Workgroup: drip Working Group
Internet-Draft: draft-ietf-drip-registries-05
Published: 11 July 2022
Intended Status: Standards Track
Expires: 12 January 2023
Authors: A. Wiethuechter (Editor) S. Card
AX Enterprize, LLC AX Enterprize, LLC
R. Moskowitz J. Reid
HTT Consulting RTFM llp

DRIP Entity Tag (DET) Registration & Lookup

Abstract

This document details the required mechanisms for the registration and discovery of DRIP Entity Tags (DETs). The registration process relies upon the Extensible Provisioning Protocol (EPP). The discovery process leverages DNS, DNSSEC, and related technology. The lookup process relies upon the Registration Data Access Protocol (RDAP). The DETs can be registered with as their "raw public keys" or in X.509 certificates.

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 12 January 2023.

Table of Contents

1. Introduction

Registries are fundamental to Remote ID (RID). Only very limited operational information can be Broadcast, but extended information is sometimes needed. The most essential element of information sent

is the UAS ID itself, the unique key for lookup of extended information in registries.

While it is expected that registry functions will be integrated with USS, who will provide them is not yet determined in most, and is expected to vary between, jurisdictions. However this evolves, the essential registry functions, starting with management of identifiers, are expected to remain the same, so are specified herein.

While most data to be sent via Broadcast or Network RID is public, much of the extended information in registries will be private. Thus AAA for registries is essential, not just to ensure that access is granted only to strongly authenticated, duly authorized parties, but also to support subsequent attribution of any leaks, audit of who accessed information when and for what purpose, etc. As specific AAA requirements will vary by jurisdictional regulation, provider philosophy, customer demand, etc., they are left to specification in policies, which should be human readable to facilitate analysis and discussion, and machine readable to enable automated enforcement, using a language amenable to both, e.g., XACML.

The intent of the negative and positive access control requirements on registries is to ensure that no member of the public would be hindered from accessing public information, while only duly authorized parties would be enabled to access private information. Mitigation of Denial of Service attacks and refusal to allow database mass scraping would be based on those behaviors, not on identity or role of the party submitting the query per se, but querant identity information might be gathered (by security systems protecting DRIP implementations) on such misbehavior.

Registration under DRIP is vital to manage the inevitable collisions in the hash portion of the DET. Forgery of the DET is still possible, but including it as a part of a public registration mitigates this risk. This document creates the DRIP DET registration and discovery ecosystem. This includes all components in the ecosystem (e.g., RAA, HDA, UA, GCS, USS). The registration process will use the Extensible Provisioning Protocol (EPP) and other protocols. The discovery process will leverage DNS and DNSSEC and related technology. The DETs can be registered with as their "raw public keys" or in X.509 certificates.

1.1. Abstract Process & Reasoning

In DRIP each entity (registry, operator and aircraft) is expected to generate a full DRIP Entity ID [drip-rid] on the local device their key is expected to be used. These are registered with a Public Information Registry within the hierarchy along with whatever data

is required by the cognizant CAA and the registry. Any PII is stored in a Private Information Registry protected through industry practice AAA or better. In response, the entity will obtain an attestation or certificate from the registry proving such registration.

Manufacturers that wish to participate in DRIP should not only support DRIP as a Session ID type for their aircraft but also generate a DET then encode it as a Serial Number. This would allow aircraft under CAA mandates to fly only ID Type 1 (Serial Number) could still use DRIP and most of its benefits. Even if DRIP is not supported for Serial Numbers by a Manufacturer it is hoped that they would still run a registry to store their Serial Numbers and allow look ups for generic model information. This look up could be especially helpful in UTM for Situational Awareness when an aircraft flying with a Serial Number is detected and allow for an aircraft profile to be displayed.

Operators are registered with a number of registries or their regional RAA. This acts as a verification check when a user performs other registration operations; such as provisioning an aircraft with a new Session ID. It is an open question if an Operator registers to their CAA (the RAA) or multiple USS's (HDA's). PII of the Operator would vary based on the CAA they are under and the registry.

Finally aircraft that support using a DET would provision per flight to a USS, proposing a DET to the registry to generate a binding between the aircraft (Session ID, Serial Number and Operational Intent), operator and registry. Aircraft then follow [drip-auth] to meet various [drip-requirements] during flight.

2. Terminology

2.1. Required Terminology

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.

2.2. Definitions

See [drip-requirements] for common DRIP terms.

HDA: Hierarchial HIT Domain Authority. The 16 bit field identifying the HIT Domain Authority under a RAA.

HID: Hierarchy ID. The 32 bit field providing the HIT Hierarchy ID.

RAA:
Registered Assigning Authority. The 16 bit field identifying the Hierarchical HIT Assigning Authority.

3. Registries

3.1. Classes

Under DRIP there 3 classes of registries, with specific variants in each.

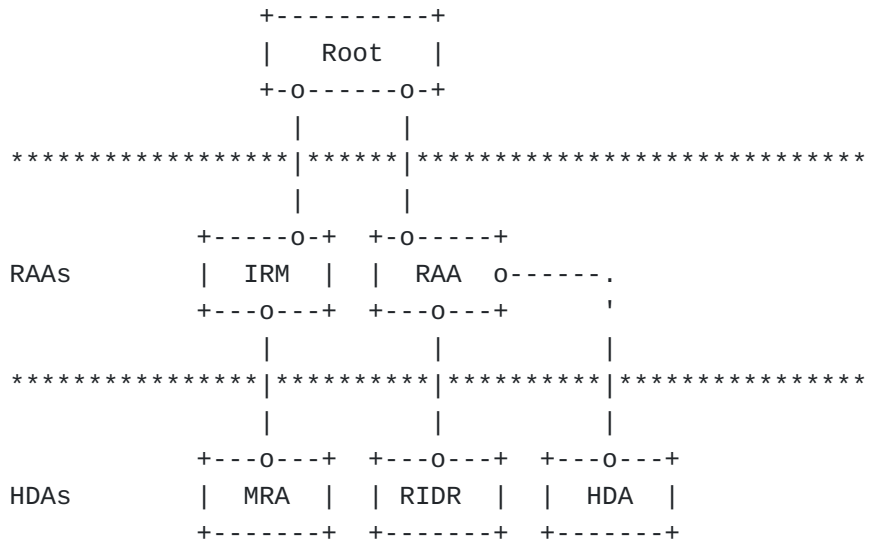


Figure 1: Registry Hierarchy

3.1.1. Root

This is a special registry holding the RAA value of 0 and HDA value of 0. It delegates out RAA values only to registries that wish to act as an RAA.

3.1.2. Registered Assigning Authorities

RAA's are the upper hierarchy in DRIP (denoted by a 14-bit field (16,384 RAAs) of a DET). An RAA is a business or organization that manages a registry of HDAs (Section 3.1.3). Most are contemplated to be Civil Aviation Authorities (CAA), such as the Federal Aviation Authority (FAA), that then delegate HDAs to manage their NAS. This is does not preclude other entities to operate an RAA if the Root server allows it.

The ICAO registration process will be available from ICAO. Once ICAO accepts an RAA, it will assign a number and create a zone delegation under the uas.icao.int. DNS zone for the RAA.

As DETs may be used in many different domains, RAA should be allocated in blocks with consideration on the likely size of a particular usage. Alternatively, different prefixes can be used to separate different domains of use of HHITs.

An RAA must provide a set of services to allocate HDAs to organizations. It must have a public policy on what is necessary to obtain an HDA. It must maintain a DNS zone minimally for discovering HID RVS servers. All RAA's use an HDA value of 0 and have their RAA value delegated to them by the Root.

3.1.2.1. ICAO Registry of Manufacturer's (IRM)

An RAA-level registry that hands out HDA values to participating Manufacturer's that hold an ICAO Manufacturer Code used in ANSI CTA2063-A Serial Numbers.

It is holds the RAA value of 1 and HDA value of 0.

3.1.3. Hierarchial HIT Domain Authorities

An HDA may be an USS, ISP, or any third party that takes on the business to register the actual UAS entities that need DETs. This includes, but is not limited to UA, GCS, and Operators. It should also provide needed UAS services including those required for HIP-enabled devices (e.g. RVS).

The HDA is a 14-bit field (16,384 HDAs per RAA) of a DET assigned by an RAA. An HDA should maintain a set of RVS servers for UAS clients that may use HIP. How this is done and scales to the potentially millions of customers are outside the scope of this document. This service should be discoverable through the DNS zone maintained by the HDA's RAA.

An RAA may assign a block of values to an individual organization. This is completely up to the individual RAA's published policy for delegation. Such policy is out of scope.

3.1.3.1. Manufacturer's Registry of Aircraft (MRA)

An HDA-level registry run by a manufacturer of UAS systems that participate in Remote ID. Stores UAS Serial Numbers under a specific ICAO Manufacturer Code (assigned to the manufacturer by ICAO).

A DET can be encoded into a Serial Number (see [drip-rid]) and this registry would hold a mapping from the Serial Number to the DET and its artifacts.

Hold RAA value of 1 and HDA values of 1+.

3.1.3.2. Remote ID Registries (RIDR)

An HDA-level registry that holds the binding between a UAS Session ID (for DRIP the DET) and the UA Serial Number. The Serial Number MUST have its access protected to allow only authorized parties to obtain. The Serial Number SHOULD be encrypted in a way only the authorized party can decrypt.

As part of the UTM system they also hold a binding between a UAS ID (Serial Number or Session ID) and an Operational Intent.

Hold RAA values of 2+ and HDA values of 1+.

3.2. Key Rollover & Federation

During key rollover the entity MUST inform all children and parents of the change - using best standard practices of a key rollover. At time of writing this is signing over the new key with the previous key in a secure fashion and it being validated by others before changing any links (in DRIPs case the NS RRs in the parent registry).

A DET has a natural ability for a single entity to hold different cryptographic identities under the same HID values. This is due to the lower 64-bits of the DET being a hash of the public key and the HID of the DET being generated. As such during key rollover, only the lower 64-bits would change and a check for a collision would be required.

This attribute of the DET to have different identities could also allow for a single registry to be "federated" across them if they share the same HID value. This method of deployment has not been thoroughly studied at this time.

4. DRIP Fully Qualified Domain Names

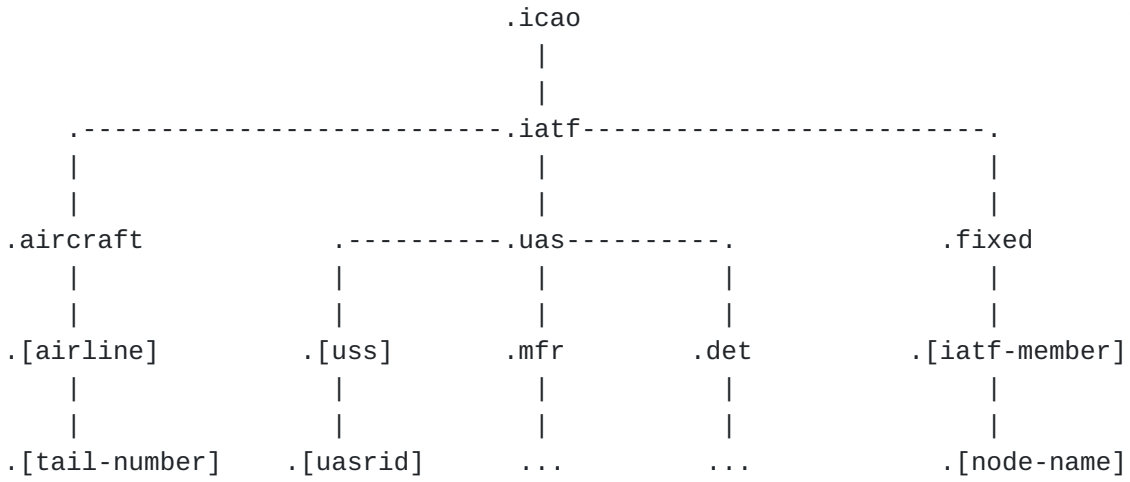
Under DRIP there are a number of FQDN forms used to allow lookups to take place.

The individual DETs may be potentially too numerous (e.g., 60 - 600M) and dynamic (e.g., new DETs every minute for some HDAs) to store in a signed, DNS zone. The HDA SHOULD provide DNS service for its zone and provide the DET detail response.

DNSSEC is strongly recommended (especially for RAA-level zones). Frequency of updates, size of the zone, and registry policy may impact its use.

4.1. ICAO DNS Structure

Under the TSVG ICAO panel, the GRAIN task group has discussed a DNS structure, with DRIP proposing to fit into it as follows:



Example DET FQDN: a3ad19520ad0a69e.
5.20.10.20010030.det.uas.iatf.icao

Example MFR FQDN: Z2T7B8RA85D19LX.8653.mfr.uas.iatf.icao

4.2. Serial Number

See Section 4.2 of [drip-rid] for how to encode DETs as Serial Numbers.

Serial Number: 8653FZ2T7B8RA85D19LX
ICAO Mfr Code: 8653
Length Code: F
ID: FZ2T7B8RA85D19LX
FQDN: Z2T7B8RA85D19LX.8653.mfr.uas.icao.int

4.3. DET

DETs SHOULD be discoverable in DNS via a hash.oga.hda.raa.prefix. structure under a ICAO managed DNS aviation tree. This document proposes a .det.uas.icao.int. branch in the current ICAO DNS domain. This is subject of discussions with ICAO. Thus if we assume a DET prefix of 2001:30::/28, the following example shows the resultant DNS entry:

```
DET: 2001:0030:00a0:0145:a3ad:1952:0ad0:a69e
ID: a3ad:1952:0ad0:a69e
OGA: 5
HDA: 0014 = 20
RAA: 000a = 10
Prefix: 20010030
FQDN: a3ad19520ad0a69e.5.20.10.20010030.det.uas.icao.int
```

When building a DET FQDN the following two things must be done:

1. The RAA and HDA values MUST be converted from hexadecimal to decimal form.

2. The FQDN must be built using the expanded form of the IPv6 address.

The prefix is included in the FQDN form to support other potential prefixes being used.

