ID: MCP IDsec 3

Version: 1.0

MCC Identity Management and Security: Public Key Infrastructure (PKI)

In addition to a unique ID in the form of an MCP MRN each MCP entity is provided with a cryptographic identity. This consists of a public/private key pair and a certificate for the public key bound to their ID. In the following, we describe the concept of the PKI that enables this, and a first set of requirements for it. We also identify issues that need to be addressed and refined in the future.

We proceed as follows. In Section 1 we explain the MCP core concepts of cryptographic identity. Section 2 details the decentral PKI. In Section 3 we specify the requirements on cryptographic keys and mechanisms. In Section 4 the format of MCP certificates is described. Moreover, in Section 5 we show how a service can use an intermediary level of service certificates. For example, this is necessary if a service comes with cryptographic requirements that do not allow the direct use of the MCP ID credentials. Finally, in Section 6 we identify further aspects to be considered.

1 CRYPTOGRAPHIC IDENTITY

The cryptographic ID of an MCP entity consists of a public/private key pair and a certificate bound to their MRN. The certificate must be issued by the identity provider responsible for the entity. The latter is clearly defined by the IPID string within the MRN of the entity.

Given an entity with MRN A (short: entity A), and its identity provider P, we use the following notation:

- pk_A is the public key of A, and pr_A is the private key of A respectively,
- $cert_P(A, pk_A, V)$ is the certificate of A signed by its identity provider P. The certificate contains the MRN A, the public key of A, and the validity period V of the certificate. (The precise format is provided in Section 3.3.)

The key pair is for use with a digital signature scheme. Hence, each MCP entity A can be verified by another party B to be the originator of a message or other data. As usual this involves the following steps:

1. Entity A signs the message, say M, using its private key $pr_{A.}$ The result is a cyphertext C.
2. Entity A makes available its certificate $cert_P(A, pk_A, V)$, and transmits the signed message M||C.
3. Entity B obtains the certificate, and receives the signed message.
4. Entity B validates the certificate. As a result B trusts that pk_A is the valid public key of the MCP entity with MRN A. (Necessary requirements on certificate validation will be specified.)
5. Entity B uses pk_A to verify whether the ciphertext C is indeed the digital signature of M. If the verification is successful then B has assurance that M indeed originates from A. (Note that without the fourth step B only has assurance that M originates from the holder of the private key counterpart of pk_A.)

Note that B does not necessarily need to be an MCP entity.

At the time of writing the MCC does not prescribe a policy on how to use ID credentials. They could be used as long-term credentials to obtain short-term credentials for use for a service, or they could be directly used as working credentials.

2 DECENTRAL PKI

One of the principles of the MCP is to make do without a global notion of trust: in the international context of the MCP we cannot expect that all parties trust each other and each other's security management uniformly. Rather the goal of the MCP is to provide the transparency that enables organizations to decide on whom to trust in which context, and to provide the technical framework to translate such decisions into executable policies. For the PKI we put forward the following three principles:

1. A security breach within the realm of one identity provider's PKI instance shall not enable an attacker to impersonate an entity within the realm (i.e. namespace) of another identity provider.
2. A security breach within an organization or set of organizations predefined by the MCC to carry out some task shall not enable an attacker to impersonate any MCP entity, unless the identity provider of the entity coincides with (one of) the organization(s). In short this means identity providers' PKI instances can always remain secure independently from any central or distributed management by the MCC.
3. It is possible for everyone to obtain assurance as to the security level of any identity provider's PKI instance.

