.27001:2013] and [ISO.27002:2013] for each consideration.

9.2. Consideration on security controls

Direction of the information security management

Objectives of security management at a crypto assets custodian contain secure protection of customer’s asset, compliance to business

requirements, laws and rules, and realization of social responsibility. Security policies and execution statements derived from such objectives are recommended to be publicly available for consumers, business partners, auditor and regulators to help their judge.

9.2.1. Security controls on signing keys

9.2.1.1. Basics of key management

In general, followings are required in management of private cryptographic keys.

- o They should be Isolated from other informational assets. Rigorous access control is mandatory.
- o Limit the number of access to private keys as minimum as possible.
- o Be prepared for unintentional lost of private keys.

Followings are three basic security control to realize above. Additional security controls specific to crypto assets custodians are described in and after sub-clause Section 9.2.2.

1. State management of private keys
   As described in Figure 2, a private key has one of multiple states, and it may be active or inactive state in its operation. The private key should be in active state when it is used for signing or decryption. It is recommended to enforce to input some secret information to activate an inactive private key. This makes keep the inactive private key away from abuse, if the adversary does not have the secret information. This method ensure security of the private key against leakage and lost. It is also recommended to minimize the term of activation to limit the risk of abuse as minimum as possible. Unnecessary activation of secret key increases the risk of abuse, leakage and theft, though keeping the activation state is efficient from business viewpoint. On the other hand, frequent activation/ inactivation may give impact to business efficiency. It is important to consider the trade-off between the risk and business efficiency and provide clear key management policy to customers.
2. Administrator role separation and mutual check-and-balance
   It is fundamental form of operation of a critical business process which uses private key to perform cryptographic operations by multiple party to prevent internal frauds and errors. For example, by setting isolated rights on digitally signing and approval to go into the area of signing operation, it

becomes difficult for single adversary to give an malicious digital signature without known by the third party. Additionally, the enforcement of attendance of other person is effective security control to internal frauds and mis-operations.

3. Backup of private key
Lost of private key makes signing operations by using the key impossible any more. Thus backup of private key is an important security control. On the other hand, risks of leakage and theft of backup keys should be considered. It is needed to inactivate the backup key.

9.2.1.2. Backup

Backup is the most fundamental and effective measure against lost of signing key. On the other hand, there are risks of leakage and lost of backup device. These risks depend on the kind backup device, thus security controls on such devices should be considered independently. Followings describe typical backup devices and leakage/theft risks associated with them.

o Cloning to tamper-resistant cryptographic key management device

If a signing key is managed by a tamper-resistant key management device (device X) and X has cloning function, cloning the key to another device Y is the most secure way to backup the key, where the cloning function is the technique to copy the key with keeping confidentiality to other devices than X and Y. The implementation of the function is recommended to be evaluated/certified by certification program like CMVP or FIPS 140. Note that, the cryptographic algorithms supported by such tamper-resistant key management devices are limited and all crypto assets systems can utilize it, but it is one of the most secure way of backup.

o Backup to storage for digital data

Here, it is assumed to backup keys to storage like USB memory and DVD. There are two types of operations; one is backup data is stored in movable devices in offline manner, the other is backup data is stored in online accessible manner. If the device is movable, the possibility of steal and lost increases, thus the device should be kept in a cabinet or a vault with key, and the access control to such cabinet/vault should be restrict.

Of the backup storage is online, risks of leakage and theft should be assumed as same as the key management function implementation inside the crypto assets custodian. In general, the same security control is recommended to such backup storage. If there is some additional

operation, for example the backup device is inactivated except for the time of restore, the security control may be modified with considering operation environment. When it is not avoided the raw key data is be outside of the key management function implementation, the custodian should deal with the problem of remained magnetics.

- Backup to paper

There is a way to backup keys in offline manner, to print them to papers as a QR code or other machine readable ways. It is movable than storage for digital data and easy to identify. There remains some risk of leakage and theft by taking a photo by smartphone and so on.

9.2.1.3. Offline management

There is a type of offline key management (as known as "cold wallet") which isolates private keys from the system network to prevent leakage and theft caused by intrusion.

In this case, some offline operation is needed to make the system use the key. Examples are, keys are usually stored inside a vault and connected to the system only when it is utilized, and USB memory is used to data transportation between an online system and an offline system. If there is not explicit approval process in the offline operation for key usage, anyone cannot stop malicious transaction. That is, this solution can prevent lost and theft, however, an explicit approval process is needed to prevent abuse of keys.

9.2.1.4. Distributed management

It is also a good security control to distribute the right to use private key to multiple entity. There are two examples; division of secret key and multi-signature.

- Division of secret key Division of the signing key to multiple parts, then manage them by multiple isolated system is an effective measure to protect the keys against leakage and theft. This document does not recommend a specific technique, but recommends to implement this control based on a certain level of security evaluation like secret sharing scheme. In that case, secure coding and mounting penetration test are needed to eliminate the implementation vulnerabilities. This method is also effective to backup devices.

- Multi-Signature
  This is a signature scheme which requires multiple isolated signing keys to sign a message. It is effective to protect each

key hold by an entity and signing mechanisms. There are many different realization of multi-signature and they are different according to specific crypto assets system. Thus, consideration on preparing multiple implementations and their interoperation is need when a crypto assets custodian operate multiple crypto assets.

9.2.1.5. Other issues

In this sub-clause, following topics will be described.

- Security of wallet implementation
- Monitoring of private key access
- Log audit

9.2.2. Security controls at crypto assets custodian

Following security controls addition to kay management in sub clause Section 9.2.2 will be described in this sub clause.

- Organization of information security
- Category and management of assets
- Access control
- Communications security
- Operations security
- Physical and environmental security
- System acquisition, development and maintenance
- Consideration in user authentication and providing API
- Flow control in transaction generation

Above controls are described with aligning to [ISO.27002:2013] and considering security controls specific to crypto assets custodians. The structure of this sub clause aligns the structure of [ISO.27002:2013].

[ISO.27001:2013]
International Organization for Standardization, "Information technology -- Security techniques -- Information security management systems -- Requirements", ISO/IEC 27001:2013, October 2013, <https://www.iso.org/standard/54534.html>.

[ISO.27002:2013]
International Organization for Standardization, "Information technology -- Security techniques -- Code of practice for information security controls", ISO/IEC 27002:2013, October 2013, <https://www.iso.org/standard/54533.html>.

[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>.

[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/info/rfc8174>.

[I-D.nakajima-crypto-asset-terminology]
Nakajima, H., Kusunoki, M., Hida, K., Suga, Y., and T. Hayashi, "Terminology for Crypto Asset", draft-nakajima-crypto-asset-terminology-00 (work in progress), July 2018.

Acknowledgements

Thanks to Masanori Kusunoki, Yasushi Matsumoto, Natsuhiko Sakimura, Yuji Suga, Tatsuya Hayashi, Keiichi Hida and other members of the Security Working Group of Virtual Currency Governance Task Force.

Authors' Addresses

Masashi Sato
SECOM Co., Ltd. Intelligent System Laboratory
SECOM SC Center
8-10-16 Shimorenjaku
Mitaka, Tokyo 181-8528
JAPAN

Email: satomasa756@gmail.com

Masaki Shimaoka
SECOM Co., Ltd. Intelligent System Laboratory
SECOM SC Center
8-10-16 Shimorenjaku
Mitaka, Tokyo 181-8528
JAPAN

Email: m-shimaoka@secom.co.jp

Hirotaka Nakajima (editor)
Mercari, Inc. R4D
Roppongi Hills Mori Tower 18F
6-10-1 Roppongi
Minato, Tokyo 106-6118
JAPAN

Email: nunnun@mercari.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 20, 2020

M. Sato
M. Shimaoka
SECOM IS Lab.
H. Nakajima, Ed.
Mercari

June 18, 2020

General Security Considerations for Cryptoassets Custodians
draft-vcgtf-crypto-assets-security-considerations-06

Abstract

This document discusses the technical and operational risks of cryptoassets custodians and its security controls to avoid the unintended transactions for its customers.

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 December 20, 2020.

Copyright Notice

Table of Contents

1. Introduction

This document gives guidance as to what security measure should the cryptoassets custodians consider and implement to protect the asset of its customers. The management of the signature key for cryptoassets especially has different aspects than other types of information systems and requires special attention.

This document reports especially on the appropriate management of the signature key by the cryptoassets custodians to avoid the unintended transactions for its customers.

The document organizes recommendations for considering security as a purpose of protecting users' assets by operators of cryptoassets custodians. Among the assets to be protected, in particular, the signature key of the cryptoassets has a different characteristic from the conventional information system and needs attention. Particular emphasis is given to points that should be kept in mind for the cryptoassets custodians to properly manage the signature key and to prevent illegal transactions that the customer does not intend.

The basic model of the cryptoassets custodians system covered in this document is shown in Section 5. A system in a form different from this basic model, for example, a system where an operator manages a signature key provided by a user (e.g. online wallet), is handled in another complementary document or later revision of this document.

2. Scope of this document

An operator covered by this document is a cryptoassets custodian that manages the signature key used in the cryptoassets. Including the case where the management of the signature key is entrusted to another custodians operator. In that case, even for operators entrusted with the management of signature key, a considerable part of the recommendation indicated in this document is considered to apply.

This document includes considerations on threats and risks for the following subjects.