4.4. Reverse DET

The DET reverse lookup should be a standard IPv6 reverse address in ip6.arpa.

```
$ORIGIN  5.4.1.0.0.a.0.0.0.3.0.0.1.0.0.2.ip6.arpa.
e.9.6.a.0.d.a.0.2.5.9.1.d.a.3.a    IN   PTR
```

5. Supported DNS Records

DRIP requires a number of resource records, some specific to certain registries to function.

5.1. HIP RR

All registries will use HIP RR [RFC8005] as the primary public source of DET HIs. The DETs are encoded (Section 4.2) in an FQDN and are the lookup key for the RR. Registries have their own DET associated with them and their respective DNS server will hold a HIP RR that is pointed to by their DET FQDN.

MRA and RIDR servers will also have HIP RRs for their registered parties (aircraft and operators).

5.2. TLSA RR

This RR, [RFC6698], is mainly used to support DTLS deployments where the DET is used (e.g. Network RID and the wider UTM system). The HI is encoded using the SubjectPublicKeyInfo selector. DANE [RFC6698] is for servers, DANCE [dane-clients] is for clients.

The TLSA RR MAY be used in place of the HIP RR, where to primary need of the DET HI is for DTLS authentication. This DNS server side optimization is for where the overhead of both RR is onerous. Thus all clients that work with the HIP RR SHOULD be able to able to extract the HI from the TLSA RR.

5.3. CERT RR

Attestations can be placed into DNS in the CERT RRs [RFC4398]. An exception to this is the Attestation Certificate made during Session ID registration. This is as this particular certificate acts similar to a car registration and should be held safe by the operator.

Attestations will be stored in Certificate Type OID Private (value 254) with an OID of 1.3.6.1.4.1.6715.2.n, where n is the Attestation Type.

Editor Note: This OID is an initial allocation under the IANA Enterprise Number OID. It is expect that a general OID will be allocated at some point.

Certificate Type X.509 as per PKIX (value 1) MAY be used to store X.509 certificates as discussed in (Appendix B).

Editor Note: Have not gotten to the part in this draft where Attestation Type, above is defined and enumerated. Obviously this is needed.

5.4. NS RR

Along with their associated "glue" record (A/AAAA) supports the traversal in DNS across the tree.

1. <mfr.remoteid.aero> on Root points to specific DET FQDN of IRM

2. <icao_mfr_code>.mfr.remoteid.aero on IRM points to specific DET FQDN of MRA

3. <raa_value>.det.remoteid.aero on Root pointing to DET FQDN of matching RAA

4. <hda_value>.<raa_value>.det.remoteid.aero on RAA pointing to DET FQDN of matching HDA

Editor Note: .areo works as a placeholder for now, but we are working with ICAO for this to at least be under uas.icao.int. and their TLD that they are working to get in the next ICANN TLD round. GRAIN already lists and uses uas.icao.int.. Perhaps we use det.uas.icao.int. for all DRIP DNS structures.

5.5. AAAA RR

DRIP requires the use of IPv6.

5.6. SVR RR

The SVR RR for DRIP always is populated at the "local" registry level. That is an HDA's DNS would hold the SVR RR that points to that HDAs private registry for all data it manages. This data includes data being stored on its children.

The best example of this is RIDR would have a SVR RR that points to a database that contains any extra information of a Session ID it has registered. Another example is the MRA has a SVR RR pointing to where the metadata of a UA registered in the MRA can be located.

In all cases the server being pointed to MUST be protected using AAA, specifically using RDAP.

6. Registry Operations

As a general rule the following processing performed for any registration operation:

1. Verify input Attestations from registering party
2. Check for collision of DET and HI
3. Populate DNS with required/optional records
4. Populate Database with PII and other info
5. Generate and return required/optional Attestations/Certificates

6.1. Registering a Registry

DRIP defines two levels of hierarchy maintained by the Registration Assigning Authority (RAA) and HHIT Domain Authority (HDA). The authors anticipate that an RAA is owned and operated by a regional CAA (or a delegated party by an CAA in a specific airspace region) with HDAs being contracted out. As such a chain of trust for registries is required to ensure trustworthiness is not compromised. More information on the registries can be found in [hhit-registries].

Both the parent and child generate their own keypairs and self-signed attestations if not generated previously. The child sends to the parent its self-signed attestation to be added into the parent DNS.

The parent confirms the attestation received is valid and that no DET collisions occur before adding a NS RR (and CERT RRs) to its DNS for the new child. An attestation, parent on child, is sent as a confirmation that provisioning was successful.

The child is now a valid member of the registry tree and uses its keypair and Self-Attestation with all provisioning requests towards it. The HIP RR for the child is populated into the local DNS along with any CERT RRs.

6.1.1. Registering an RAA

Specifically handled by the Root (Section 3.1.1).

Inputs (Optional)	DNS Entries (Optional)	Outputs (Optional)
IP Address of RAA	Root: <raa_value>.det.remoteid.aero NS <raa_det_fqdn>	Attestation: Root, RAA
RAA Self-Attestation	Root: <raa_det_fqdn> AAAA <raa_ip>	(Concise Attestation: Root, RAA)
	RAA: <raa_det_fqdn> HIP <hip_rr_data>	(Broadcast Attestation: Root, RAA)
	RAA: <raa_det_fqdn> CERT <raa_self_attestation>	
	(Root: <raa_det_fqdn> CERT <attestation_root_raa>)	
	(Root: <raa_det_fqdn> CERT <concise_attestation_root_raa>)	
	(Root: <raa_det_fqdn> CERT <broadcast_attestation_root_raa>)	
	(RAA: <raa_det_fqdn> CERT <attestation_root_raa>)	
	(RAA: <raa_det_fqdn> CERT <concise_attestation_root_raa>)	
	(RAA: <raa_det_fqdn> CERT <broadcast_attestation_root_raa>)	

Table 1

6.1.2. Registering an IRM

Specifically handled by the Root (Section 3.1.1).

Inputs (Optional)	DNS Entries (Optional)	Outputs (Optional)
IP Address of IRM	Root: mfr.remoteid.aero NS <irm_det_fqdn>	Attestation: Root, IRM

Inputs (Optional)	DNS Entries (Optional)	Outputs (Optional)
IRM Self-Attestation	Root: 1.det.remoteid.aero NS <irm_det_fqdn>	(Concise Attestation: Root, IRM)
	Root: <irm_det_fqdn> AAAA <irm_ip>	(Broadcast Attestation: Root, IRM)
	IRM: <irm_det_fqdn> HIP <hip_rr_data>	
	IRM: <irm_det_fqdn> CERT <irm_self_attestation>	
	(Root: <irm_det_fqdn> CERT <attestation_root_irm>)	
	(Root: <irm_det_fqdn> CERT <concise_attestation_root_irm>)	
	(Root: <irm_det_fqdn> CERT <broadcast_attestation_root_irm>)	
	(IRM: <irm_det_fqdn> CERT <attestation_root_irm>)	
	(IRM: <irm_det_fqdn> CERT <concise_attestation_root_irm>)	
	(IRM: <irm_det_fqdn> CERT <broadcast_attestation_root_irm>)	

Table 2

6.1.3. Registering an HDA

Specifically handled by an RAA (Section 3.1.2).

Inputs (Optional)	DNS Entries (Optional)	Outputs (Optional)
IP Address of HDA	RAA: <hda_value>.<raa_value>.det.remoteid.aero NS <hda_det_fqdn>	Attestation: RAA, HDA
HDA Self-Attestation	RAA: <hda_det_fqdn> AAAA <hda_ip>	(Concise Attestation: RAA, HDA)
	RAA: <hda_det_fqdn> HIP <hip_rr_data>	(Broadcast Attestation: RAA, HDA)
	HDA: <hda_det_fqdn> CERT <hda_self_attestation>	
	(RAA: <hda_det_fqdn> CERT <attestation_raa_hda>)	
	(RAA: <hda_det_fqdn> CERT <concise_attestation_raa_hda>)	
	(RAA: <hda_det_fqdn> CERT <broadcast_attestation_raa_hda>)	

Inputs (Optional)	DNS Entries (Optional)	Outputs (Optional)
	(HDA: <hda_det_fqdn> CERT <attestation_raa_hda>)	
	(HDA: <hda_det_fqdn> CERT <concise_attestation_raa_hda>)	
	(HDA: <hda_det_fqdn> CERT <broadcast_attestation_raa_hda>)	

Table 3

6.1.4. Registering an MRA

Specifically handled by the IRM (Section 3.1.2.1).

Inputs (Optional)	DNS Entries (Optional)	Outputs (Optional)
IP Address of MRA	IRM: <icao_mfr_code>.mfr.remoteid.aero NS <mra_det_fqdn>	Attestation: IRM, MRA
MRA Self-Attestation	IRM: <hda_value>.1.det.remoteid.aero NS <mra_det_fqdn>	(Concise Attestation: IRM, MRA)
ICAO Manufacturer Code	IRM: <mra_det_fqdn> AAAA <mra_ip>	(Broadcast Attestation: IRM, MRA)
	MRA: <mra_det_fqdn> HIP <hip_rr_data>	
	MRA: <mra_det_fqdn> CERT <mra_self_attestation>	
	(IRM: <mra_det_fqdn> CERT <attestation_irm_mra>)	
	(IRM: <mra_det_fqdn> CERT <concise_attestation_irm_mra>)	
	(IRM: <mra_det_fqdn> CERT <broadcast_attestation_irm_mra>)	
	(MRA: <mra_det_fqdn> CERT <attestation_irm_mra>)	
	(MRA: <mra_det_fqdn> CERT <concise_attestation_irm_mra>)	
	(MRA: <mra_det_fqdn> CERT <broadcast_attestation_irm_mra>)	

Table 4

6.2. Registering a Serial Number

Specifically handled by an MRA (Section 3.1.3.1).

Inputs (Optional)	DNS Entries (Optional)	Outputs (Optional)
Serial Number	(<sn_det_fqdn> HIP <hip_rr_data>)	

Inputs (Optional)	DNS Entries (Optional)	Outputs (Optional)
		(Attestation: MRA, UA)
(UA Self-Attestation)	(<sn_det_fqdn> CERT <sn_self_attestation>)	(Broadcast Attestation: MRA, UA
UA Metadata	(<sn_det_fqdn> CERT <attestation_mra_sn>)	(Concise Attestation: MRA, UA)
	(<sn_det_fqdn> CERT <concise_attestation_mra_sn>)	
	(<sn_det_fqdn> CERT <broadcast_attestation_mra_sn>)	

Table 5

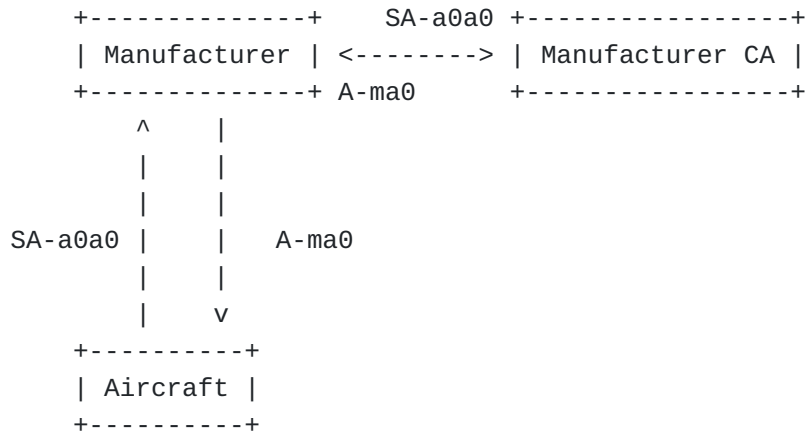


Figure 2: Manufacturer Provision

During the initial configuration and production at the factory the Aircraft MUST be configured to have a serial number. ASTM defines this to be an ANSI/CTA-2063A Serial Number for UAS. Under DRIP a DET can be encoded as such to be able to convert back and forth between them. This is covered in [drip-rid].

Under DRIP the Manufacturer SHOULD be using DETs and have their own keypair and Self-Attestation: Manufacturer (SA-m). (Ed. Note: some words on aircraft keypair and certs here?).

Self-Attestation: Aircraft 0 on Aircraft 0 (SA-a0a0) is extracted by the manufacturer and sent to their Certificate Authority (CA) to be verified and added. A resulting attestation (Attestation: Manufacturer on Aircraft 0 [A-ma0]) SHOULD be a DRIP Attestation - however this could be a X.509 certificate binding the serial number to the manufacturer.

6.3. Registering an Operator

Specifically handled by a RIDR (Section 3.1.3.2).