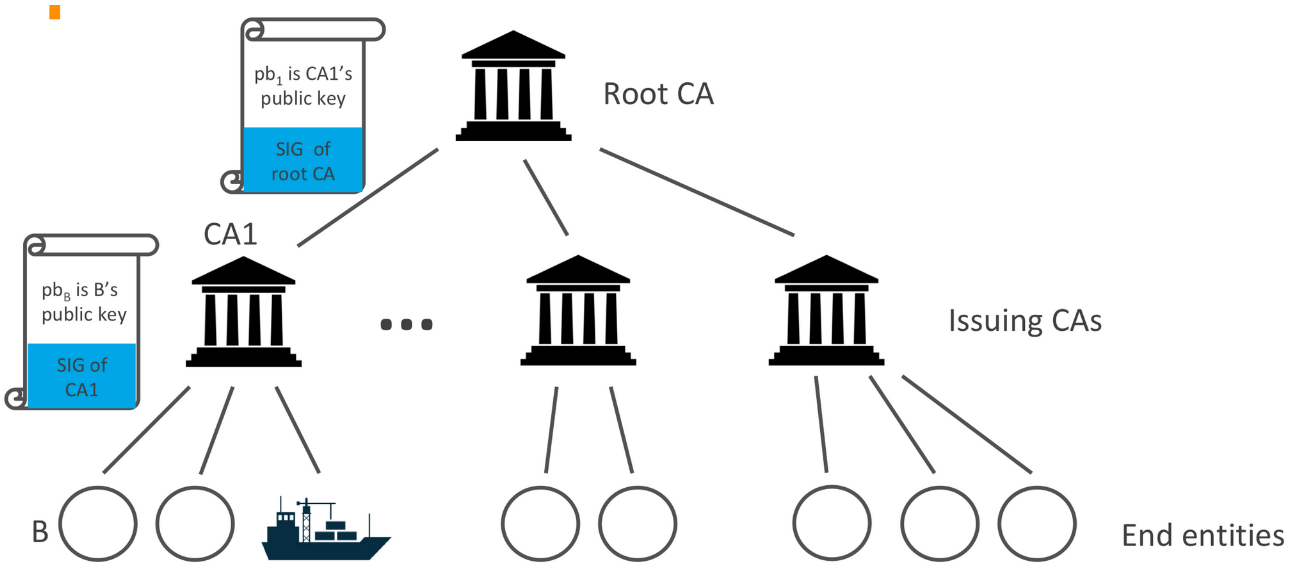

Figure 2: Hierarchical X.509 PKI Structure

The first two principles immediately imply that a classical hierarchical PKI with a root CA hosted by the MCC won't do. We illustrate this by giving examples of impersonation attacks. Assume a hierarchical PKI structure with MCC root CA as shown in Fig. 2. To verify a MCP certificate $cert_P(A, pk_A, V)$ a receiving party has to verify the signature of the identity provider P with the public key of P provided in an intermediary certificate $cert_{MCC}(P, pk_P, V)$ issued by the MCC root CA. Further, to verify the intermediary certificate the receiving party has to verify the signature of the MCC with the public key provided in the MCC root certificate $cert_{MCC}(MCC, pk_{MCC}, V)$. The latter provides the trust anchor accepted by the receiving party.

Examples:

a. *Single point of attack:* Assume the MCC root key pr_{MCC} is compromised. This will allow an attacker to impersonate any MCP entity A. Say P is the identity provider responsible for A. First, the attacker generates a key pair $pk_{I(P)}$, $pr_{I(P)}$ and generates a fake certificate for P with his own key $pk_{I(P)}$: $cert_{MCC}(P, pk_{I(P)}, V)$. This is possible since the attacker knows pr_{MCC}. Second, the attacker generates a key pair $pk_{I(A)}$, $pr_{I(A)}$ and generates a fake certificate for A and his own key $pk_{I(A)}$: $cert_{I(P)}(P, pk_{I(A)}, V)$. This is possible since he knows $pr_{I(P)}$. Altogether, the attacker can now present a valid certificate chain that establishes $pk_{I(A)}$ to be the public key of A while he knows the private counterpart $pr_{I(A)}$. Hence, he can impersonate A. Altogether this violates principle 2 above.

b. *Weakest Link I:* Say the attacker wishes to impersonate entity A of identity provider P. Note that in classic X.509 certificate validation it is only verified that there is a certificate chain up to a trusted root certificate. Say the attacker can easily obtain fake certificates signed by another identity provider P', perhaps, because the attacker is a state actor and P' is under his governance. Then, analogously to above, he only needs to generate his own key pair $pb_{I(A)}$, $pr_{I(A)}$ and generate a fake certificate for A and $pb_{I(A)}$ signed by P': $cert_{P'}(A, pb_{I(A)}, V)$. Since there is no check whether P' is indeed the identity provider of A this gives the attacker a valid certificate chain and corresponding private key, with which he can impersonate A. This violates principle 1 above.

c. *Weakest Link II:* Assume P is an identity provider of low security level, e.g. with a vetting procedure that can easily be undermined. Assume an attacker aims to join the MCP under a false identity so that he is able to inject fake messages without the risk of being traced. The attacker will simply choose P as the identity provider from whom to obtain his false identity. Without principle 3 in place a receiving party has no way to consider the low security level of P when processing the information within the message.