- A cryptoassets custodians system that provides cryptoassets custodians work to customers (consumers and other exchanges)
- Assets information managed by the cryptoassets custodians system (including the signature key of the cryptoassets)
- The social impact which can be exerted by imperfect security measures of the cryptoassets custodians system

This document does not focus on the following items.

- Security measures for information systems used by daily operations by custodians operators
- Security measures against blockchains that provide the mechanism of cryptoassets and distributed ledger itself
- Operator’s own management risk
- Specific requirements on separation of assets of customers and custodians/exchanges

3. 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.

4. Terminology

Terms used in this document are defined in [I-D.nakajima-crypto-asset-terminology]

5. Basic description of a model system of a cryptoassets custodian

5.1. General

In this section, a model of a cryptoassets custodians system that is used to explain the concepts and provisions in this document are explained.

5.2. A basic model of cryptoassets custodians system and its functional components

Followings are the basic model of a crypto assets custodian that this document deals with. A basic model of cryptoassets custodians system is shown on Figure 1.

Figure 1: Basic Model of Cryptoassets Custodians system

- Interface (Web Application, APIs) Provides screen and input functions such as login process, account management (deposit/ withdrawal instruction etc.) and trade instruction for the customers(users). Web application, API, etc.

- Customer Authentication Function Performs user authentication process for login to the cryptoassets custodians.

- Customer Credentials Manages required IDs for login and verification information related to user authentication process (e.g. password verification info.).

- Customer Assets Management Function A group of functions to manage customer accounts. Receive instructions for deposit or withdrawal (outgoing coins) and perform processing according to the user instructions. Retrieve or update assets data.

- Blockchain Node Connects to another blockchain nodes to retrieve blockchain data.

- Incoming Coin management Function Checks transaction stored in blockchain and confirm whether incoming coins are involved in the specified addresses. Update an assets database according to the transaction from blockchain.

- Order processing function A group of functions that receives orders from customers and performs processing related to trading of cryptoassets. Retrieves and updates assets data based on the orders.

- Assets Database Manages holdings of fiat currencies and cryptoassets. The database does not include the private keys for transaction signature. Assets are managed separately from the assets of the custodian for each customer.

- Transaction Singing Function

  * Transaction Generator Generates transactions to be sent to the blockchain based on instructions from the customer asset management function or the custodians operation function.

  * Transaction Broadcaster Broadcasts the signed transaction to the blockchain. Connects to other nodes on the blockchain.

  * Transaction Signing Function Generates digital signatures based on the instructed transaction contents and the signature key (or its IDs and its addresses).

  * Address Management Manages public keys with related to the signature keys, or addresses (such as values calculated from the public keys).

  * Signature Key Management Function Manages the signature keys of the cryptoassets (keys used for signing the transaction). Sometimes signature keys are separately stored into the cold-wallet as security countermeasure.

  * Signature key generator Generates signature keys. The generated keys are registered in the signature key management function, and the public keys and addresses are registered in the address management function.

- Custodians Operation Modules A group of functions for custodians' operators or administrators. Based on operations from administrators, the module instructs generating new signature keys or transferring cryptoassets.

- Operator Authentication Function Authenticates the administrators.

- Operator Audit Database Manages auditing data related to the authentication of the administrators.

We defined each functional element to distinguish functions logically, and do not show the actual arrangement on the actual system. For example, in our actual system, the address management unit may be managed by an integrated database. Also, there are implementations with multiple functions packaged together. For example, each functional element of the transaction signature system

may be integrated with the customer property management system, or the transaction signature system may be operating as another system.

When using existing implementations such as bitcoin wallet, bitcoin wallet is thought to provide the functions of the transaction signature system as just one implementation as a whole. It is also conceivable that some functions are provided by a remote subcontractor as in a form in which the function of the transaction signature system is provided by a remote server.

5.3. The flow leading to the sending of the transaction

- Deposit Phase

  1. Customers send fiat to custodian’s bank account.

  2. Custodians shall confirm to receive fiat, and shall update assets database to reflect customer asset information.

- Input coin phase

  1. Customer transfer cryptoassets to the address instructed by custodians. The transfer shall be made by cryptoassets wallet for the customer such as tools or services (other custodians or Web wallet)

  2. Custodians shall confirm cryptoassets has been transferred to the address instructed and shall update the asset database to reflect asset information of the customer.

- Trading phase

  1. Customer access to interfaces to make instructions.

  2. Instructions to transfer shall be processed by custodians operations functions. The result of trade processed by custodians operations functions shall be updated into the asset database.

- Instructions to output coins from customers

  1. Customers access to interface and instruct it to transfer its cryptoassets to other address. (Instruct to output coins)

  2. Instructions to output coins shall be processed by customer assets management functions. Transaction generator shall make transaction messages based on instructions such as receive address or amount of cryptoassets.

3. Transaction messages shall be added a digital signature by transaction signing functions.

4. Transaction messages with a digital signature shall be delivered to all nodes on blockchain by transaction broadcaster.

- Instruction to transfer from Customer Assets Management Function

1. Administrator instructs to send cryptoassets to address through the interface of Management Functions. For Example, it may send between address managed inside custodians.

2. Instructions to transfer shall be processed on Management Function, and shall be processed as described 2 to 4 on "output coin". Transactions with digital signature shall be delivered to all nodes on blockchain.

5.4. Types of keys that are used for signature and encryption

5.4.1. Type of keys

Types	Description
Signature Key	A private key for signing transactions (asymmetric key cryptography)
Verification Key	A public key for verification of transactions (asymmetric key cryptography). Recipient address of transactions are the unique value calculated from verification key
Encryption/decryption key for signature key	Secret key used to keep signature key (symmetric key cryptography) confidential / protected
Master Seed	A seed, e.g. random number, to generate a signature key in deterministic wallet

Table 1: Type of keys

5.4.2. Flow for the key generation and key usage

Figure 2: Lifecycle of signature key, verification key and encryption/decryption key for signature key

After a pair of keys (signature & verification, hereafter "key pair") is generated, an addressed to receive transaction is derived from the verification key. By providing a sender of digital assets this address, the sender is able to transfer one or more assets to this address. When the recipient transfers the assets to another address, the original recipient signs the transaction data which includes the transfer order.

A signature key is considered to be in an inactive state when it is stored in a confidential manner (ie. cannot be directly used to sign), for example within the key management function in Figure 1. An example of how to set a signature key in an inactive state is to encrypt the signature key using an encryption key (ie. passphrase).

The opposite process of decrypting the signature key will return the key inactive state. The activation of a key is assumed to be executed within the transaction signing function in Figure 1.

Activation and deactivation of keys is part of the function set of certain wallets.

The signature key is not needed after its generation until a transaction has to be signed. Therefore this allows for the store and manages signature keys offline while keeping the verification key and addresses online (See: Section 7.3.6.2).

Figure 3: Lifecycle of signature key, verification key and encryption/decryption key for signature key in case of deterministic wallet

The deterministic wallet is a mechanism that generates one master seed and generates multiple signature key pairs from that master seed. It is possible to regenerate each signature key pair from the master seed by backing up the master seed and restoring it. On the other hand, if the master seed is stolen, the crypto assets which are managed by all signature key pairs (and addresses) derived from the master seed may be stolen. Also, if the master seed is lost, all signature key pairs will not be able to be regenerated.

As an extension of the deterministic wallet, there is a hierarchical deterministic wallet (HD wallet). In the case of HD wallet, a master key pair is created from the master seed, and child key pairs are derived from the master key pair. Furthermore, descendant key pairs can be derived from the child key pairs in a hierarchical manner. Since the child key pair can be created from the parent key pair, it is not necessary to access the master seed when generating the child key pair. The implementation of hierarchical key pair generation

depends on the signature algorithm, and some currencies cannot be realized in principle. Although this document refers mainly to the management of the signature keys in the security control measures, the master seed also needs security management equal to or higher than the signature keys.

5.4.3. On the use of multiple keys

There are some cases to use cryptoassets where one user uses one address, one user uses multiple addresses. The number of addresses and pairs depends on the number of cryptassets and method of management. For example, cryptoassets that can contain tags related to the transaction such as Ripple and NEM, cryptoassets custodian may distinguish customers by each tag if custodian uses one address. On the other hand, cryptoassets that cannot contain any tags for transactions, custodians have to make addresses for each customer, so the number of addresses and key pairs would be increased. It is considered to use multiple addresses and key pairs by risk evaluation with not only a variety of cryptoassets (e.g., Bitcoin, Ethereum, etc.) but also management by the hot wallet and cold wallet.

It is recommended not to reuse key pair for general. But it is focussed for anonymous transactions by private use, so this is not suitable for custodians from viewpoint of efficiency and practicality. Cryptoassets custodians shall make effective controls considered by risk evaluations and control objective.

5.4.4. On the suspension of keys

Even if Figure 2 indicates operations on operations of custodian, cancellation of transaction cannot be made for cryptoassets. Also, it is difficult to revoke the signature key after suspension of using keys. For example, it may happen to input coins to the address user has suspended to use. To return coins to the sender, custodian needs a signature key for the suspended address. cryptoassets custodians shall assume those cases, and shall consider about revoking signature keys carefully.

5.5. Characteristics of cryptoassets in blockchain and distributed ledger

5.5.1. About this section

In the handling of cryptoassets using blockchain / distributed ledger, there are things to emphasize and different characteristics compared with general information systems and usages of private/ encryption keys. In considering the risk assessment described in

Section 6 and the security requirements and measures based thereon, it is necessary to pay attention to these characteristics.