Inputs (Optional)	DNS Entries (Optional)	Outputs (Optional)
Operator Self-Attestation	<op_det_fqdn> HIP <hip_rr_data>	Attestation: RIDR, Operator
Operator PII	(<op_det_fqdn> CERT <op_self_attestation>)	Broadcast Attestation: RIDR, Operator
	(<op_det_fqdn> CERT <attestation_ridr_op>)	(Concise Attestation: RIDR, Operator)
	(<op_det_fqdn> CERT <concise_attestation_ridr_op>)	
	(<op_det_fqdn> CERT <broadcast_attestation_ridr_op>)	

Table 6

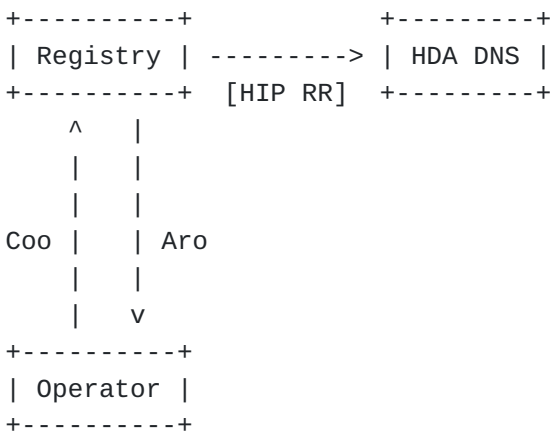


Figure 3: Operator Provision

The Operator generates a keypair and DET as specified in DRIP UAS RID. A self-signed attestation (Self-Attestation: Operator on Operator [SA-oo]) is generated and sent to the desired registry (HDA). Other relevant information and possibly personally identifiable information needed may also be required to be sent to the registry (all over a secure channel - the method of which is out of scope for this document).

The registry cross checks any personally identifiable information as required. Certificate: Operator on Operator is verified (both using the expiration timestamp and signature). The DET is searched in the Registries database to confirm that no collision occurs. A new attestation is generated (Attestation: Registry on Operator) and

sent securely back to the Operator. Optionally the DET/HI pairing can be added to the Registries DNS in to form of a HIP Resource Record (RR). Other RRs, such as CERT and TXT, may also be used to hold public information.

With the receipt of Attestation: Registry on Operator (A-ro) the provisioning of an Operator is complete.

6.4. Registering a Session ID

Specifically handled by a RIDR (Section 3.1.3.2).

Inputs (Optional)	DNS Entries (Optional)	Outputs (Optional)
Attestation: RIDR, Operator	<session_det_fqdn> HIP <hip_rr_data>	Attestation: RIDR, Operator
Attestation: Operator, UA	<session_det_fqdn> CERT <session_self_attestation>	Broadcast Attestation: RIDR, Operator
Serial Number	<session_det_fqdn> CERT <broadcast_attestation_ridr_session>	Attestation Certificate: RIDR, Operator, UA
(Concise Attestation: Operator, UA)	(<session_det_fqdn> CERT <attestation_ridr_session>)	(Concise Attestation: RIDR, Operator)
(Mutual Attestation: Operator, UA)	(<session_det_fqdn> CERT <concise_attestation_ridr_session>)	(Mutual Certificate: RIDR, Operator, UA)
(Link Attestation: Operator, UA)		(Concise Certificate: RIDR, Operator, UA)
(Operational Intent)		(Link Certificate: RIDR, Operator, UA)
		(Broadcast Attestation: RAA, RIDR)
		(Broadcast Attestation: Root, RAA)

Table 7

6.4.1. Standard Provisioning

Under standard provisioning the Aircraft has its own connectivity to the registry, the method which is out of scope for this document.

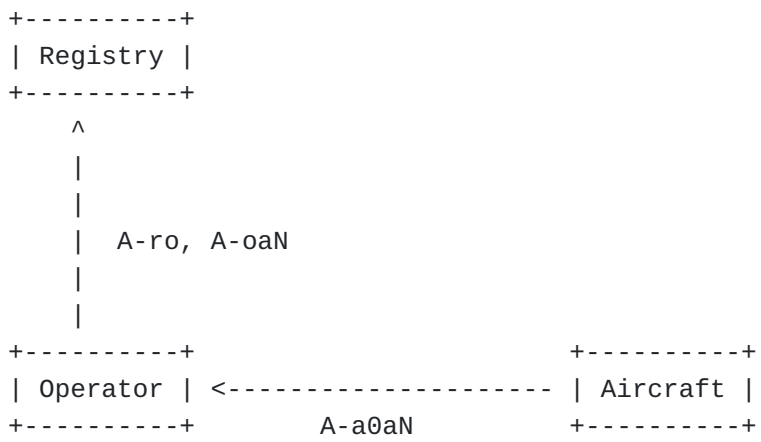


Figure 4: Standard Provision: Step 1

Through mechanisms not specified in this document the Operator should have methods to instruct the Aircraft onboard systems to generate a keypair and certificate. This certificate is chained to the factory provisioned certificate (Self-Attestation: Aircraft 0 on Aircraft 0 [SA-a0a0]). This new attestation (Attestation: Aircraft 0 on Aircraft N [A-a0aN]) is securely extracted by the Operator.

With A-a0aN the sub-attestation (Self-Attestation: Aircraft N on Aircraft N [SA-aNaN]) is used by the Operator to generate Attestation: Operator on Aircraft N (A-oaN). This along with Attestation: Registry on Operator (A-ro) is sent to the registry.

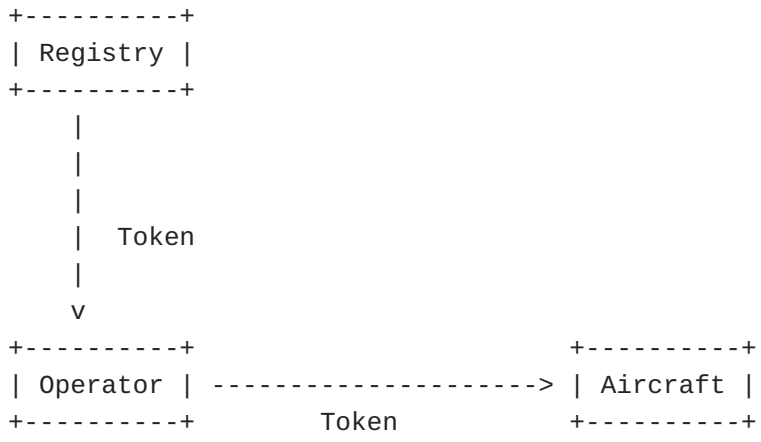


Figure 5: Standard Provision: Step 2

On the registry, A-ro is verified and used as confirmation that the Operator is already registered. A-oaN also undergoes a validation check and used to generate a token to return to the Operator to continue provisioning.

Upon receipt of this token, the Operator injects it into the Aircraft and its used to form a secure connection to the registry. The Aircraft then sends Attestation: Manufacturer on Aircraft 0 (A-ma0) and Attestation: Aircraft 0 to Aircraft N (A-a0aN).

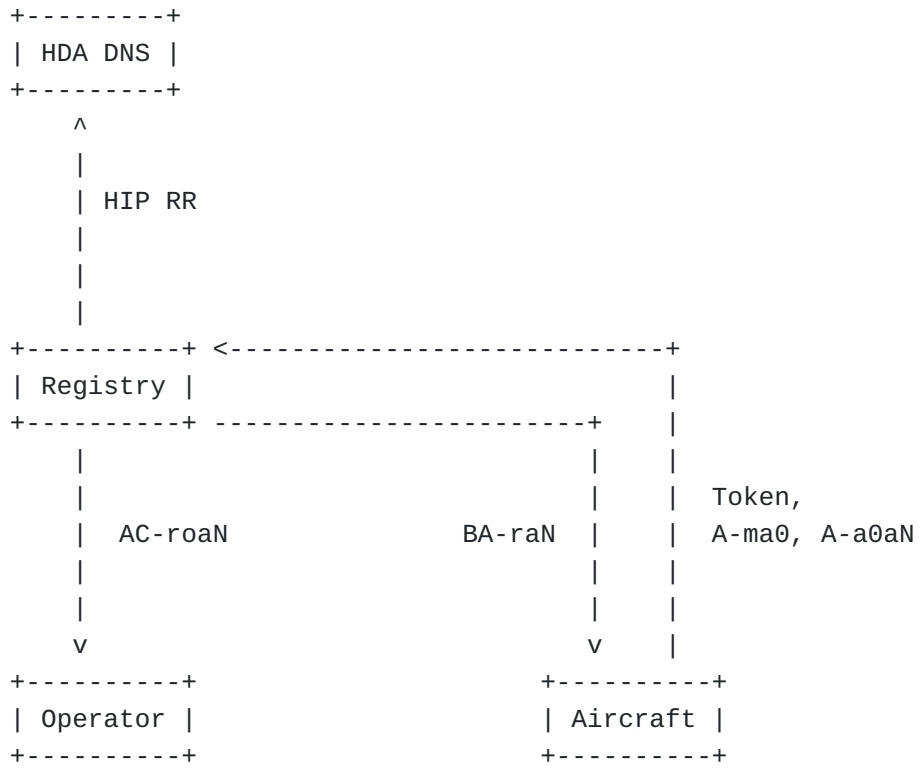


Figure 6: Standard Provision: Step 3

The registry uses Attestation: Manufacturer on Aircraft 0 (with an external database if supported) to confirm the validity of the Aircraft. Attestation: Aircraft 0 on Aircraft N is correlated with Attestation: Operator on Aircraft N and Attestation: Manufacturer on Aircraft 0 to see the chain of ownership. The new DET tied to Aircraft N is then checked for collisions in the HDA. With the information the registry generates two items: Attestation Certificate: Registry on Operator on Aircraft N (AC-roaN) and Broadcast Attestation: Registry on Aircraft N (BA-raN). A HIP RR (and other RR types as needed) are generated and inserted into the HDA.

AC-roaN is sent via a secure channel back to the Operator to be stored. BA-raN is sent to the Aircraft to be used in Broadcast RID as specified in [drip-auth].

6.4.2. Operator Assisted Provisioning

This provisioning scheme is for when the Aircraft is unable to connect to the registry itself or does not have the hardware required to generate keypairs and attestations.

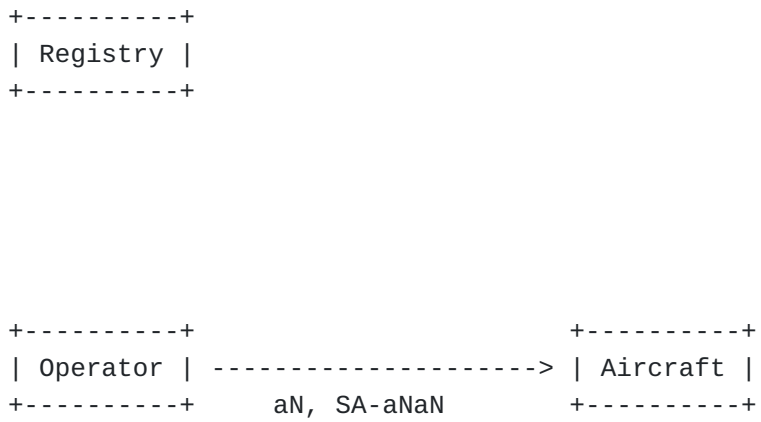


Figure 7: Operator Assisted Provision: Step 1

To start the Operator generates on behalf of the Aircraft a new keypair and Attestation: Aircraft N on Aircraft N (SA-aNaN). This keypair and certificate are injected into the Aircraft for it to generate Attestation: Aircraft 0 on Aircraft N (A-a0aN). After injecting the keypair and certificate, the Operator MUST destroy all copies of the keypair.

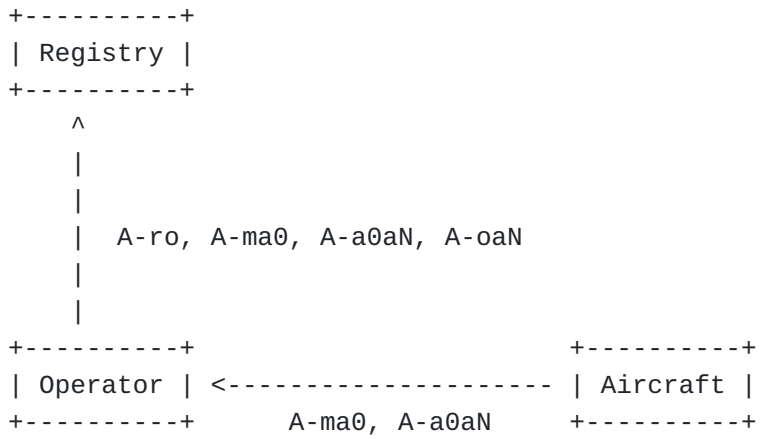


Figure 8: Operator Assisted Provision: Step 2

Attestation: Manufacturer on Aircraft 0 (A-ma0) and Attestation: Aircraft 0 on Aircraft N (A-a0aN) is extracted by the Operator and the following data items are sent to the Registry; Attestation: Registry on Operator (A-ro), Attestation: Manufacturer on Aircraft 0 (A-ma0), Attestation: Aircraft 0 on Aircraft N (A-a0aN), Attestation: Operator on Aircraft N (A-oaN).

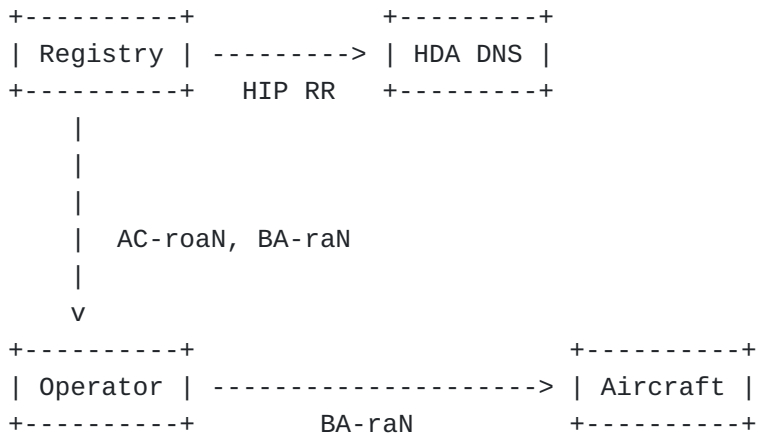


Figure 9: Operator Assisted Provision: Step 3

On the registry validation checks are done on all attestations as per the previous sections. Once complete then the registry checks for a DET collision, adding to the HDA if clear and generates Attestation Certificate: Registry on Operator on Aircraft N (AC-roaN) and Broadcast Attestation: Registry on Aircraft N (BA-raN). Both are sent back to the Operator.

The Operator securely inject BA-raN and securely stores AC-roaN of Aircraft N.

6.4.3. Initial Provisioning

A special form of provisioning is used when the Aircraft is first sold to an Operator. Instead of generating a new keypair, the built in keypair and certificate done by the Manufacturer is used to provision and register the aircraft to the owner.

For this either Standard or Operator Assisted methods can be used.

7. EPP Command Mappings

7.1. Common Attributes

There are a number of common attributes between the various EPP commands under DRIP that has specific encoding rules.

*The hi attribute is a Base64 encoding of the Host Identity.

*The det attribute is a string from of the DET.

7.2. EPP Query Commands

7.2.1. EPP <check> Command

7.2.1.1. Registry

7.2.1.2. Operator

7.2.1.3. Aircraft Serial Number

7.2.1.4. Aircraft Session ID

7.2.2. EPP <info> Command

7.2.2.1. Registry

7.2.2.2. Operator

7.2.2.3. Aircraft Serial Number

7.2.2.4. Aircraft Session ID

7.2.3. EPP <transfer> Command

Transfer semantics do not apply in DRIP, so there is no mapping defined for the EPP <transfer> command.

7.3. EPP Transform Commands

7.3.1. EPP <create> Command

7.3.1.1. Registry

The abbreviation field has a max of 6 characters, and is used by RID receivers to display a short decoded form for display of a received DET in the form of {RAA Abbreviation} {HDA Abbreviation} {Last 4 of DET Hash}. An example of this would be US FAA FE23. If the abbreviation is unknown then the receiver SHOULD use the hexadecimal encoding of the respective RAA/HDA field of the HID as the value in the form. For example if the HDA is unknown and the HDA value is 20 then the decoded display would be: DE 14 FE23. Typically for RAAs the abbreviation is RECOMMENDED to be set to the ISO 3166 country code (either Alpha-2 or Alpha-3) for the CAA. Dashes or underscores should be used in place of spaces.

The mfrCode field is only used by an MRA (Section 3.1.3.1) when registering with an IRM (Section 3.1.2.1) and holds the ICAO

assigned Manufacturer Code for ANSI CTA2063-A Serial Numbers. It has a max of 4 characters.

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<extension>
  <dripRegistry:dripRegistry xmlns:dripRegistry="urn:ietf:params:xml:ns:
    <dripRegistry:det>2001:0030:00a0:0145:a3ad:1952:0ad0:a69e</dripRegis
    <dripRegistry:hi></dripRegistry:hi>
    <dripRegistry:selfAttestation>Hex of SelfAttestation(Registry)</drip
    <dripRegistry:raa>10</dripRegistry:raa>
    <dripRegistry:hda>20</dripRegistry:hda>
    <dripRegistry:abbreviation>FAA</dripRegistry:abbreviation>
    <dripRegistry:mfrCode>MFR0</dripRegistry:mfrCode>
    <dripRegistry:postalInfo type="int">
      <dripRegistry:name>Federal Aviation Administration</dripRegistry:n
      <dripRegistry:phys_addr>
        <dripRegistry:street1>Orville Wright Federal Building</dripRegis
        <dripRegistry:street2>800 Independence Avenue SW</dripRegistry:s
        <dripRegistry:city>Washington</dripRegistry:city>
        <dripRegistry:sp>DC</dripRegistry:sp>
        <dripRegistry:pc>20591</dripRegistry:pc>
        <dripRegistry:cc>US</dripRegistry:cc>
      </dripRegistry:phys_addr>
    </dripRegistry:postalInfo>
    <dripRegistry:voice x="1234">1 (866) 835-5322</dripRegistry:voice>
    <dripRegistry:email>stephen.dickson@faa.gov</dripRegistry:email>
  </dripRegistry:dripRegistry>
</extension>
```

7.3.1.2. Operator

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<extension>
  <dripOperator:dripOperator xmlns:dripOperator="urn:ietf:params:xml:ns:
    <dripOperator:postalInfo type="int">
      <dripOperator:phys_addr>
        <dripOperator:street1>123 Example Dr.</dripOperator:street1>
        <dripOperator:street2>Suite 100</dripOperator:street2>
        <dripOperator:city>Dulles</dripOperator:city>
        <dripOperator:sp>VA</dripOperator:sp>
        <dripOperator:pc>20166-6503</dripOperator:pc>
        <dripOperator:cc>US</dripOperator:cc>
      </dripOperator:phys_addr>
    </dripOperator:postalInfo>
    <dripOperator:part107_acct_name>some_faa_account</dripOperator:part1
    <dripOperator:rec_flyer_id>123456</dripOperator:rec_flyer_id>
    <dripOperator:caaId></dripOperator:caaId>
    <dripOperator:det></dripOperator:det>
    <dripOperator:hi></dripOperator:hi>
    <dripOperator:selfAttestation>Hex of SelfAttestation(Operator)</drip
    <dripOperator:attestation>Hex of Attestation(Registry, Operator)</dr
  </dripOperator::dripOperator>
</extension>
```

7.3.1.3. Aircraft Serial Number

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<extension>
  <dripSerial:dripSerial xmlns:dripSerial="urn:ietf:params:xml:ns:dripSe
    <dripSerial:serial>0000F000000000000000</dripSerial:serial>
    <dripSerial:det></dripSerial:det>
    <dripSerial:hi></dripSerial:hi>
    <dripSerial:selfAttestation>Hex of SelfAttestation(Aircraft)</dripSe
    <dripSerial:broadcastAttestation>Hex of BroadcastAttestation(Registr
    <dripSerial:manufacturer>Drones R Us</dripSerial:manufacturer>
    <dripSerial:make>Fast Drone</dripSerial:make>
    <dripSerial:model>9000</dripSerial:model>
    <dripSerial:color>White</dripSerial:color>
    <dripSerial:material>Plastic</dripSerial:material>
    <dripSerial:weight>12.0</dripSerial:weight>
    <dripSerial:length>5.0</dripSerial:length>
    <dripSerial:width>4.0</dripSerial:width>
    <dripSerial:height>3.0</dripSerial:height>
    <dripSerial:numRotors>4</dripSerial:numRotors>
    <dripSerial:propLength>2.0</dripSerial:propLength>
    <dripSerial:batteryCapacity>5000</dripSerial:batterCapacity>
    <dripSerial:batteryVoltage>12</dripSerial:batteryVoltage>
    <dripSerial:batteryWeight>5.2</dripSerial:batteryWeight>
    <dripSerial:batteryChemistry>Lithium-Ion</dripSerial:batteryChemistr
    <dripSerial:takeOffWeight>15</dripSerial:takeOffWeight>
    <dripSerial:maxTakeOffWeight>25</dripSerial:maxTakeOffWeight>
    <dripSerial:maxPayloadWeight>10</dripSerial:maxPayloadWeight>
    <dripSerial:maxFlightTime>15</dripSerial:maxFlightTime>
    <dripSerial:minOperatingTemp>35</dripSerial:minOperatingTemp>
    <dripSerial:maxOperatingTemp>90</dripSerial:maxOperatingTemp>
    <dripSerial:ipRating>55</dripSerial:ipRating>
  </dripSerial:dripSerial>
</extension>
```

7.3.1.4. Aircraft Session ID

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<extension>
  <dripSession:dripSession xmlns:dripSession="urn:ietf:params:xml:ns:dri
    <dripSession:serial>0000F000000000000000</dripSession:serial>
    <dripSession:uasId></dripSession:uasId>
    <dripSession:sessionHi></dripSession:sessionHi>
    <dripSession:broadcastAttestation>Hex of BroadcastAttestation(Regist
    <dripSession:attestationCertificate>Hex of AttestationCertificate(Re
    <dripSession:operationalIntent></dripSession:operationalIntent>
    <dripSession:operationalIntentSrc>uss.example.com</dripSession:opera
    <dripSession:operatorId>NOP123456</dripSession:operatorId>
    <dripSession:operatorDet></dripSession:operatorDet>
    <dripSession:attestation>Hex of Attestation(Operator, Aircraft)</dri
    <dripSession:mutualAttestation>Hex of MutualAttestation(Registry, Op
    <dripSession:fa3>N1232456</dripSession:fa3>
    <dripSession:sessionStart>2022-04-09T15:43:13Z</dripSession:sessionS
    <dripSession:sessionEnd>2022-04-09T20:43:13Z</dripSession:sessionEnd
  </dripSession:dripSession>
</extension>
```

7.3.2. EPP <delete> Command

7.3.2.1. Registry

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <delete>
      <dripRegistry:delete xmlns:dripRegistry="urn:ietf:params:xml:ns:dr
        <dripRegistry:det>2001:0030:00a0:0145:a3ad:1952:0ad0:a69e</dripR
      </dripRegistry:delete>
    </delete>
    <clTRID>DEL-REGIS</clTRID>
  </command>
</epp>
```

7.3.2.2. Operator

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <delete>
      <dripOperator:delete xmlns:dripOperator="urn:ietf:params:xml:ns:dr
        <dripOperator:caaId></dripOperator:caaId>
        <dripOperator:det></dripOperator:det>
      </dripOperator:delete>
    </delete>
    <clTRID>DEL-OPER</clTRID>
  </command>
</epp>
```

7.3.2.3. Aircraft Serial Number

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <delete>
      <dripSerial:delete xmlns:dripSerial="urn:ietf:params:xml:ns:dripSe
        <dripSerial:serial>0000F000000000000000</dripSerial:serial>
      </dripSerial:delete>
    </delete>
    <clTRID>DEL-AIRCRFT</clTRID>
  </command>
</epp>
```

7.3.2.4. Aircraft Session ID

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <delete>
      <dripSession:delete xmlns:dripSession="urn:ietf:params:xml:ns:drip
        <dripSession:uasId></dripSession:uasId>
      </dripSession:delete>
    </delete>
    <clTRID>DEL-SID</clTRID>
  </command>
</epp>
```

7.3.3. EPP <renew> Command

Renewal semantics do not apply in DRIP, so there is no mapping defined for the EPP <renew> command.

7.3.4. EPP <transfer> Command

Transfer semantics do not apply in DRIP, so there is no mapping defined for the EPP <transfer> command.

7.3.5. EPP <update> Command

7.3.5.1. Registry

7.3.5.2. Operator

7.3.5.3. Aircraft Serial Number

7.3.5.4. Aircraft Session ID

8. RDAP Definitions

9. IANA Considerations

10. Security Considerations

10.1. DET Generation

Under the FAA [NPRM], it is expecting that IDs for UAS are assigned by the UTM and are generally one-time use. The methods for this however are unspecified leaving two options.

1. The entity generates its own DET, discovering and using the RAA and HDA for the target registry. The method for discovering a registry's RAA and HDA is out of scope here. This allows for the device to generate an DET to send to the registry to be accepted (thus generating the required Self-Attestation) or denied.

2. The entity sends to the registry its HI for it to be hashed and result in the DET. The registry would then either accept (returning the DET to the device) or deny this pairing.

Keypairs are expected to be generated on the device hardware it will be used on. Due to hardware limitations (see Section 10) and connectivity it is acceptable under DRIP to generate keypairs for the Aircraft on Operator devices and later securely inject them into the Aircraft (as defined in Section 6.4.2). The methods to securely inject and store keypair information in a "secure element" of the Aircraft is out of scope of this document.

In either case the registry must decide on if the HI/DET pairing is valid. This in its simplest form is checking the current registry for a collision on the DET and HI.

Upon accepting a HI/DET pair the registry MUST populate the required the DNS serving the HDA with the HIP RR and other relevant RR types (such as TXT and CERT). The registry MUST also generate the appropriate attestations/certificates for the given operation.

If the registry denied the HI/DET pair, because there was a DET collision or any other reason, the registry MUST signal back to the device being provisioned that a new HI needs to be generated.

11. Acknowledgements

*Scott Hollenbeck for his initial guidance with EPP/RDAP

12. Contributors

*Andrei Gurtov for his insights as a pilot

*Len Bayles for his help in formatting EPP definitions and creating an extension for FRED

13. References

13.1. Normative References

[F3411-19] "Standard Specification for Remote ID and Tracking", February 2020.

[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>.

13.2. Informative References

[dane-clients] Huque, S., Dukhovni, V., and A. Wilson, "TLS Client Authentication via DANE TLSA records", Work in Progress, Internet-Draft, draft-ietf-dance-client-auth-00, 24 March 2022, <https://www.ietf.org/archive/id/draft-ietf-dance-client-auth-00.txt>.

[drip-arch] Card, S. W., Wiethuechter, A., Moskowitz, R., Zhao, S., and A. Gurtov, "Drone Remote Identification Protocol (DRIP) Architecture", Work in Progress, Internet-Draft, draft-ietf-drip-arch-24, 10 June 2022, <https://www.ietf.org/archive/id/draft-ietf-drip-arch-24.txt>.

[drip-auth] Wiethuechter, A., Card, S., and R. Moskowitz, "DRIP Entity Tag Authentication Formats & Protocols for Broadcast Remote ID", Work in Progress, Internet-Draft, draft-ietf-drip-auth-14, 21 June 2022, <https://www.ietf.org/archive/id/draft-ietf-drip-auth-14.txt>.

[drip-requirements] Card, S., Ed., Wiethuechter, A., Moskowitz, R., and A. Gurtov, "Drone Remote Identification Protocol (DRIP) Requirements and Terminology", RFC 9153, DOI 10.17487/RFC9153, February 2022, <https://www.rfc-editor.org/info/rfc9153>.

[drip-rid] Moskowitz, R., Card, S. W., Wiethuechter, A., and A. Gurtov, "UAS Remote ID", Work in Progress, Internet-Draft, draft-ietf-drip-uas-rid-01, 9 September 2020, <https://www.ietf.org/archive/id/draft-ietf-drip-uas-rid-01.txt>.