5.5.2. Importance of signature keys

As described in Section 5.3, by signing transactions using the signature keys, it is possible to instruct the transfer of the values of cryptoassets to other addresses. Once this transaction is written to the block or ledger data and the transfer of the values of cryptoassets is approved it is difficult to revert it or to invalidate the transfer by revocation procedure etc. This property is in contrast to taking time until the remittance gets caught or the process can be canceled during remittance and be reassembled, even if it requires complicated administrative procedures in the process of remittance, and illegal remittances occur. In addition, when the private signature keys have vanished in the cryptoasset scheme, there will be a case that the cryptoasset held by the address corresponding to the signature private key is impossible to transfer to the other. In cryptoassets having such irreversible nature, it must pay attention to the theft, fraudulent use and disappearance of the signature secret key.

5.5.3. Diversity of implementations

There are various cryptoassets including Bitcoin. The specifications also vary widely from cryptoassets to cryptoassets. For example, there are differences in the using of encryption algorithms, hash functions, the methods of generating/spreading transactions, and wallet implementations to protect the signature key(s), and so on. Due to these differences in specifications, effective countermeasures for a specific cryptoasset may not be able to be carried out under the specification of another cryptoassets. And also, from the current feaver trends of the cryptoassets, the appearance of new cryptoassets and the speed of functional expansion and specification change of existing cryptoassets mechanisms are very fast.

5.5.3.1. Cryptographic algorithm of cryptoassets

There are cases that new cryptographic algorithms in cryptoassets that are not sufficiently reviewed for security may be adopted. In ordinary use cases of cryptography technology, designers often use cryptographic algorithms that are scientifically verified, mathematically proved secure, and approved by official authorities/ agencies, however, cryptoassets designers are often adopting "immature and unverified" cryptographic algorithms. This means that it takes time to archive provable security for algorithms and approve by official authorities/agencies, while in the blockchain where competition and evolution are remarkable, the maturity level is low

as technology, and differentiation and blocking from other cryptoassets. It must be optimized the technology specific to the chain. These algorithms are likely to have no properly reviewed implementation, or the risk of a vulnerability being discovered later and compromising (compared to mature algorithms) is high.

5.5.4. Possibility of blockchain forks

In the blockchain using Proof-of-Work and the like typified by bitcoins, a state such as a temporary fork of a chain due to specification change of software or a single chain of branched chains (re-organization) can arise. Also, as another case, due to the division of the developer community, blockchains are divided from the point of time and sometimes operated as separate cryptoassets. In the real world, there are various forks, it may be difficult to respond to all of them, and it should be consider countermeasures according to the risks.

5.5.4.1. Rolling back due to re-organization

If the chain is discarded due to a reorganization, the history of transactions contained in the discarded chain will be lost. In that case, the transaction on the block discarded within the reorganization period may not be reflected in the main-chain.

5.5.4.2. Handling forks of cryptoassets

As in the case of bitcoins and ether symbols, blockchains are divided and sometimes managed as another cryptoassets (here, called a fork coin). The fork coin is also derived from the same software as the original cryptoassets and uses the same technology and compatible technology (A description that incorporates the case where different technologies are adopted for a fork is necessary). In addition, the chain until just before splitting has exact identical data. By using its functionality, it becomes possible to attack, for example, replay attacks. A replay attack is an attack in which transactions used in the original cryptoassets are retransmitted to the sender of the transaction at the fork coin chain and the fork coin is illegally acquired as a result. In this kind of replay attacks, countermeasures such as monitoring of the transaction sender, for fork coin chain, measures to be sent before transactions that return coins to their own other address are required.

In addition, if a fork coin occurs in the cryptoassets held by the exchanger, there is also a problem that the fork coin is not returned to the user unless the fork coin is assigned to the user of the exchanger in the exchange system.

5.5.5. Risks for Unauthorized Transactions

5.5.5.1. About this section

Just by sending the transaction instructing the transfer of the coins(assets) to the node of the blockchain does not instantly reflect the cryptoassets transfer. In order for a transaction to be approved, it is stored in a block created every decided period and needs to be accepted by the majority of mining nodes. It may be difficult to confirm that the transaction has been approved for the following reasons.

5.5.5.2. Handling unapproved transactions

In a cryptasset using a distributed ledger, there are a variety of cryptoassets (such as Bitcoin, Ethereum, etc.) that the transaction sender sends transactions with a transaction fee. This transaction fee is acquired by the miner who creates the block, and the higher the transaction cost, It is easy to store in blocks (transactions are easily approved immediately). If the cost of the transaction sent from the cryptoassets custodian to the blockchain is low, it may take times to approve the transaction, or there is a possibility that the time will expire without being approved. Besides the case due to the transaction fee, the temporary chain fork as in Section 5.5.4.1 can be occurred that the transaction that should have been approved once becomes the unapproved state and the dual spend of cryptoassets. In usage scenes where cryptoassets transfer is required immediately, such as payments in real stores, it may be difficult to take time to confirm the approval of the transaction, and it is necessary to assume the risk of unauthorized transactions.

5.5.5.3. Transaction failure due to vulnerabilities from cryptoassets specifications and implementations

Although it is not exactly the case of unauthorized transactions, there was a vulnerability called transaction malleability as a past case of bitcoins. With this vulnerability, if the node relaying the transaction is malicious, it is also possible to make transactions illegally manipulate, thereby making it impossible to find the transaction stored in the block (make it impossible to search by transaction ID). There is also the possibility of an attack that makes a duplicate by requesting transmission of the cryptoassets again from the counterparty by making the approved transaction appear as not approved. This attack is performed after sending the transaction to the nodes, so it is characteristic that the sender can not take measures beforehand before sending. Regarding transaction malleability, it is now possible to avoid it by using SegWit in bitcoins. However, as a lesson from this case, effective defense

measures cannot be made effective only with the cryptoassets custodian that becomes the sender or receiver of the cryptoassets with respect to faults and threats due to another vulnerability of bitcoins and other cryptoassets.

6. Risks of cryptoassets custodian

6.1. About this chapter

Below in this section, some risks custodian shall consider for the system and for foreign factor outside of control from custodian such as blockchain is described. The risks for systems in custodians are listed as a threat, factor, and actor may cause threat. The risks for foreign factor outside of control from custodian such as blockchain are listed from the incident. Some risks may be caused by property or quality described in Section 5.5.

On the other hand, there are some risks based on operations or systems implemented by each custodians. Custodians shall pick up risks to deal with control to refer these risks with understanding with system or operation of custodian. Custodians shall evaluate impacts may be affected by risks and shall decide controls and its priority.

6.2. Risks of cryptoassets custodian system

In this section, major risks regarding information asset which cryptoassets custodian system holds are listed. Among the fundamental model shown in Section 5, the signature key and asset data are focused as significant information asset to protect customers asset.

The attacker may be able to broadcast a malicious transaction to nodes of distributed-ledger after generating the transaction if the signature key and surrounding environment are not safe.

Withdrawing transaction is almost impossible once the malicious transaction has been broadcasted and built into the blockchain. Therefore, prior countermeasures to prevent generating malicious transaction are essential.

Moreover, consideration of a loss of signature key is also essential. Cryptoassets stored in the address associated with the signature key become unavailable in a case where the signature key has been lost.

Risk regarding the signature key including the signature key and surrounding environment are mentioned in Section 6.2.1 based on Figure 1.

In this document, the model is described as more abstract as the content of data, data format, management model or details of processing regarding asset data varies among custodians. Record such as client assets (both cryptoassets and fiat currency), assets of custodians(both cryptoassets and fiat currency), clients' account information, or address of cryptoassets is listed as common content of asset data subject to protection. Manipulation to those asset data caused by the attacker results in damage to client assets or affect to the custodians' operation. Risks related to assets data are discussed in Section 6.2.2.

Risks of system outage MUST be considered concerning availability which allows clients to control their assets in addition to the protection of important information such as the signature key or assets data. Risks of system control are discussed in Section 6.2.3.2.

In addition to information or risks mentioned in this section, system specific risks varied among cryptoassets custodian or risks regarding external contractor MUST be considered. Detailed risk analysis MUST be performed against the actual system of the cryptoassets custodian.

6.2.1. Risks related to signature keys

Both role and risks of signature keys are extremely large on cryptoasset exchange. Signature keys enable to transfer coins, but it comes from properties of difficulties for revocation of lost, leakage, stolen, and rollback transaction. Some risks about signature keys are listed in this section. In addition, risks about supply chain related to risks install wallets handles signature keys.

6.2.1.1. Risk analysis related to signature key

Risk analysis may depend on threats assumption, the structure of the system, and threats model, the results for each custodians shall be different. Some case studies are described in this section.

Threats for signature keys and its actors are assumed as listed below. And actors are assumed as the input of signature key in Figure 1.

- Threats:
  - Loss
  - Leakage, Theft
  - Unauthorized Use

- Factors of Threats:
  - Error in operation
  - Maliciousness (of legitimate person)
  - Spoofing (for legitimate person))
  - Malicious intentions of outsiders
  - Unintended behavior (system)
- Actors:
  - Custodians operation modules
  - Transaction Signing modules
  - Customer assets management function
  - Incoming Coin management function

Factors of threats are organized as follow.

Error in operation: A human error caused by an authorized user (including an administrator) during operation of the system. For example, the expected operation was to withdraw coin equivalent to 100,000 JPY. But, the actual operation is withdrawing coin equivalent to 1,000,000 JPY.