[drip-secure-nrid-c2] Moskowitz, R., Card, S. W., Wiethuechter, A., and A. Gurtov, "Secure UAS Network RID and C2 Transport", Work in Progress, Internet-Draft, draft-moskowitz-drip-secure-nrid-c2-10, 27 June 2022, <https://www.ietf.org/archive/id/draft-moskowitz-drip-secure-nrid-c2-10.txt>.

[hhit-registries] Moskowitz, R., Card, S. W., and A. Wiethuechter, "Hierarchical HIT Registries", Work in Progress, Internet-Draft, draft-moskowitz-hip-hhit-registries-02, 9 March 2020, <https://www.ietf.org/archive/id/draft-moskowitz-hip-hhit-registries-02.txt>.

[NPRM] "Notice of Proposed Rule Making on Remote Identification of Unmanned Aircraft Systems", December 2019.

[RFC4398] Josefsson, S., "Storing Certificates in the Domain Name System (DNS)", RFC 4398, DOI 10.17487/RFC4398, March 2006, <https://www.rfc-editor.org/info/rfc4398>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/

RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC8005] Laganier, J., "Host Identity Protocol (HIP) Domain Name System (DNS) Extension", RFC 8005, DOI 10.17487/RFC8005, October 2016, <https://www.rfc-editor.org/info/rfc8005>.

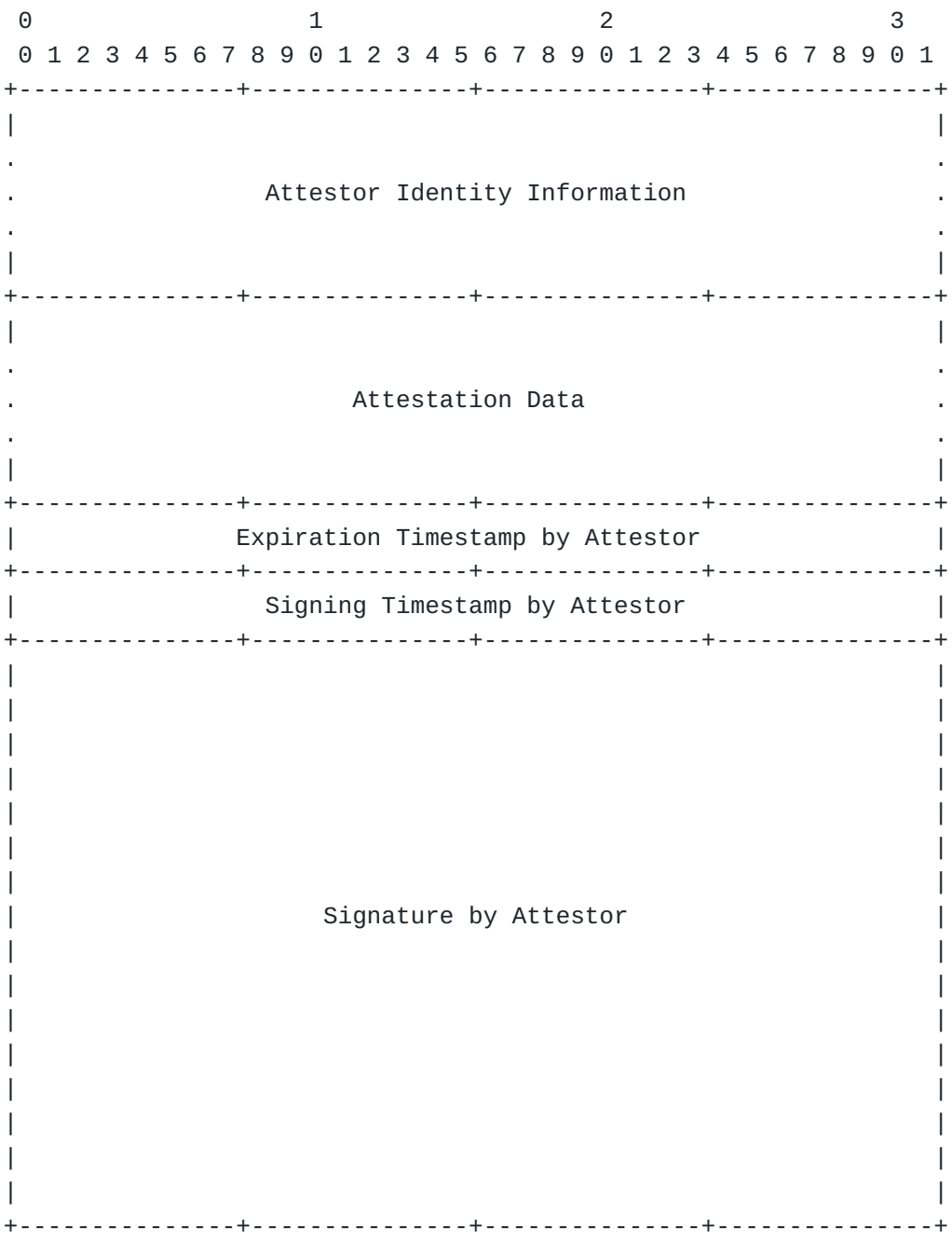
[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/info/rfc8392>.

Appendix A. DRIP Attestations & Certificates

See [drip-arch] for definitions of claim, assertion, attestation and certificate as used in this document.

A.1. Attestation Structure

All Attestations (Appendix A.2) and Certificates (Appendix A.3) under DRIP share the following format structure:



Attestor Identity Information: (0, 16-bytes or 120-bytes)
    Field containing Attestor Identity Information in various forms.

Attestation Data:
    A field of variable length containing the attestation data.

Expiration Timestamp by Attestor (4 bytes):
    Timestamp denoting recommended time to trust data to.

Signing Timestamp by Attestor (4 bytes):
    Current time at signing.

```
Attestor Signature (64 bytes):
    Signature over preceding fields using the keypair of
    the Attestor.
```

Figure 10: Attestation Structure

A.1.1. Attestor Identity Information

This can be either of the following:

1. Attestor DET: 16-bytes

2. Attestor Self-Attestation: 120-bytes

A specific definition of an Attestation (Appendix A.2) or Certificate (Appendix A.3) defines which of these are used.

A.1.2. Attestation Data

The data being attested to. It can be one of the following forms:

1. Claims

2. Assertions

3. Attestations

This field is variable length with no limit and specific definitions of an Attestation (Appendix A.2) or Certificate (Appendix A.3) indicate the fields, size and ordering of any subfields.

A.1.3. Expiration Timestamp

A UTC timestamp set some time into the future to indicate a point the Attestation Structure should not be trusted.

The time delta into the future is of important concern as replay attacks on during flight could compromise the goals of DRIP. Attestations and certificates intended for public use and lower in the tree (importantly any generated for a Session ID (Section 6.4)).

For this reason deltas SHOULD be kept as short as possible for the given use-case to avoid issues with replays.

A.1.4. Signing Timestamp

A UTC timestamp set to the time when the Attestation Structure was signed.

A.1.5. Signature

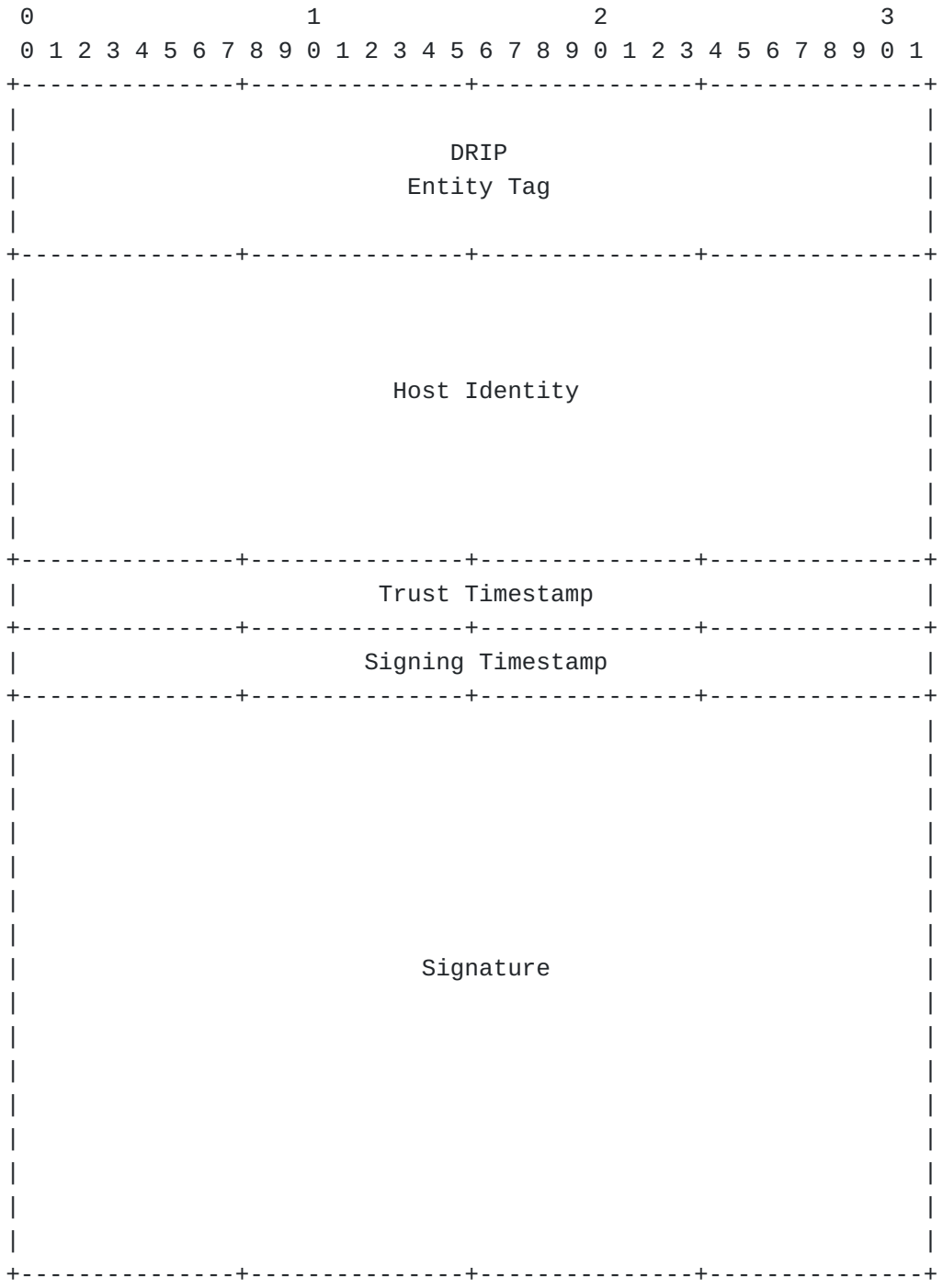
An EdDSA25519 signature using the signing parties private key over the preceding fields in the Attestation Structure.

> Note: the preceding fields of the Attestation Structure actually form an Assertion, with all fields acting as Claims

A.2. Attestations

A.2.1. Self-Attestation (SA-x)

The only attestation to use a claim (the Host Identity) in the Attestation Data with the DET acting as the Attestor Identity Information.

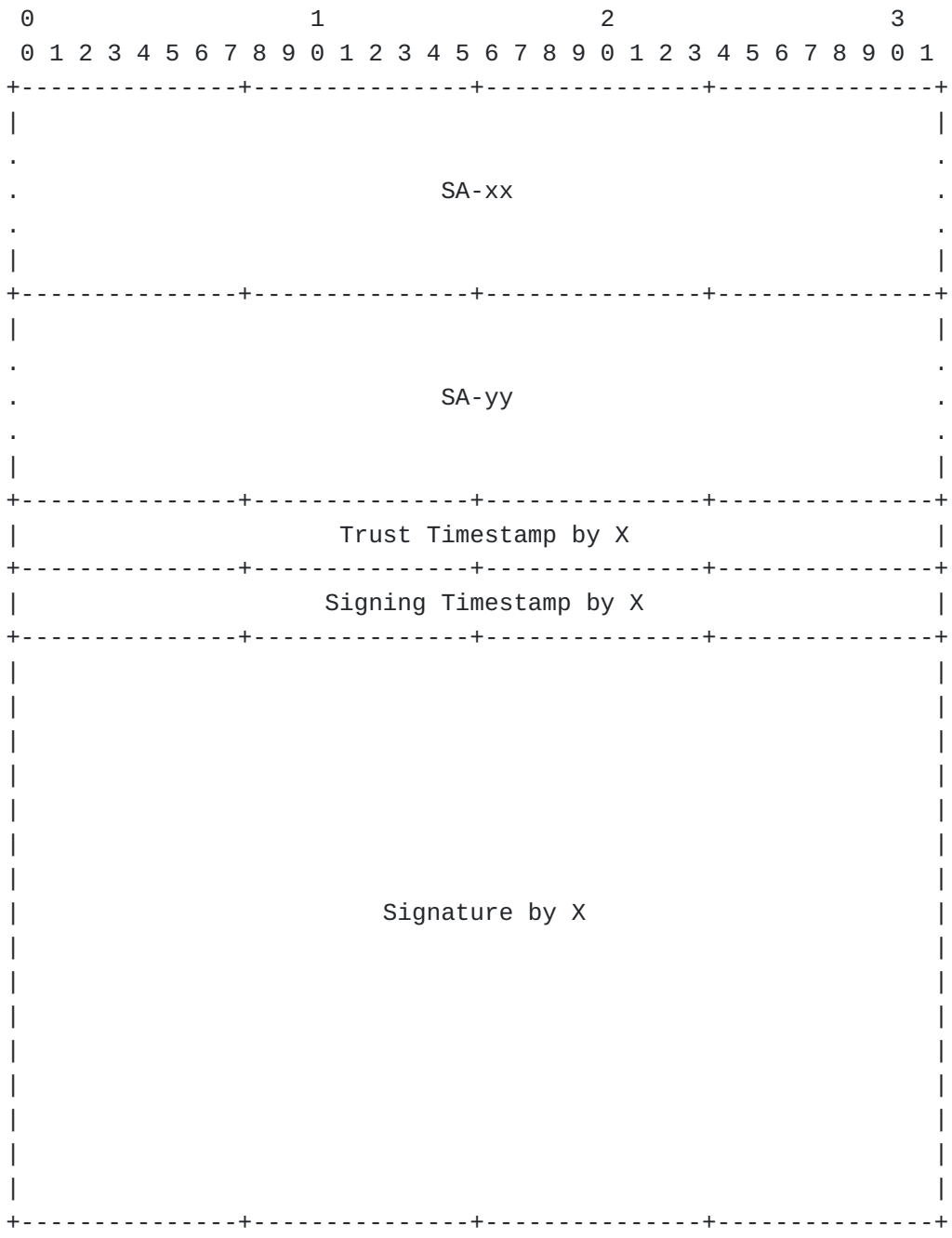


Length = 120-bytes

Figure 11: DRIP Self-Attestation

A.2.2. Attestation (A-x.y)

The standard first level DRIP Attestation form using a Self-Attestations of the signer and of the data being signed.

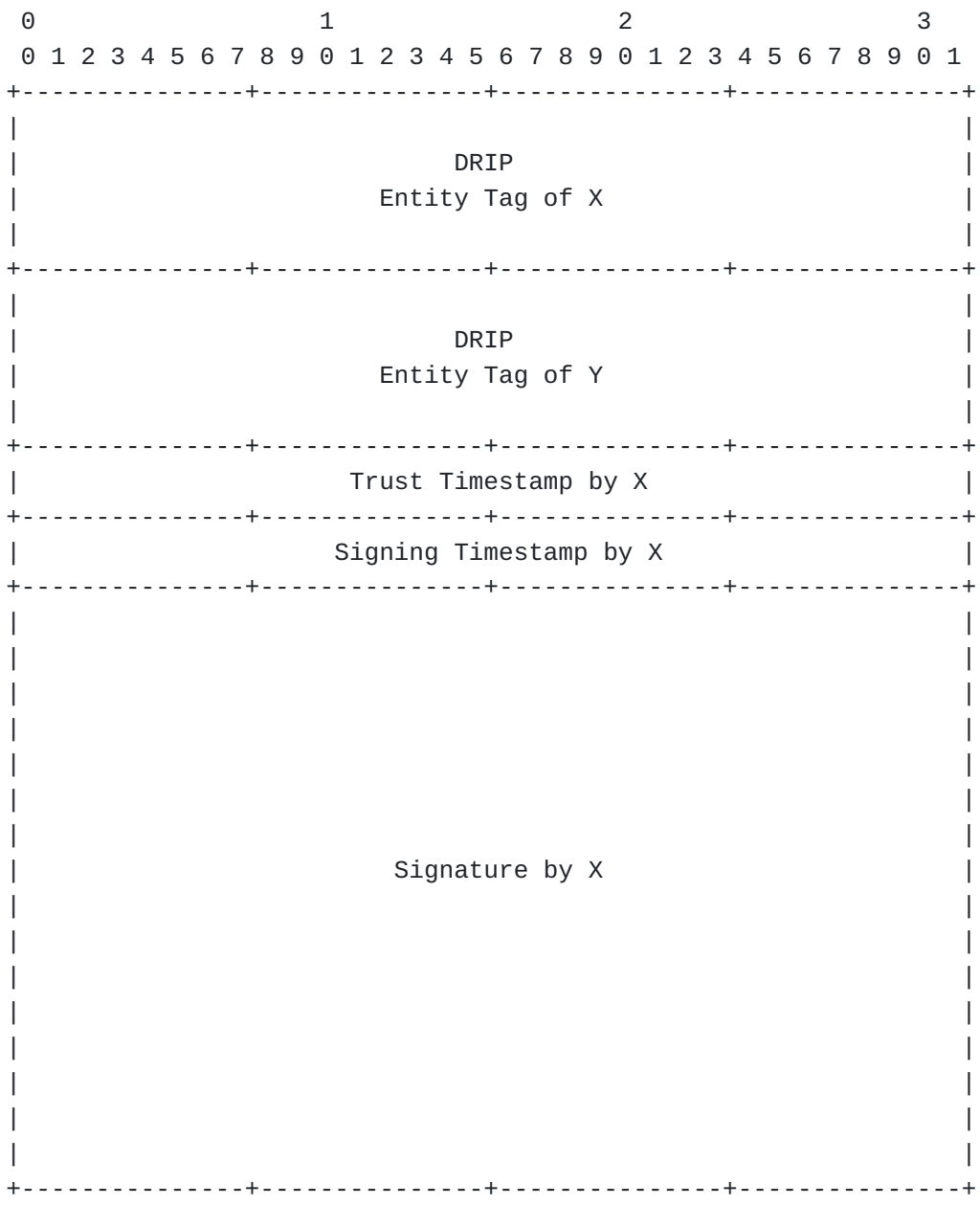


Length = 312-bytes

Figure 12: DRIP Attestation

A.2.3. Concise Attestation (CA-x.y)

In constrained environments and when there is the guarantee of being able to lookup the DETs to obtain HIs this attestation can be used.



Length = 104-bytes

Figure 13: DRIP Concise Attestation

A.2.4. Mutual Attestation (MA-x.y)

An attestation that perform a sign over an existing Attestation where the signer is the second party of the embedded attestation. The DET of party Y is used as the Attestor Identity Information (Appendix A.1.1).

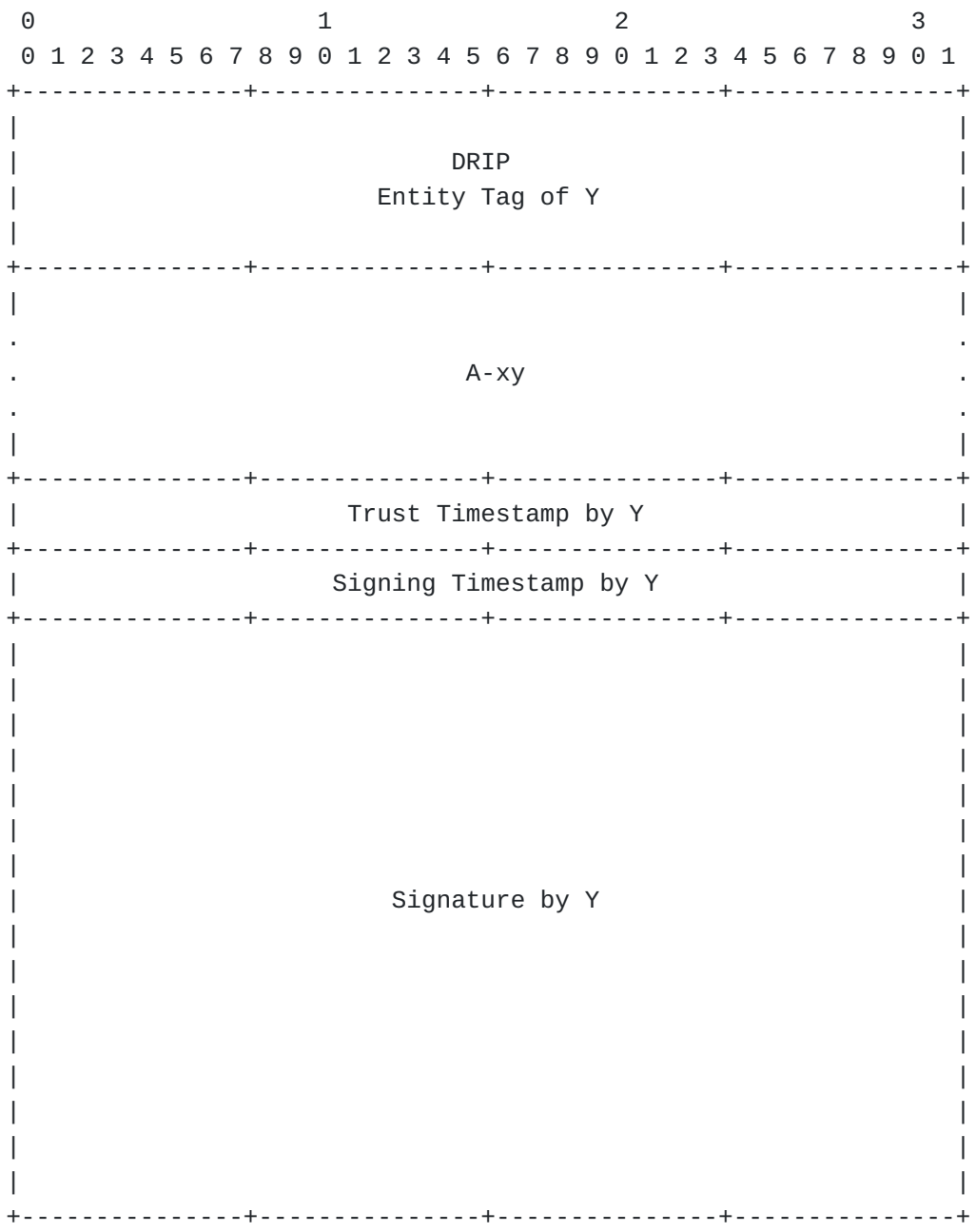
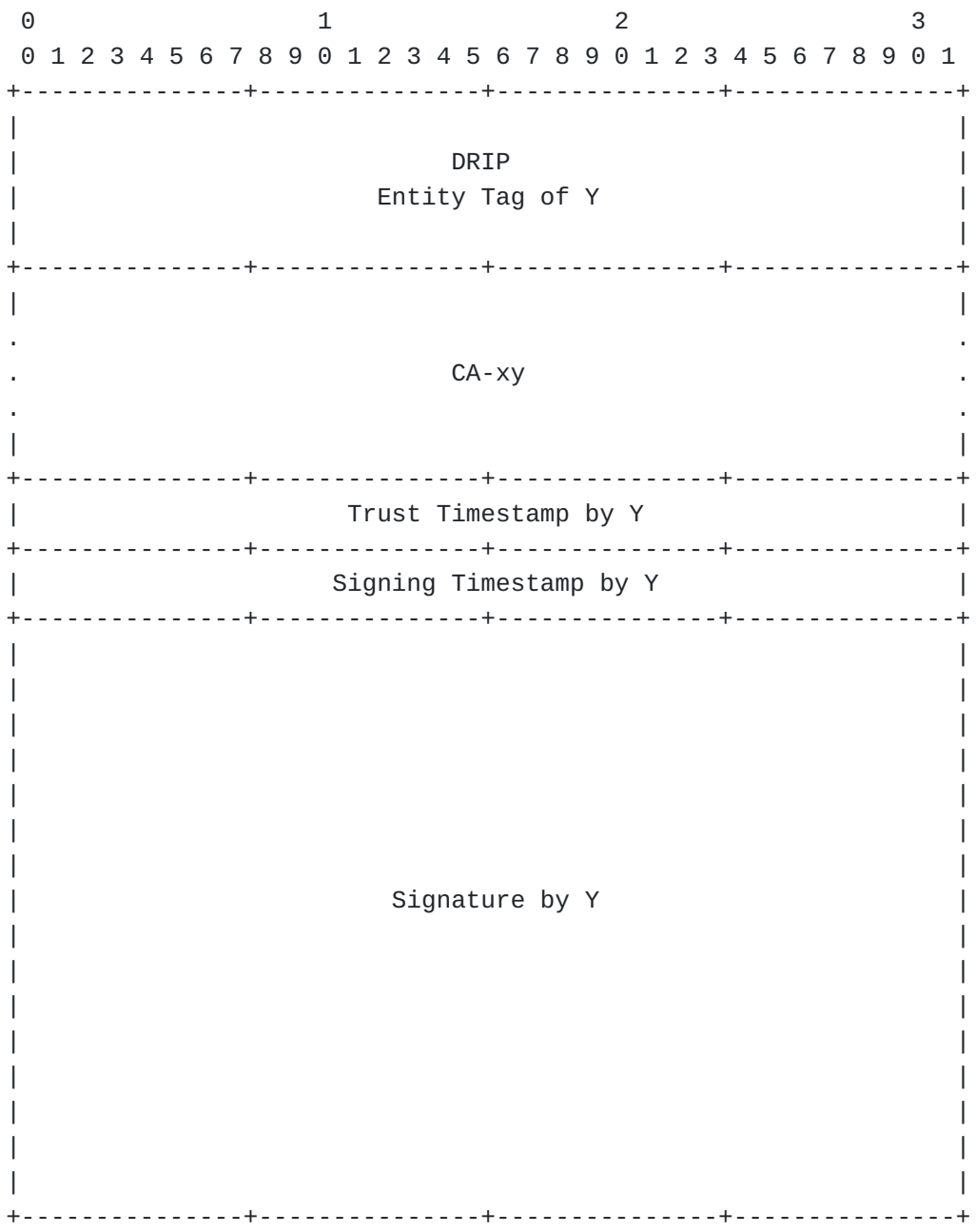


Figure 14: DRIP Mutual Attestation

A.2.5. Link Attestation (LA-x.y)

An attestations that perform a sign over an existing Concise Attestation where the signer is the second party of the embedded attestation. The DET of party Y is used as the Attestor Identity Information (Appendix A.1.1).