Malicious acts by authorized person: An act committed with malice by an authorized person (including an administrator). For example, theft or unauthorized use of the signature key by the insider. Purpose or incentive of the act is not concerned.

Spoofing(of authorized person): Impersonation with a stolen credential of an authorized person. For example, the order to sell/buy/transfer cryptoassets by an external attacker impersonating a client; the malicious order of transfer or generation/signing of a transaction through access to the system with the legitimate operator/administrator credential by an unauthorized insider. Especially, theft and abuse of credential upon an account registration by impersonating a legitimate user MUST be considered. Note: Impersonation which is not caused by theft of legitimate user/ authorized person's credential (e.g., Privilege escalation) are mentioned in "malicious acts by outsiders."

Malicious acts by outsiders: Access or operation to the system by outsiders with malicious purpose excluding spoofing. (e.g., external unauthorized access by exploiting a vulnerability; remote access to the system which enables outsiders to operate to the signature key or generate a transaction by a targetted attack to an administrator of the custodians' system.)

Unintended behavior: An unintended behavior of the system regardless of intention or malice. (e.g., leakage of the signature key caused by bugs of the system, generation of a transaction including an incorrect amount of assets regardless of operation.)

Theft and unauthorized use are threats that can only be caused by a clear malicious factor. Risks to be considered as a result of threats are listed in Table 2. Please note that theft and unauthorized use could happen in a case where multiple factors such as an error in operation or unintended behavior have occurred. (e.g., insertion of backdoor that transmits a signature key or tampers a signing order to the transaction in conjunction with a specific legitimate operation.) This case can be covered in countermeasures of theft or unauthorized use.

Risk	Factor	Loss	Leakage	Theft	Unauthorized Use
Illegal operation(Route is legitimate)	End user's malicious operation	Y	Y	Y	Y
	Malicious operation by the administrator of customer assets management function	Y	Y	Y	Y
	Impersonation to end users	Y	Y	Y	Y
	Insider impersonating an administrator	Y	Y	Y	Y
Intrusion from outside	Intrusion into Tx signing function	Y	Y	Y	Y

	Intrusion into incoming coin management function	Y	Y	Y	Y
	Intrusion into customer asset management function	Y	Y	Y	Y
	Intrusion into custodian operation function	Y	Y	Y	Y
Incorrect behavior is different from operation instruction	Unintended behaviors of Tx signing function	Y	Y	-	-
	Unintended behaviors of incoming coin management function	Y	Y	-	-
	Unintended behaviors of customer asset management function	Y	Y	-	-
	Unintended behaviors of custodian operation function	Y	Y	-	-
Human error	Error in operation by end user	Y	Y	-	-
	Error in operation by administrator of	Y	Y	-	-

	customer asset management function					

Table 2: List of possible risks for the signature key, Y means applicable risk exists, - means no applicable risk exists

The following sections outline each risk. The control measures corresponding to each risk are shown in Section 7.3.

6.2.1.2. Risk of loss of signature key

Risks listed below are an event which causes loss of the signature key from a viewpoint of input to the signature key such as order or operation.

As a typical event, the loss of the signature key caused by human error in operation by the administrator of the custodians' system may be considered.

6.2.1.3. Leakage and theft risk of signature key

In most case, theft is caused by the operation of a malicious person. By contrast, leakage could happen by error or fault not requiring the malice. Therefore, the risk of theft and the risk of leakage MUST be separately considered.

The risks of leakage shown in Table 2 are lists of the event which potentially causes leakage of the signature key including the leakage caused by error/fault regarding the input to the signature key such as an order or an operation. For example, an internal criminal, unintentional behavior of the system and intrusion to the system.

Likewise, the risks of theft are lists of the event which potentially causes the theft of the signature key by a malicious person. For example, an internal criminal and intrusion to the system.

Regarding the leakage of sensitive information to the outside, both leakage and theft are similar, and the countermeasures are the same. The countermeasures are discussed in Section 7.3.6.

6.2.1.4. Risks of unauthorized use of the signature key

The risks of unauthorized use shown in Table 2 are lists of the event which causes unauthorized use by a malicious person. For example, spoofing of the authorized person and intrusion to the system.

Unauthorized use of the signature key could be caused by unauthorized operation of pre-processes of an unsigned transaction at transaction signing function in addition to the direct unauthorized use of the signature key. Following example shows unauthorized use at an early stage of the process.

- o A destination address of cryptoassets or amount of assets is manipulated due to tampering of software at transaction signing function. The tamper disables designed validation process at the transaction signing function.

- o A destination address of cryptoassets or amount of assets is manipulated due to tampering of the unsigned transaction generated by transaction generator. Besides, an unauthorized transaction has generated and given to the transaction signing function.

- o A destination address of cryptoassets or amount of assets is manipulated due to tampering of software at transaction generator. An unsigned transaction has generated with an unauthorized direct operation to transaction generator.

- o An incorrect amount or incorrect destination address of cryptoassets has transmitted from custodian operation function through transaction generator due to an internal crime, error in operation, or spoofing of the identity by the administrator.

- o Assets database has tampered in a case where the operation/order to transaction generator refers to the assets database. (See: Section 6.2.2)

As shown in the above example, the attacker is able to obtain cryptoassets without attacking to the signature key illicitly. In particular, countermeasures MUST be considered in a case where the system automates each process.

Security control measures to the signature key MUST be performed. Moreover, security control measures to the entire custodian's system MUST be performed against these complex risks. Security control measures are discussed in Section 7.

6.2.1.5. Other risks

6.2.1.5.1. Supply chain risk of hardware wallet

Hardware-wallet is known to have a function to manage signature keys. In most hardware-wallet, key administration is done on an administrative terminal connecting via USB such as PC.

Cryptographic module validation program for products having a cryptographic key management function such as FIPS 140-2 are provided. However, most of the cryptographic algorithms used in cryptoassets are not covered by those validation programs. Therefore, third-party safety validation program subject to hardware-wallet for cryptoassets is not well provided. For this reason, the users of hardware-wallet MUST understand that safety level of the hardware-wallet available at a market differs among the product.

Furthermore, the safety could be threatened by tampering the product during distribution channel even though the product has a certain level of safety in the factory. For example, hardware-wallet tampered in a distribution channel to have a malware enables the attacker to restore the signature key generated by a legitimate owner without acquiring the hardware-wallet.

6.2.2. Risks related to assets data

Assets data is data to manage an amount of cryptoassets/fiat currencies held by clients or custodian itself. The signature key for transaction signing is not recorded in the assets data. (See: Section 5.2)

As mentioned earlier, assets data differs among the custodians, an abstracted model is used in this section. In this section, a brief thought is given since detailed threat assessment and risk analysis MUST be performed against assets data of the actual custodians' system.

Major threats to the assets data are unauthorized manipulation, loss, and leakage. The factors are an error in operation by the administrator, malicious acts by the authorized person, spoofing of the authorized person, malicious acts by outsiders, and unintended behavior of the system.

In a case of the basic model shown in Section 5.2, attack surfaces are custodian operation function, assets database, and incoming coin management function.

Following example shows the incidents caused by unauthorized manipulation among the risks to assets data.

- An incident that the malicious transaction generated by assets database which refers manipulated assets data has broadcasted through a legitimate process. (See: Section 6.2.1.4)

- Unauthorized manipulation to a number of assets stored in asset data between clients and/or between clients and custodians by

tampering a list of cryptoassets address linked to clients. This enables losing assets of clients or custodians without broadcasting the transaction to the blockchain.

Risks of assets data may be considered as risks of system in financial service and settlement service. However, countermeasures to the incident that transaction(s) has merged into blockchain as a result of unauthorized manipulation to the assets data MUST be considered with an understanding that transaction broadcast to the network is irreversible.

6.2.3. Risks of suspension on system and operation

Cryptoassets custodians' systems are composed of software, hardware, networks. Operations are classified as monitoring, opening an account, an order of transfer, deposit/withdrawal of (crypto/fiat) assets from the wallet, and any operations by the operator. The system may be suspended due to various factors.

Cryptoassets custodians' system tends to be a subject to the attack due to following: the systems are connected to the Internet for 24 hours 365 days, not by the leased line, many of the systems are deployed on cloud services, prices of cryptoassets are effected from operating condition of the cryptoassets custodians. Therefore, countermeasures to the attack MUST be considered.

6.2.3.1. Risks related to network congestion

Cryptoassets custodians may be attacked by DoS and traffic flooding. In general, targets of attack are a top page of the Website, API endpoint, etc., but operation and monitoring system deployed on the Internet may be a target of DoS attack in a case where the attacker acquired the information of the system beforehand.

6.2.3.2. Risks of system suspension due to infrastructure

System and operation may be suspended in a case data center or cloud infrastructure where custodian's system is deployed are suspended. The system may be suspended due to various factors such as blackout and disruption of communication due to acts of nature, due to operation failure by cloud or infrastructure, and failure of software release.

6.2.3.3. Risks of system suspension due to the operator

Even if the system is in operation, there is a possibility that the service may be suspended if operation monitoring and the activities of the operator in charge of work are hindered. For example, there

is a possibility that business would be suspended due to various factors such as periodic inspection of power supply facilities at operational sites, disruption of transportation by disaster, strikes, and obstruction of building access by protest activities and rush of reporters. There are also risks that many personnel cannot operate due to the same reasons, such as using the same transportation method, participating in the same event, or traffic accident or food poisoning.