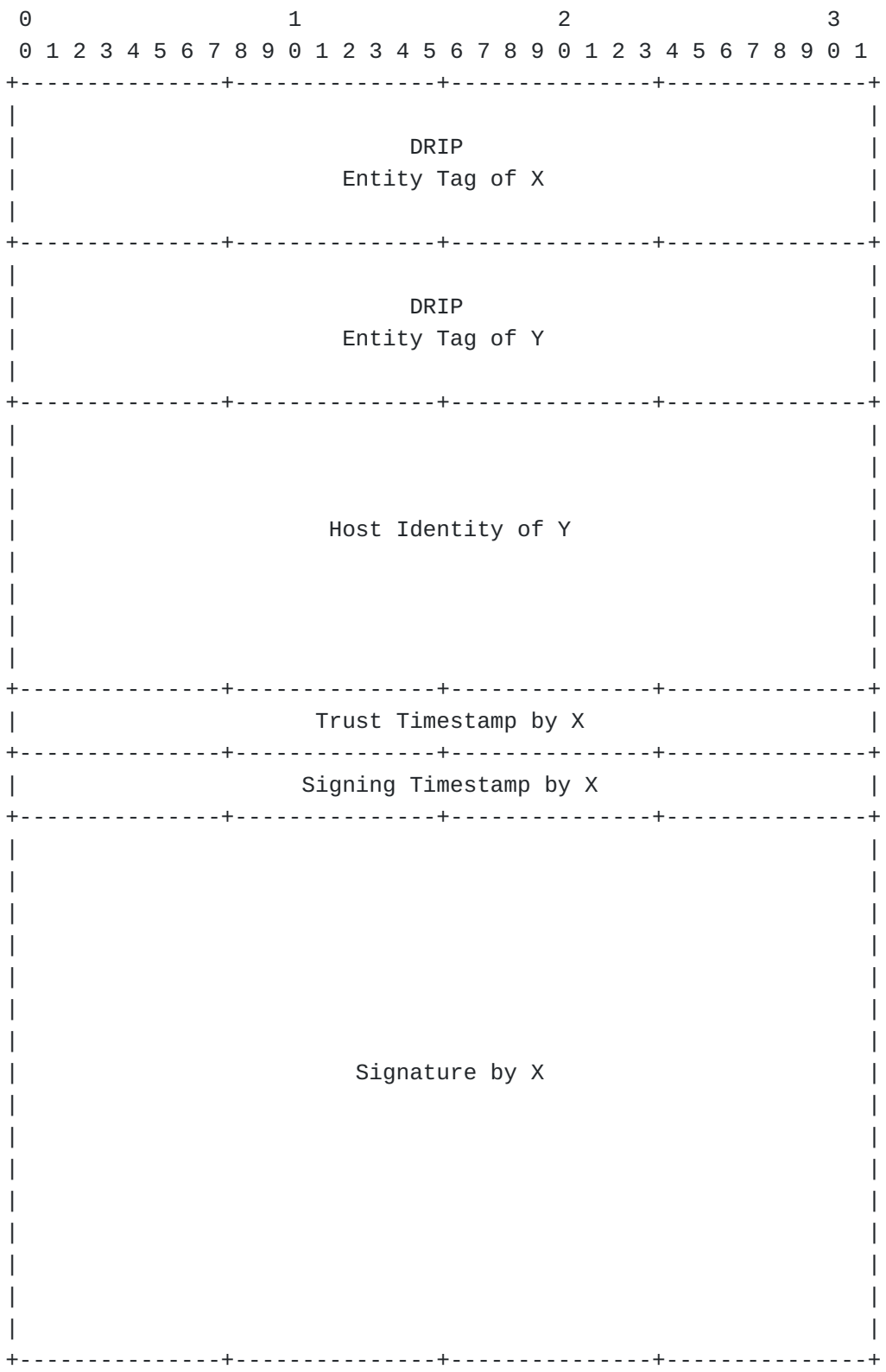


Length = 192-bytes

Figure 15: DRIP Link Attestation

A.2.6. Broadcast Attestation (BA-x.y)

Required by DRIP Authentication Formats for Broadcast RID ([drip-auth]) to satisfy [drip-requirements] GEN-1 and GEN-3.



Length = 136-bytes

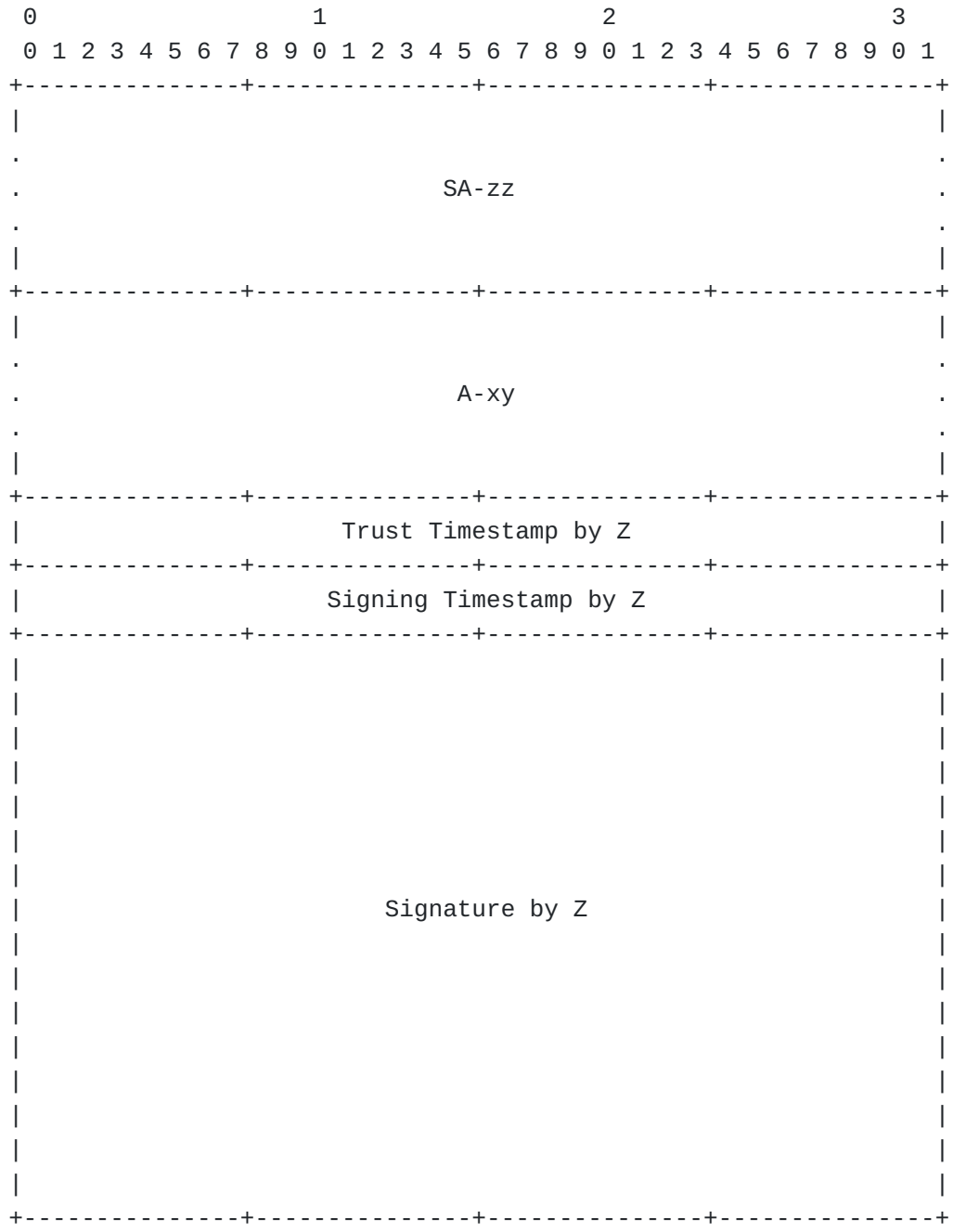
Figure 16: DRIP Broadcast Attestation

A.3. Certificates

In DRIP certificates are signed by a third party that has no stake in the claims/assertions/attestations being attested to.

It is analogous to a third party in legal system that signs a document as a "witness" and bears no responsibility in the document.

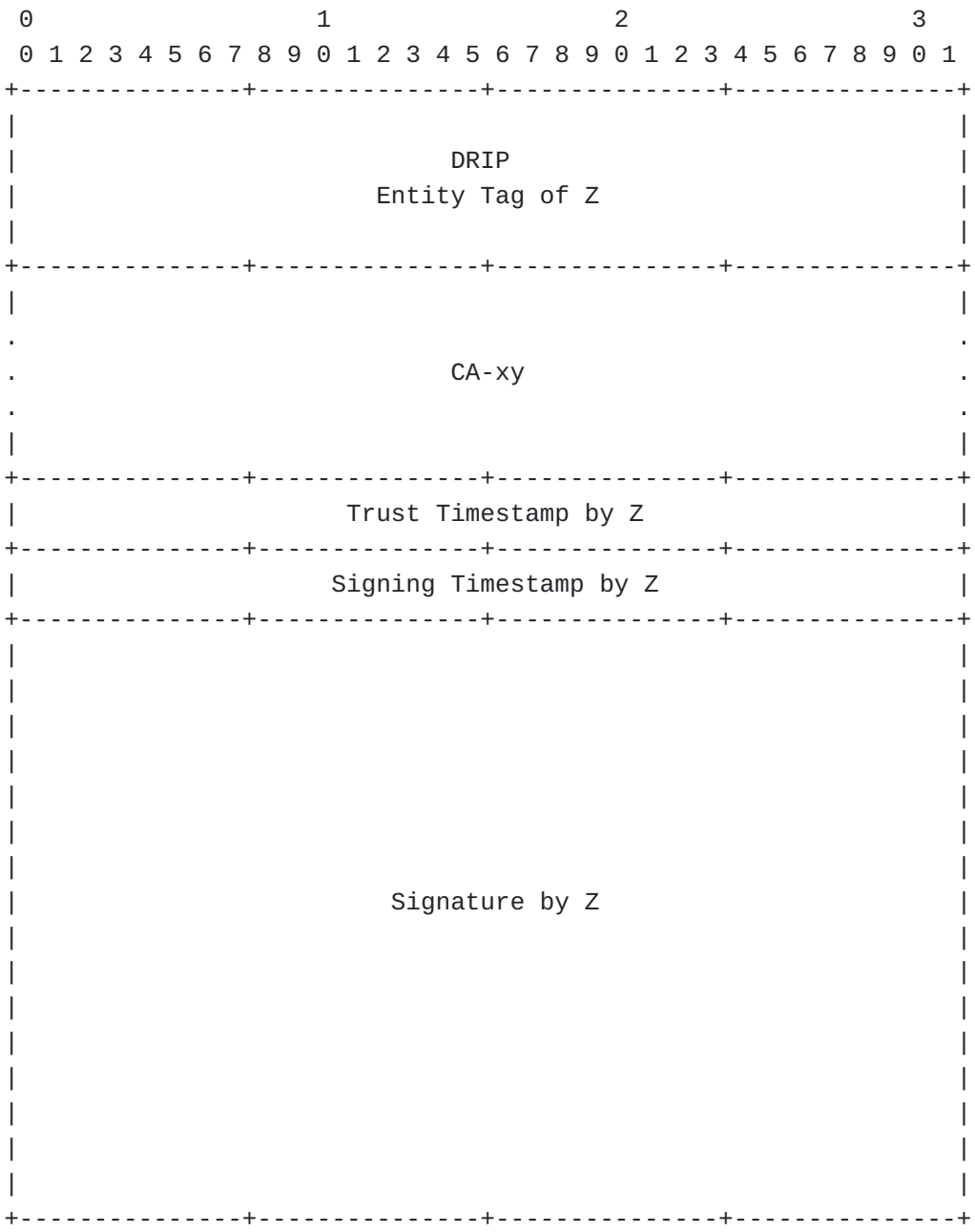
A.3.1. Attestation Certificate (AC-z.x.y)