6.2.3.4. Regulatory risks

In countries where the cryptoassets custodian is defined by law and should be licensed or registered, operations may be suspended by order of business improvement, operation suspends, deletion of license or registration issued by the authority.

6.3. Risks from external factors

Even if a cryptoassets custodian performs its operation appropriately, the cryptoassets custodian could not continue the service or might not execute transactions when encountering attack to the blockchain network and/or the network infrastructure connecting each node.

6.3.1. Risks related to the Internet, Web PKI, and users environment

6.3.1.1. Attack to Internet routing and DNS

Attackers can lower the reachability to cryptoassets custodians, lure a user into the fake cryptoassets custodian, or fork deliberately by preventing the synchronization of the blockchain, through the intervention in routing or DNS, such as BGP hijacking. These methods might be used by not only malicious attackers, ISPs acting governments order.

6.3.1.2. Attack to Web PKI

Most cryptoassets custodians provide their services on the Web and use TLS and server certificates for authenticity and confidentiality of their website. When the certification authority issuing their certificates encounter an attack, it yields to enable to spoofing the cryptoassets custodians' website. When the certificate is revoked, the cryptoassets custodian might not be able to provide own service.

6.3.1.3. Attack to messaging systems

Attackers can swindle or block the e-mail and SMS using for delivering One-Time Password, through the intervention in messaging systems such as SMS or e-mail. When a users message is swindled, attackers can log in as the spoofed user or reset the password.

6.3.1.4. Risks related to users environment infection

When a user's environment such as PC and smartphone is infected by malware, any secrets such as credentials in the environment might be swindled.

6.3.2. Risks related to cryptocurrency blockchain

6.3.2.1. Split or fork of blockchain

A distributed ledger might be forked by specification changes without consensus in the developers community. There are two cases around the fork; one is that the transaction before the fork is executed and recorded in both ledgers after the fork, another one is that the transaction before the fork is executed and recorded in only one ledger.

6.3.2.2. Blockchain Re-organization caused by 51% attack or selfish mining

When a block which is committed in the past is discarded, the transaction included in the discarded block might be rolled back. The transaction included in the discarded block is disabled, and cryptoassets or fiat money paid in compensation for the transaction might be swindled.

6.3.2.3. Compromising cryptographic algorithm and hash function

Improvement of performance of computing power and the discovery of effective attack might cause being compromisation of the cryptographic algorithm and hash function.

6.3.2.4. Inadequate blockchain specification and implementation

In the cryptoassets Lisk, there were implementations in which the timestamp value of the transaction allowed implementation of numerical value input in a range not permitted by the internal database so that each node could not process the transaction and block generation stopped[LISK-ISSUE:2088]. This issue was fixed within several hours after the problem occurred and the node updated the client software, and the network was sequentially recovered.

However, the transactions could not be processed in the blockchain for a certain period.

There are cases that token value collapses due to inadequate implementations of smart contract. In Beautychain Token (BEC) of ERC20 token issued on Ethereum, there is a vulnerability that causes overflow in the smart contract, so there is an attack which derives greatly exceeded tokens over the upper limit, then the worth of BEC was collapsed. [CVE-2018-10299]

6.3.2.5. Rapid changes in the hashrate

When the hash rate increases or decreases rapidly, it might take a very long time for generating blocks using the remaining node.

6.3.3. Risks from external reputation

6.3.3.1. Bank account frozen

Banks might freeze an account of cryptoassets custodians operation, by the guidance of regulatory as a countermeasure for AML/CFT, or by some accidents/incidents. This freeze results in a suspending a deposit/withdraw operation of clients fiat assets.

6.3.3.2. Address of cryptocurency

As countermeasures for AML/CFT, other cryptoassets custodian Y might assess whether the destination address of cryptoassets custodian X have a high deal risk when a user of Y transfers some assets to the address of X. If an address of X is blacklisted, the transaction between X and Y might not be executed smoothly.

Since criminals often transfer the stolen "cryptoassets" to unmalicious third party’s address for disrupting investigation, the address might be involuntarily categorized as high-risk.

6.3.3.3. Filtering or blocking website

Users might not be able to access cryptoassets custodian when its URL is filtered out by network operators or is blocked by ISPs. When a cryptoassets custodian’s website is recognized as used for malware distribution, its URL might not be appeared in search results or not be able to browse in the browser.

6.3.3.4. Email

Most mail servers provide a filtering service or a classifying service based on reputation, as countermeasures for spam mail. If the e-mail from the cryptoassets custodian is recognized as spam mail, the custodian might not be able to contact the user.

6.3.3.5. Appraisal of a smartphone application

Application delivery platform might limit applications from handling cryptoassets. When the application provided by a cryptoassets custodian could not be approved by the platforms, a user cannot download the application for access to the custodian, and cannot use the services.

6.3.3.6. ID theft

There is some case where the attacker acts malicious instruction spoofing as a user, for example: - list based attack, - theft of ID, password or other credentials, by a malware infection, and - theft of API access token.

The distinctive purposes of spoofing are: - theft of fiat currency or cryptoassets by unauthorized withdrawals, - money laundering by cashing cryptoassets with an account in the name of other people, and - profit shifting by market manipulation by unauthorized buy and sell cryptoassets.

7. Considerations of security controls on Cryptoassets Custodians

7.1. General

Below is a basis of security controls about risks written in Section 6.

To promote understanding and coverage, all security controls in this chapter are followed by below: [ISO.27001:2013] , [ISO.27002:2013]. There are some specific considerations for Cryptoassets Custodians to follow ISOs. Especially, the organization shall consider for strong controls to manage signature keys for cryptoassets backed by assets.

Other security controls are expected to be referred to similar operations by the financial sector. Security controls should be included concrete content from results of risk analysis and vulnerability diagnosis. Threats of cybersecurity are changing, reviews of security controls according to situations are important. Articles below are expected to describe contents by references and completion of description.

7.2. Basis for consideration about security management

There are some standards of requirement for information security, [ISO.27001:2013] and [ISO.27002:2013]. Cryptoassets Custodians shall refer the requirement or guidance of these standards and consider security controls needed and shall establish, implement, maintain and continually improve security management. Cryptoassets Custodians has data of customers asset, self asset, customer information, signature keys. Those shall be protected from leakage, loss, tampering, and misuse. Cryptoassets Custodians shall consider about risks of lost assets by foreign factors such as blockchains or network, suspension of system, and shall act properly. Cryptoassets Custodians shall mainly consider about security management described below:

- Interested parties (from "4. Context of organization", [ISO.27001:2013]) To protect assets of cryptoassets custodian's customer. Division of responsibility between outsourced and cryptoassets custodians such as management of signature keys for cryptoassets. Impact of business such as money laundering shall be considered from another viewpoint.

- Policy (from "5. Leadership", [ISO.27001:2013]) Cryptoassets custodians shall establish an information security policy that includes information security objectives and controls. Information security policy shall be disclosed so that customers can browse.

- Continual improvement and risk assessment (from "6. Planning", "8. Operation", "9. Performance evaluation", and "10.Improvement", [ISO.27001:2013]) As described in Section 6.3.2, numbers of cryptoassets have been developed and its speed of evolving is rapid, Cryptoassets Custodians shall monitor security risks about cryptoassets in addition to information security management applied in general. Cryptoassets Custodians shall review and improve security controls according to the situation.

7.3. Considerations about security controls on Cryptoassets custodians

Cryptoassets Custodians shall determine information security objectives and controls from the viewpoint listed below:

- Risk treatment options to prevent from loss, steal, leakage, misuse of secret keys used for cryptoassets, customer data, and customer asset.

- Compliance with business

- Compliance with legal and contractual requirements

There are some considerations described in Section 7.2 about security controls based on system risks at Cryptoassets Custodians. There is a guidance for security controls as [ISO.27002:2013], Cryptoassets Custodians shall refer it to design and / or identify security controls. Section 7.3.1 to Section 7.3.13 below are followed to [ISO.27002:2013] and describe items to be especially noted in the virtual currency exchange system.

7.3.1. Information security policies

Information security policies shall be defined to follow Section 5 on [ISO.27002:2013]. Information security objectives on Cryptoassets Custodians shall include conservation of customer's asset, requirements of the business, compliance with legal and contractual requirements, social responsibilities. Information security policies shall contain policies about access controls (on Section 7.3.5), cryptographic controls (on Section 7.3.6), operations security (on Section 7.3.8), and communications security (on Section 7.3.9) .

7.3.2. Organization of information security

Cryptoassets custodians shall follow "6. Organization of information security" on [ISO.27002:2013], and shall establish a management framework to implement and operate information security. Cryptoassets custodians shall consider about threats such as an illegal acquisition of signature keys or illegal creation of transaction carefully. Segregation of duties shall be fully examined to manage signature keys for signing or to permit create transactions.

7.3.3. Human resource security

Cryptoassets custodians shall follow section "7. Human resource security" on [ISO.27002:2013]. To examine and evaluate security controls, cryptoassets custodians shall deploy human resources with expertise not only in information security applied in general but also in cryptoassets and blockchain technology. All employees may handle assets and shall receive appropriate education and training and regular updates in organizational policies and procedures.

7.3.4. Asset management

Cryptoassets custodians shall follow section "8. Asset management" on [ISO.27002:2013]. Cryptoassets custodians shall contain any pieces of information to manage assets, and information and asset of the customer such as the signature key. Cryptoassets custodians shall determine controls suitable for risks to follow this section if cryptoassets custodians operate hardware wallets. To protect assets

of customers, cryptoassets custodians shall separate assets into customers and custodians to follow compliances with accounting.

7.3.5. Access control

Cryptoassets Custodians shall follow section "9. Access Controls" on [ISO.27002:2013].

Users are separated into 2 parties; Permitted operators and administrators within outsourced, and customers. Some considerations for operators and administrators are written in Section 7.3.5.1, and for customer is written in Section 7.3.5.2

7.3.5.1. Access controls for operators and administrators

There are some cases for operators and administrators.

- Operators and administrators for custodians system. They will command to create keys or to transfer funds by software or terminal.

- Administrators to maintain hardware, OS, databases, and middleware.

Management measures of signature keys such as activate, backup, restore are described on Section 7.3.6. Cryptoassets custodian shall be carried out to assign authority to operate properly and shall set access control. Access controls shall be include authorize and permit to connect custodians system from remote, authorize for external service if using as functions for cryptoasset custodians, authorize as a user for OS and databases, permit to enter and leave facilities systems or terminals installed. There are some factors to permit access: Only office hours or predetermined hours, Only IP addresses assigned for specific terminals, Confirm by credentials to connect from operators or terminals predetermined. Cryptoassets custodians shall consider for access control policies by roles or authorities of operators and administrators for each system. Access control shall be set the minimum to run functions or software permitted for operators or administrators, not only for applications.

Any damage may be happened by miss or injustice operations on transferring assets or managing signature keys as described Section 6.2. To deter these threats, Confirmation of or approval by multiple operators or multiple administrators shall be needed on important operations such as transferring assets and operations for the signature key. Cryptoassets custodians shall not concentrate duties for one operator or administrator, decentralize of duties for multiple operators or administrators shall be needed.

7.3.5.2. Access control for customers (user authentication / API)

- Strict personal identification on setup account The account shall be set up by strict personal identification, and account information shall be sent to the person itself. For example, personal identification shall be operated by an identification document issued by the public organization and shall be sent a letter to the address without forwarding. Personal identification shall be carried out in accordance with relevant laws, regulations, treaty such as FATF. Replacement of pictures on an identification document or falsification of attribute information is typical treats for personal identification. In order to operate personal identification strictly, it shall be carried out to verify by software or visual check and verify by an electric method such as signature that is hard to falsification.

- Managing credential and multi-factor authentication For user authentication, it is expected to prevent from spoofing and internal injustice by installing risk-based authentication on not normal access (such as a characteristic of terminal or route, and different time slot from usual) and multi factor authentication on spoofing by leakage of single credential. It is NOT recommended to deliver one-time-password by unprotected transmission line as email because there is a risk of impersonation or fraud on the transfer route. Confirming telephone number by SMS was valid for verifying owner and reachability, but that has been RESTRICTED by NIST, so personal authentication technology such as possession identification and transaction authentication technology should be applied. SMS may be one factor used to recovery account, but not measure to confirm the existence and authenticate.

- Multi-factor authentication, risk-based authentication It shall be carried out to register customer and set access controls strictly to avoid defraud customer funds, changing to fiat and money laundering by spoofing customers.

- Confirmation of intention according to the risk of operation To be consistent with the convenience of customers and safety of service, It shall be considered to make a different level to authenticate by risks of customer’s operation. For example, low-risk operations such as display balance of account or details of trade may be allowed by single-factor authentication, but update transactions such as trading coins or changing address or account shall be authenticated by an additional factor. In addition, operations it may cause damages such as output coin or order of fiat transaction shall be ordered to confirm by additional authenticate or to confirm intention by an operator.

- Data preservation on deleting an account Cryptoassets custodians shall implement system be able to rollback after erasing for a certain period if customer stated spoofed or unauthorized access. Cryptoassets custodians shall delete the account if requested from a customer, but they also shall consider about risks that attacker spoofed to customer requests to delete the account.

- Signature key preservation on discontinue addresses Signature keys linked to an account shall not be deleted even if the address of cryptoassets has no value. On a prediction for the general cryptoassets customer is allowed to send assets for any addresses and not technically prevented to send, signature keys for wallet stopped to use shall be taken back up for reuse because the possibility of receive coins to the address exists.

- Consideration for supplying APIs To set access control for operations by a customer, it shall be considered about not only operations of dialogue operations on the web but also APIs connecting from the application by smartphone and from external systems. For providing APIs, It shall be implemented to consider cases that are difficult to get explicit approval from customer. It shall comply with best practices shared in the industry based on the attack risk peculiar to API. For reference, it may be followed to Financial API by OpenID Foundation.

7.3.6. Security controls on signature keys

It SHALL conform to Section 10 "Cipher" of [ISO.27002:2013].

Particularly, some security controls for the signature key, an issue specific to cryptoassets custodians, are closely related to the controls in other sub-sections in this section (e.g., Section 7.3.5).

Amount of cryptoassets in Hot Wallet MUST be limited to a minimum amount and isolate their remain assets to another secure place, e.g., Cold wallet. The minimum amount means the amount which can be temporarily paid within the time it takes to withdraw the assets from the secure place. Custodian can be refunded to the customers from the remain assets even if the assets in Hot Wallet leaks.

Custodians MUST choose an appropriate cryptographic technology that has been evaluated its security by the third party in accordance with the purpose of use, as with general information systems. Also, they MUST decide the life cycle of a signature key and MUST implement and operate appropriate controls.

7.3.6.1. Basics of Signature Key Management

In general, followings are required in the management of private keys including signature keys.

- They should be isolated from other informational assets. Rigorous access control is mandatory.
- Limit the number of access to signature keys as minimum as possible.
- Be prepared for unintentional lost of signature keys.

Followings are three basic security control to realize above. Additional security controls specific to crypto assets custodians are described in and after sub-sections Section 7.3.6.2.

1. State management of signature keys As described in Figure 2, a signature key has one of the multiple states generally, and it may be an active or inactive state in its operation. The signature key MUST be in an active state when it is used for signing (or decryption). It is recommended to enforce to input some secret information to activate from the inactive signature key. This makes keeping the inactive signature key away from abuse if the adversary does not have the secret information. This method ensures also the security of the signature key against leakage and lost. It is also recommended to minimize the term of activation to limit the risk of abuse as minimum as possible. Unnecessary activation of the signature key increases the risk of abuse, leakage, and theft, though keeping the activation state is efficient from a business viewpoint. On the other hand, frequent activation/inactivation may give impact to business efficiency. It is important to consider the trade-off between the risk and business efficiency and provide clear key management policy to customers.
2. Administrator role separation and two-person rule It is a fundamental form of operation of a critical business process which uses the signature key to perform cryptographic operations by multiple parties to prevent internal frauds and errors. For example, by setting separated privilege on digitally signing and approval to go into the area of signing operation, it becomes difficult for the single adversary to give a malicious digital signature without known by the third party. Additionally, the enforcement of the two-person rule is effective security control to internal frauds and misoperations.

3. Backup of a signature key Lost of the signature key makes signing operations (by using the key) impossible any more. Thus backup of the signature key is an important security control. Since lost of the signature key makes signing operations impossible any more, backup of the signature key is an important security control. On the other hand, risks of leakage and theft of backup keys MUST be considered. It is needed to inactivate the backup keys. Additionally, monitoring the blockchain whether to perform the outgoing-coin from that address to detect the inappropriate backup and the illegal-use of little-used address.

7.3.6.2. Offline Key Management

There is a type of offline key management (as known as "cold wallet") which isolates signature keys from the system network to prevent leakage and theft caused by the intrusion.

Figure 4: Example of offline signature key management

In this case, it REQUIREs some kind of offline operations to make the system use the signature key.

Examples are a) it requires to move a signature key from the vault and to connect to the online system, b) input/output between online system and offline (key management) system does perform through a kind of storage, such a USB Flash Drive.

If there is not an explicit approval process for the signature key used in the offline operation, anyone cannot stop the malicious transaction. That is, for achieving this solution can prevent abuse, loss, and theft of signature keys, an explicit approval process is needed for this solution.

7.3.6.3. Privilege separation of signature keys (Authorization process)

Both privilege separation and two-person control of signature key management are effective as shown in Section 7.3.6. In addition, there is multi-signature as a typical scheme for blockchain[BIP-0010][BIP-0011]. Multi-signature REQUIREs an authorization process with multi-stakeholders, and it is achieved by signing with the signature keys managed by each stakeholder. Each stakeholder MUST verify other signatures technically if exists, and MUST validate the practical consistency of the transaction.

Authorization process with multiple stakeholders can expect for a general countermeasure for malicious generation of a transaction.

Note, however, that security controls for the leakage and/or loss of the signature key are still needed.

Since a multi-signature scheme is provided by software, its logic and implementations are varied with some blockchain. e.g., multi-signature in Ethereum is implemented on smart contract, so that there are various implementations with each wallet software. Also, some blockchains might not support multi-signature, therefore some cryptoassets could not adopt multi-signature.

Also, there is another similar scheme "Secret Sharing Scheme" which is applicable to privilege separation. This is a management technology in a distributed environment which has divided secret respectively, and one of the countermeasures for leakage and/or lost of signature key. However, this scheme is rather a technology for single stakeholder with multi-location operation than multi-stakeholders, because it REQUIREs a validation scheme separately for the transaction to each stakeholder and management of the divided secret is rather depend to implementation than the signature key.

7.3.6.4. Backup for Signature Key

Backup is the most fundamental and effective measure against lost of signature key. On the other hand, there are risks of leakage and loss of the backup device.