Length = 504-bytes

Figure 17: DRIP Attestation Certificate

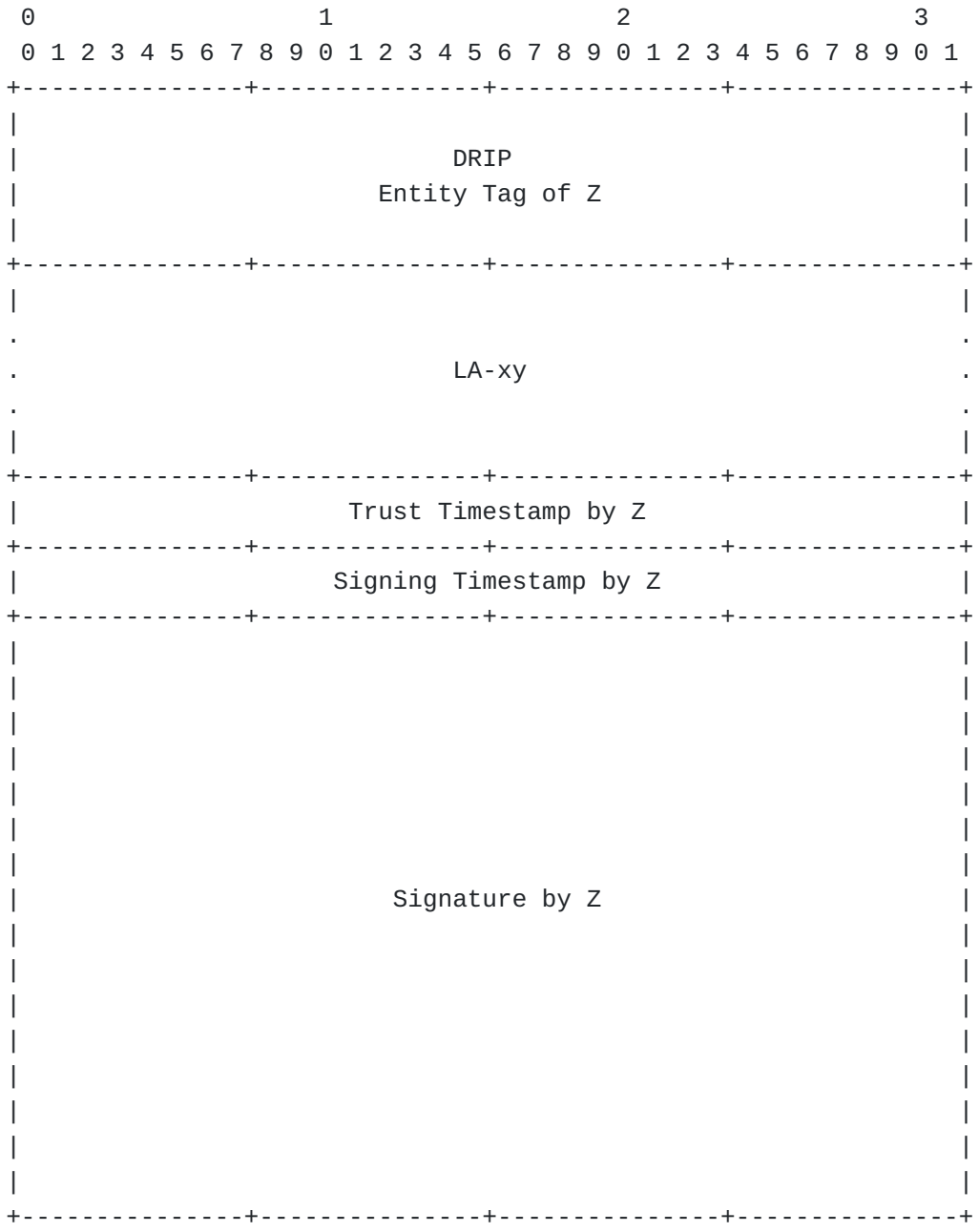
A.3.2. Concise Certificate (CC-z.x.y)



Length = 192-bytes

Figure 18: DRIP Concise Certificate

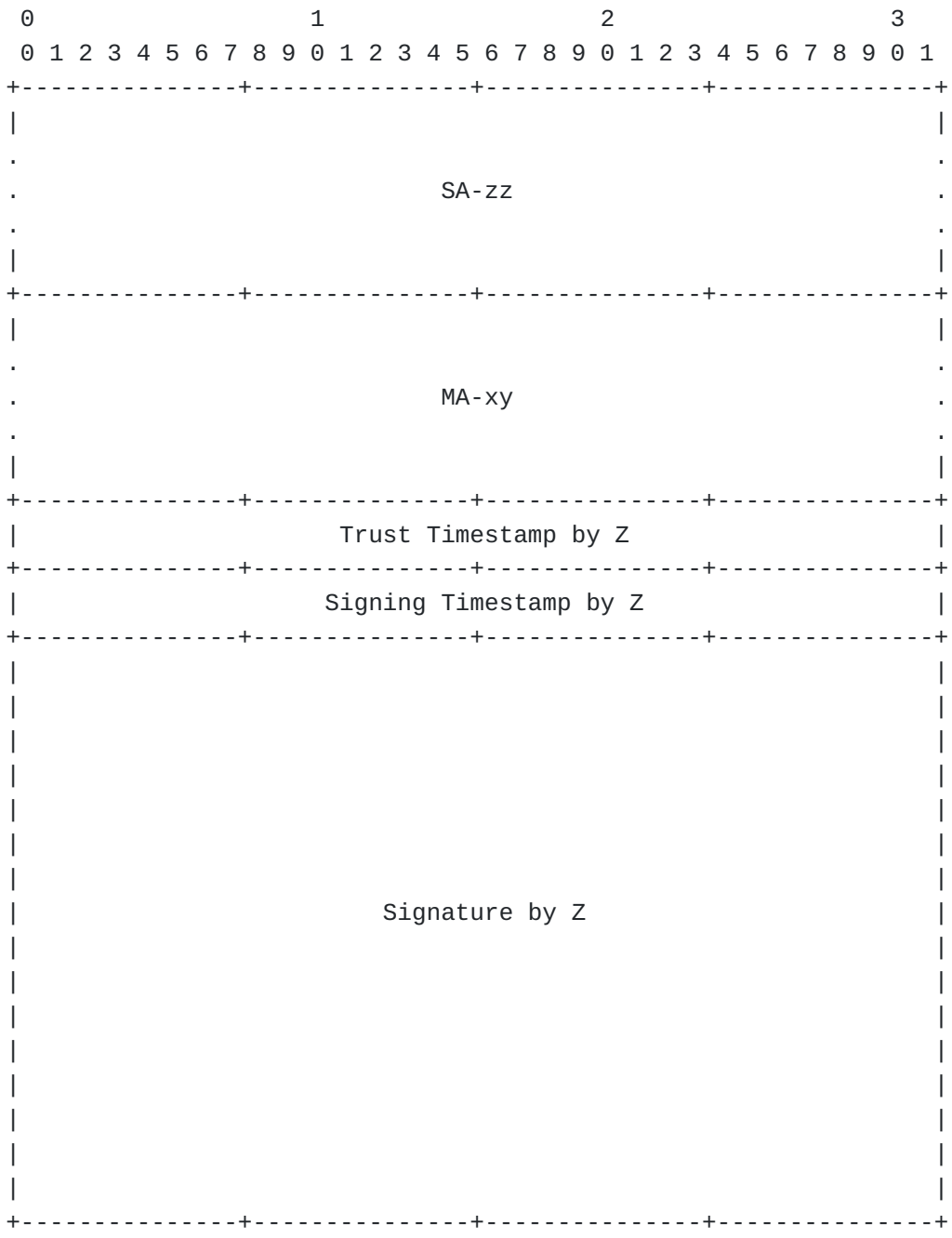
A.3.3. Link Certificate (LC-z.x.y)



Length = 300-bytes

Figure 19: DRIP Link Certificate

A.3.4. Mutual Certificate (MC-z.x.y)



Length = 576-bytes

Figure 20: DRIP Mutual Certificate

A.4. Abbreviations & File Naming Conventions

The names of attestation and certificates can become quite long and tedious to write out. As such this section provides a guide to a somewhat standardized way they are written in text.

A.4.1. In Text Abbreviation

In a long form the name of a particular attestation/certification can be written as follows:

- *Self-Attestation: Unmanned Aircraft
- *Attestation: Operator on Aircraft or Attestation: Operator, Aircraft
- *Attestation Certificate: Registry on Operator on Aircraft or Attestation Certificate: Registry, Operator, Aircraft

When multiple entities are listed they can be separated by either on or by ,. These long forms can be shortened:

- *SA(Unmanned Aircraft) or SA-ua
- *A(Operator, Unmanned Aircraft) or A-op.ua
- *AC(Registry, Operator, Aircraft) or AC-reg.op.ua

Typical abbreviations for the entity can be used such as Unmanned Aircraft being shorthanded to ua.

A.4.2. File Naming

For file naming of various certificates a similar format to the short form is used:

- *sa-{hash of entity}
- *a-{hash of entity x}_{hash of entity y}
- *ac-{hash of entity z}_{hash of entity x}_{hash of entity y}

Some examples of file names:

- *sa-79d8a404d48f2ef9.cert
- *a-120b8f25b198c1e1_79d8a404d48f2ef9.cert
- *ac-aac6b00abba268b7_120b8f25b198c1e1_79d8a404d48f2ef9.cert

Appendix B. X.509 Certificates

B.1. Certificate Policy and Certificate Stores

X.509 certificates are optional for the DRIP entities covered in this document. DRIP endpoint entities (EE) (i.e., UA, GCS, and Operators) may benefit from having X.509 certificates. Most of these

certificates will be for their DET and some will be for other UAS identities. To provide for these certificates, some of the other entities covered in this document will also have certificates to create and manage the necessary PKI structure.

Any Certificate Authority (CA) supporting DRIP entities SHOULD adhere to the ICAO's International Aviation Trust Framework (IATF) Certificate Policy [ICAO-IATF-CP-draft]. The CA(s) supporting this CP MUST either be a part of the IATF Bridge PKI or part of the IATF CA Trust List.

EEs may use their X.509 certificates, rather than their rawPublicKey (i.e. HI) in authentication protocols (as not all may support rawPublicKey identities). Some EE HI may not be 'worth' supporting the overhead of X.509. Short lived DETs like those used for a single operation or even for a day's operations may not benefit from X.509. Creating then almost immediately revoking these certificates is a considerable burden on all parts of the system. Even using a short notAfterDate will completely mitigate the burden of managing these certificates. That said, many EEs will benefit to offset the effort. It may also be a regulator requirement to have these certificates.

Typically an HDA either does or does not issue a certificate for all its DETs. An RAA may specifically have some HDAs for DETs that do not want/need certificates and other HDAs for DETs that do need them. These types of HDAs could be managed by a single entity thus providing both environments for its customers.

It is recommended that DRIP X.509 certificates be stored as DNS TLSA Resource Records. This not only generally improves certificate lookups, but also enables use of DANE [RFC6698] for the various servers in the UTM and particularly DRIP registry environment and DANCE [dane-clients] for EEs (e.g. [drip-secure-nrid-c2]). All DRIP certificates MUST be available via RDAP. LDAP/OCSP access for other UTM and ICAO uses SHOULD also be provided.

B.2. Certificate Management

(mostly TBD still)

PKIX standard X.509 issuance practices should be used. The certificate request SHOULD be included in the DET registration request (Section 6). A successful DET registration then MUST include certificate creation, store, and return to the DET registrant.

Certificate revocation will parallel DET revocation. TLSA RR MUST be deleted from DNS and RDAP, LDAP, and OCSP return revoked responses. CRLs SHOULD be maintained per the CP.

Details of this are left out, as there are a number of approaches and further research and experience will be needed.

B.3. Examples

TBD

B.4. Alternative Certificate Encoding

(CBOR encoded certs here. TBD)

Appendix C. Blockchain-based Registries

The implementation of the registries and Network Remote Identification (Network RID; identify a UA through the network) in DRIP is yet to be determined. Blockchain, being synonymous with ledger, is a technology that could naturally fulfil the role of a registry, while simultaneously offering its benefits such as auditability, persistency and decentralization. We suggest that blockchain is an ample candidate to be used as registry within DRIP. We also show that it can be used to effectively leverage Network RID in certain scenarios. Thus 1) We propose a novel drone ID architecture based on Hyperledger Iroha and describe its proof-of-concept implementation with DRIP. 2) Its performance and scalability is empirically evaluated. 3) We perform an informal security analysis of the system against various types of attacks [Secure Drone Identification with Hyperledger Iroha].

Figure 1: Architecture using blockchain as registry for DRIP

The proposed architecture is presented in Fig. 1. It consists of the usual actors in a UAS network, along with the blockchain registry based on Hyperledger Iroha. Key components: o Authorized users (administrators) can register new UAs to the network, and store with them any relevant data such as public keys and certificates.

Drones can either send location updates directly to the blockchain, given that they are connected to the Internet, or send location updates to their connected Ground Control Station (GCS) that forwards it on behalf of the drones. o Observers can receive drone messages either through bluetooth and WiFi broadcasts from drones, or by polling the blockchain. They can also fetch the public key associated with a drone in order to validate its messages. o The blockchain network and its nodes are an entirely separate entity, no other actor participates in the consensus of new blocks.

Actors within DRIP (except observers) will be registered as accounts on the blockchain network. Each of these accounts will have their DRIP identities, certificates and public keys stored and available so that they can be validated and used for validation by any account

on the blockchain. Note that DRIP crypto key-pairs are separate from the blockchain crypto key-pairs. DRIP key-pairs are used to sign, verify and validate DRIP identities and messages, while the blockchain key-pairs are used to sign, verify and validate transactions on the blockchain.

The DRIP requirements for a registry are the following: (1) REG-1: Public lookup (2) REG-2: Private lookup (3) REG-3: Provisioning (of static/dynamic data of UAs) (4) REG-4: AAA Policy

REG-1 & REG-2. In Hyperledger Iroha, accounts are created on domains. The same account name can be used for multiple domains, and these are seen as separate accounts on Iroha. PII for an account can therefore be stored on a separate account (with the same account name) existing on a separate domain, that only allows certain accounts to view its account details. Accordingly, a registry using Iroha would need at least two domains associated with it for any given account, one for public lookup and one for private lookup.

REG-3 & REG-4. The details for an account are set with a key/value pair. Key/value pairs can not be removed once they are set, values can only be modified through the corresponding key. Furthermore, the account that sets a key/value pair is included in the account details as a key/value pair itself, meaning one account can not modify details set by another account. See Listing 1 for clarification. Notice that both accounts have set the same key but contain different values. This sort of implementation supports both non-repudiation, but also trust in the sense that a drone (assuming the drone is not compromised) can always trust its own data, and does not have to interpret data coming from other accounts. Similarly, other accounts accessing another account's data can trust that it is set by the corresponding account (e.g. fetching gps data).

Authors' Addresses

Adam Wiethuechter
AX Enterprize, LLC
4947 Commercial Drive
Yorkville, NY 13495
United States of America

Email: adam.wiethuechter@axenterprize.com

Stuart Card
AX Enterprize, LLC
4947 Commercial Drive
Yorkville, NY 13495
United States of America

Email: stu.card@axenterprize.com

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237
United States of America

Email: rgm@labs.htt-consult.com

Jim Reid
RTFM llp
St Andrews House
382 Hillington Road, Glasgow Scotland
G51 4BL
United Kingdom

Email: jim@rfc1035.com