These risks depend on the kind backup device, thus security controls on such devices MUST be considered independently. Followings describe typical backup devices and leakage/theft risks associated with them.

- Cloning to the tamper-resistant cryptographic key management device If a signature key is managed by a tamper-resistant key management device (device X) and X has cloning function, cloning the key to another device Y is the most secure way to back up the key, where the cloning function is the technique to copy the key with keeping confidentiality to other devices than X and Y [_11]. The implementation of the function is recommended to be evaluated/certified by certification programs like CMVP or FIPS 140. Note that, the cryptographic algorithms supported by such tamper-resistant key management devices are limited and all crypto assets systems can utilize it, but it is one of the most secure ways of backup.

- Backup to storage for digital data Here, it is assumed to backup keys to storage like USB memory and DVD. There are two types of operations; one is backup data is stored in movable devices in an offline manner, the other is backup data is stored in an online

accessible manner. If the device is movable, the possibility of steal and lost increases, thus the device MUST be kept in a cabinet or a vault with the key, and the access control to such cabinet/vault MUST be restricted. Of the backup storage is online, risks of leakage and theft MUST be assumed as same as the key management function implementation inside the cryptoassets custodian. In general, the same security control is recommended to such backup storage. If there is some additional operation, for example, the backup device is inactivated except for the time of restore, the security control may be modified with considering the operating environment. When it is not avoided the raw key data is outside of the key management function implementation, the custodian MUST deal with the problem of remained magnetics.

- Backup to paper There is a way to backup keys in an offline manner, to print them to papers as a QR code or other machine readable ways. It is movable than storage for digital data and easy to identify. There remains some risk of leakage and theft by taking a photo by smartphone and so on.

- Redundant with Sharing secret scheme Dividing of signature key to multiple parts, then managing them by multiple isolated systems is an effective measure to protect the keys against leakage and theft. This document does not recommend a specific technique but RECOMMENDs to implement this control based on a certain level of security evaluation like a secret sharing scheme. In that case, secure coding and mounting penetration test are REQUIRED to eliminate the implementation vulnerabilities. This method is also effective for backup devices.

7.3.6.5. Procurement of hardware wallet

When introducing a wallet, it is RECOMMEND to use a product whose technical security is guaranteed like HSM which is originally used for existing PKI service etc. However, some products may not be applicable currently because they often do not support a kind of cryptographic algorithm used by crypto assets. Therefore, if introducing a wallet, it is RECOMMEND to operate in mind the following points with accepting the technical insufficiency:

- MUST not use hardware obtained through the untrusted procurement route.

- MUST apply the latest firmware and patches provided by the manufacturer.

- Initialization and key generation MUST do themselves, SHOULD NOT use default settings without careful considerations.

- MUST consider trustworthy of software instructing a sign to hardware wallet, especially whether it supports multi-signature or signing at the offline environment.

Additionally, when custodian uses only hardware wallets in the marketplace, they MUST manage it according to section Section 7.3.4.

On the other hand, hardware wallets MUST be subject to the third-party or independent certification scheme for security. If introducing a software wallet from outside, it MUST consider the potentiality of containing malicious code, vulnerability, and bugs.

7.3.7. Physical and environmental security

Cryptoassets custodians system MUST follow section "11. Physical and environmental security" on [ISO.27002:2013].

Cryptoassets custodians system MUST consider strict physical security protections for the following elements.

- Media containing a signature key. (Signature key management shown in Figure 1)
- Media containing a signature key for cold wallet environment. (Signature key management for offline management shown in Figure 4.)
- Media containing a backup data of signature key

If the signature key mentioned above is stored in the deactivated state, and also key encryption key to activate the signature key is controlled separately, the media containing the key encryption key MUST be strictly managed.

The security control to these signature key MUST be separated from the security control of the crypto assets custodian system. In addition to this control, access to facilities and environments which store media containing a signature key or information required to operate the signature key MUST be restricted. (See: Section 7.3.6)

Furthermore, countermeasures to loss or theft for the operational device MUST be taken place if the administration or operation is executed from a remote place such as out of a facility.

7.3.8. Operations security

Crypto Assets Custodian systems MUST follow section "12. Operations security" on [ISO.27002:2013]. In addition to the standard, cryptoassets custodian systems SHALL comply with the security controls mentioned following sections.

7.3.8.1. Protections from malicious software (Related to ISO.27002:2013 12.2)

Detection and recovery measures of malware MUST be appropriately taken place according to configurations, the environment of cryptoassets custodian systems and confidentiality and importance of information handled in the systems.

In general, one of the prevention measures for malware is applying security patches to operating systems, middlewares of cryptoassets custodian systems. However, those patches MUST be applied upon sufficient confirmation based on the importance and urgency of a patch. Moreover, testing and deployment procedure of security patch MUST be considered beforehand just in case attacks against the vulnerability have already confirmed.

7.3.8.2. Backup (Related to ISO.27002:2013 12.3)

Upon making a backup of systems, strict security controls to important data which suffered severe damage by leakages such as the signature key or master seed MUST be applied same as data subject to backup (e.g., an appropriate selection for storage, and enforcement of strict access controls.) Security controls such as distributed storage mentioned in Section 7.3.6), proper privilege separation on backup and restore between operators and people making an authorization, and operation with multiple parties are also important.

7.3.8.3. Logging and monitoring (Related to ISO.27002:2013 12.4)

Crypto asset custodians systems MUST obtain/monitor/record logs properly (not limited to but include following logs).

- Logs on the environment where the cryptoassets custodian system Collecting and monitoring of event log outputted from the system components such as middleware, operating systems, and computers detects an abnormal state of environment where the system runs. Collected logs are used to investigate a cause in the case of the incident.

- Logs on the processing of components of crypto assets custodians system Collecting and monitoring of the processing logs from each component detects an abnormal state of crypto assets custodians system. Collecting proper logs are used as a proof of proper processing inside the crypto assets custodians system, and also used to investigate a cause in a case of the incident.

- Access log of signature key Information such as date, a source terminal, an operator(not a role but information to identify an operator) MUST be obtained and recorded in a case of operations such as activation and deactivation of the signature key, access to the activated signature key and backup/restore. Those records MUST be validated against the records such as operational procedures, operating hours, on a periodical inspection such as weekly inspection. Moreover, in a case where the signature key is managed online, operational log such as the creation of a transaction signature by operator MUST be recorded and validated as well.

- Operational Log of a wallet managed by custodians Logs on remittance MUST be monitored real-time against the attempt of outgoing coin transfer in a case where the signature key and backup are unexpectedly leaked. In a case where an unexpected remittance has occurred in one of the wallets, monitoring logs help timely detecting the incidents, suspending all signing operations, rechecking on other existing wallets, and migrating to other wallets using a different signature key.

- Access log of administration remote terminal If a remote access to cryptoassets custodian system is permitted, audit information such as date, source IP address, terminal information(e.g. terminal ID, latest result of security evaluation if it's possible) and destination IP address (or hostname) MUST be obtained and recorded for auditing which checks the accesses are from/in authorized range.

- Traffic log between the inside and the outside (e.g., the Internet) As mentioned in Section 7.3.9.1, Inbound traffic to cryotoassets custodian systems such as traffic from the Internet MUST be restricted to a permitted external network or permitted protocol. Inbound traffic from disallowed network and traffic using disallowed protocol are denied at the firewall and other middleboxes. Logs from that equipment are effective to protect customers from malicious access in terms of not only cryptoassets custodian system but also the information security. Usually, outbound traffic from protected assets such as cryptoassets custodian systems to the Internet and other systems is not a subject to logging. However, those logs are useful in cases such

as investigations on incidents (e.g., malicious usage of the signature key, theft of signature key) and detection of the incident, so entire traffic or network flow are RECOMMENDED to be acquired according to protocols/destinations.

- Customers access log Customers access log MUST be obtained since those logs are used to detect malicious login or request. Also, those logs are used as evidence in a case of incidents. In a case of malicious login, custodians MUST notify its customer.

  * Provide information about the malicious activity to customers Providing a feature to allow a customer to confirm login history, source IP address, region, and terminal information, and login notification by a push-notification or an e-mail are effective to detect malicious access after the incident. Feature protecting an account and alerting to a user in cases when detecting login from unknown source address or terminal, or detecting consecutive login to multiple accounts from the same source IP address, are effective to protect a user from malicious access.

- Images/videos recorded by a surveillance camera and entry/exit records Storing images/videos recorded by the surveillance camera and entry/exit records for proper period enables validating if physical safety control measures work properly after the incident.

Detecting a malicious process execution (e.g., malware), malicious access, an abnormal state of cryptoassets custodians system by monitoring logs mentioned above comprehensively is important. Moreover, storing this evidence is important to prevent internal fraud and exonerate person involved from the charge. Security Operation Center (SOC) may help to monitor the system. Outsourcing to trusted operators about detection and notification of threats in the operation of SOC may be helpful.

7.3.9. Communications security

Cryptoassets custodians system MUST follow section "13. Communications security" on [ISO.27002:2013].

Since assets are managed in a state accessible from the Internet on cryptoassets custodians system, preventive measures, detection measures, countermeasures and recovery measures as measures to prevent information leakage, MUST be considered according to the risk.

7.3.9.1. Network security management (Related to ISO.27002:2013 clause 13.1.1)

As same as security control measures to general systems, measures such as a definition of a boundary to the external network, restriction of connection to a network system(e.g., firewall), stop unnecessary services or close unnecessary ports, obtaining and monitoring logs and malicious access detection MUST be considered and performed.

For logs, logs of internal systems MUST be monitored to detect internal malicious access, as well as monitoring of boundary to the external network. (See: Section 7.3.8.3)

Secure communication with proper mutual authentication such as TLS(Transport Layer Security) MUST be used to protect from attacks to communication between modules such as eavesdropping and manipulation in a case where modules of cryptoassets custodians systems are remotely located.

7.3.9.2. Network segmentation (Related to ISO.27002:2013 13.1.3)

It is important to limit a connection between cryptoassets custodians systems and other systems/the Internet as minimum as possible to reduce the risk of exposing against attacks through a network. Measures as follow such as network segmentation and limitation to connection MUST be considered.

- Network isolation between custodians systems and other information systems
  - Objectives: Preventing a connection to custodians systems through information systems used in daily operations, which has been compromised due to malware infections caused by external attacks such as targeted attack.
  - Countermeasures: Isolate a network between information systems used in daily operations and custodians system by segmentation of network or limiting access.
- Network isolation at the boundary to the Internet
  - Objective: Preventing access to critical information such as a signature key from attack through the Internet by minimizing and isolating modules which connect to the Internet.
  - Countermeasures: Features which connects external services on the Internet to achieve the functionality of custodians system,

transmit transactions or obtain blockchain data MUST be packaged as a module as minimum as possible or be isolated from other systems such as locating on DMZ. Moreover, if modules are connecting to external services, access controls to those services MUST be adequately performed.

- Limitation on a terminal used in custodians system administration
  - Objective: Preventing a malicious operation due to a hijacking of terminal used in custodians system administration.
  - Countermeasures: Limiting a terminal which can connect to custodians system, such as a terminal to manage a custodians system administration function and a terminal running an administrative tool to order operation to custodians system.

7.3.9.3. System acquisition, development and maintenance

Cryptoassets custodians system MUST follow section "14. System acquisition, development, and maintenance" on [ISO.27002:2013].

Cryptoassets handled by cryptoassets custodians ranges from high liquidity cryptoassets dealt with by multiple custodians to emerging cryptoassets. It is important to reduce a risk regarding system acquisition, development and maintenance in addition to [ISO.27002:2013] as characteristics of blockchain network used by those cryptoassets varies. For example, the following countermeasures are effective.

- Software development method Secure software development method such as secure coding and code review MUST be used in the software development of the custodian system. Code review not only with the development team but also with an operational team is effective to detect a vulnerability from the viewpoint of operation.
- Penetration test Conducting a penetration test helps to detect a known vulnerability at systems and results in obviating the attacking risk by the attacker in advance.
- Integration test with blockchain network Test MUST be performed not only with the test network of blockchain but also with the production network of the blockchain. Risk assessment MUST be taken with an understanding of the limitation of test on the production network such as high-load test.
- Privilege separation on the operation Privilege separation such as limiting code reviewed software deployment to the production

environment to the system operating team is effective to prevent tampering attacks from internal.

- Prohibiting using default (factory-configured) values Any factory-configured authentication information such as password MUST NOT be used regardless of hardware/software, development environment or production environment.

7.3.10. Supplier relationships

Cryptoassets custodians system MUST follow section "15. Supplier relationships" on [ISO.27002:2013].

Outsourcing wallet-related services may be a reasonable choice in a case technical security of those services has been secured.

Administrative measures according to [ISO.27002:2013] MUST be taken in terms of outsourcing contractors or security controls of cloud service providers in cases where signature key in multi-signature is delegated to contractors or custodians system is implemented on cloud services.

7.3.11. Information security incident management

Cryptoassets custodians system MUST follow section "16. Information security incident management" on [ISO.27002:2013].

Since cyber attacks got complex, cyber security incidents unprecedented in the past could occur, especially in cryptoassets custodians. In addition to security control measures as a preparation to expected threat in advance, Emergency response framework MUST be prepared in a case of incidents caused by an unknown threat. For example, the establishment of internal CSIRT(Computer Security Incident Response Team) and building a relationship with external organizations.

7.3.12. Information security aspect of business continuity management

Cryptoassets custodians system MUST follow section "17. Information security aspect of business continuity management" on [ISO.27002:2013].

Requirements, Processes, Procedures and control measures to secure information security for the cryptoassets custodian in a case of the severe situation(such as disaster or crisis) MUST be established, documented, performed and maintained. In this case, administrative measures in a case where countermeasures have performed or in a period of a severe situation MUST be verified periodically.

Moreover, operators MUST consider to shut down the system situationally.

- In a case where facilities (including facilities used as an office) are unavailable
  - Power outage
  - Damages of building
  - An act of nature (e.g., earthquakes, fires (including sprayed water for neighborhoods fire), water outage, flood)
  - Other reasons (e.g., facilities are unavailable, or access to the facilities are prohibited by law/regulations/authorities.)
- In a case where it's difficult to continue the system
  - In a case of becoming difficult to continue running an emergency electric generator.
  - Long suspension of public transportation services, a pandemic of disease, lack of human resources by an act of nature.
  - Failure of a communication network
  - Failure of equipment
  - Failure of the system (regardless of reasons such as failure of a program or cyber attacks)
  - Loss of paper wallet or hardware wallet.
  - Suspension of outsourcing contractor's business
  - Leakage or loss of signature key
- In the case of becoming difficult to continue business
  - Business-suspension order by law/regulations.

7.3.12.1. Maintaining availability of the system

Cryptoassets custodians system MUST be designed and implemented to have enough scalability and redundancy for users with consideration of a number of users, peak date/time of transactions, system response time, maintenance period/frequency and securing a human resource for operation. Moreover, consideration for increasing the capacity of

the system MUST be performed in advance with enough threshold (e.g., number of transactions or memory usage during a peak period).

7.3.13. Compliance

Cryptoassets custodians MUST respect the guidelines or laws of the region or country. (See Appendix 3 for a country of Japan)

7.4. Other cryptoassets custodians system specific issues

7.4.1. Advance notice to user for maintenance

Cryptoassets custodians are RECOMMENDED to publish a notice of maintenance schedule in advance in a case where periodical schedule especially service suspension is planned in a night. Also, Cryptoassets custodians are RECOMMENDED to provide information regarding the failure of the system at other FQDN/IP addresses to avert high volume traffic to the web server in addition to usual way of notice such as by e-mail or on the website, in a case of emergency maintenance.

Moreover, cryptoassets custodians are RECOMMENDED to put forth an effort to minimize an affected area from a viewpoint of user protection in a case of service suspension caused by immediate issues such as attacks from external.

8. Future work

Discussion of distributed exchange (DEX) is currently out-of-the-scope of this document.

9. Security Considerations

Security Considerations are included in the main section of this document.

10. IANA Considerations

None.

11. References

11.1. Normative References

[ISO.27001:2013]
International Organization for Standardization, "Information technology -- Security techniques -- Information security management systems -- Requirements", ISO/IEC 27001:2013, October 2013, <https://www.iso.org/standard/54534.html>.

[ISO.27002:2013]
International Organization for Standardization, "Information technology -- Security techniques -- Code of practice for information security controls", ISO/IEC 27002:2013, October 2013, <https://www.iso.org/standard/54533.html>.

[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>.

[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/info/rfc8174>.

11.2. Informative References

[BIP-0010]
Reiner, A., "Multi-Sig Transaction Distribution", BIP 10, October 2011, <https://github.com/bitcoin/bips/blob/master/bip-0010.mediawiki>.

[BIP-0011]
Andresen, G., "M-of-N Standard Transactions", BIP 10, October 2011, <https://github.com/bitcoin/bips/blob/master/bip-0011.mediawiki>.

[CVE-2018-10299]
MITRE Corporation, "CVE-2018-10299", CVE 2018-10299, April 2018, <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-10299>.

[I-D.nakajima-crypto-asset-terminology]
Nakajima, H., Kusunoki, M., Hida, K., Suga, Y., and T. Hayashi, "Terminology for Cryptoassets", draft-nakajima-crypto-asset-terminology-03 (work in progress), December 2019.

[LISK-ISSUE:2088]
MaciejBaj, ., "Check INT_32 range for transaction timestamps", June 2018, <https://github.com/LiskHQ/lisk/issues/2088>.

11.3. URIs

[1] https://vcgtf.github.io

Acknowledgements

Thanks to Masanori Kusunoki, Yasushi Matsumoto, Natsuhiko Sakimura, Yuji Suga, Tatsuya Hayashi, Keiichi Hida, Kenichi Sugawara, Naochika Hanamura, and other members of the Security Working Group of CryptoAssets Governance Task Force [1].

Editorial Comments

[_11] For example, cloning via PC does not meet this requirement when the signature key is read into memory on the PC in the cloning.

Authors' Addresses

Masashi Sato
SECOM Co., Ltd. Intelligent System Laboratory
Shimorenjaku 8-10-16
SECOM SC Center
Tokyo, Mitaka 181-8528
JAPAN

Email: satomasa756@gmail.com

Masaki Shimaoka
SECOM Co., Ltd. Intelligent System Laboratory
Shimorenjaku 8-10-16
SECOM SC Center
Tokyo, Mitaka 181-8528
JAPAN

Email: m-shimaoka@secom.co.jp

Hirotaka Nakajima (editor)
Mercari, Inc.
Roppongi 6-10-1
Roppongi Hills Mori Tower 18F
Tokyo, Minato 106-6143
JAPAN

Email: nunnun@mercari.com