This motivates the following requirements:

PKI1.1 (PKI Structure) There shall be no root CA at the top level of the MCC. Every identity provider that hosts a PKI instance is to provide their own root CA.

PKI1.2. (Validation of IPID) When a receiving party verifies a MCP certificate, say $cert_P(A, pk_A, V)$, it must verify that the certificate is indeed signed by the identity provider responsible for A. The identity provider responsible for A can be read by the receiving party from the IDIP string within the MRN A.

The following requirements ensure that information on root certificates and security levels are made publicly available.

PKI1.3. Every identity provider is to publish their currently valid root certificate in a suitable fashion. For example, this can be made accessible via their web page, or they can commission a generally accepted authority or assurer to do so.

PKI1.4. Every identity provider is to make their security level publicly accessible in a suitable fashion. For example, they can provide a link from their web page to the database of their certification body.

From this the MCC will provide a secure way to automatically find and give basic trust in the authenticity of the MCP identity providers.

PKI1.5. The MCC will publish one current and valid root certificate that is used to authenticate (sign) each identity provider certificate.

PKI1.6. The MCC will provide a list of identity providers, links to obtain their root certificates, security levels, and signatures of certificates signed with the given root certificate. Including a revocation list.

The MCC board will manage this root certificate and detail guidelines and rules for its operation. These rules should follow best practice. The management can be delegated by the board to a specific host member.

Note, that this does not break with the above claim that the MCC will not work as a root CA. This is intended to only give a basic trust that the MCP instances are endorsed by the MCC and, to the best of MCCs knowledge, are operating within rules and guidelines as defined by the MCC. As stated earlier, full trust can only be established between each organisation and if deeper trust is needed, we must refer to other PKI systems, which will be discussed at a future point in time.

2.1 Application programming interface and implementation

The details of the implementation can be found in [1]. This give the API for MCP Root Certificates Storage Service. It also provides coding examples of how the API can be used.

2.2 Security Requirements and Profiles

Security requirements to be defined will fall into the following categories:

1. Requirements on vetting. This can be specified similarly to classes such as EV (extended validation).
2. Requirements on certificate revocation
3. Requirements on the validity period of certificates
4. Requirements on security of keys and origin of signing - CA side (including requirements on HSMs)
5. Requirements on security of keys and origin of signing - MCP entity side (including requirements on HSMs)

The requirements will be dependent on the currently emerging profiles:

1. MCP entities generate their ID key pair themselves and in own responsibility and provide this to the responsible CA for certification.
2. The CA (perhaps together with a manufacturer) provisions the initial ID key pair and certificate securely within HSMs (for/within endpoints) to be distributed to the MCP entities.

3 CRYPTOGRAPHIC REQUIREMENTS

The cryptographic mechanism approved for ID digital signatures is the Elliptic Curve Digital Signature Algorithm (ECDSA) [FIPS 186-3] with the appropriate hash algorithm from the SHA-2 family [FIPS 180-3]. The approved elliptic curve domain parameters are specified by reference to standardized curves. Currently the following combinations are approved:

ECDSA Key Size (bits)	Hash Algorithm	Elliptic Curve Domain Parameters
384	SHA-384	P-384 [FIPS 186-3] (= secp384r1)
256	SHA-256	P-256 [FIPS 186-3] (= secp256r1)

Note: The reference implementation currently uses SHA-256 for 384 bits key size. This will be changed.

Future extensions:

- Requirements on key pair generation and checks for key pair validity will be given by reference to standards. Also we will check whether there are relevant recommendations in the last version [FIPS 186-5].
- Currently the only approved curve parameters are the NIST recommended curves. It will be checked whether this needs to be extended with regards to cryptographic recommendations of member states' security agencies (e.g. BSI and brainpool curves). Also, if a curve is found to be weak in the future it will be good to have an alternative curve per key size already approved.
- We will also consider matters of crypto agility.

4 CERTIFICATE FORMAT

We now specify the format of the MCP ID certificates. The format is based on the X.509 standard[1]. The standard information present in an X.509 certificate includes:

- **Version** – which X.509 version applies to the certificate (which indicates what data the certificate must include)
  - **Serial number** – A unique assigned serial number that distinguishes it from other certificates
  - **Algorithm information** – the algorithm used to sign the certificate
  - **Issuer distinguished name** – the name of the entity issuing the certificate (MCP)
  - **Validity period of the certificate** – start/end date and time
  - **Subject distinguished name** – the name of the identity the certificate is issued to
  - **Subject public key information** – the public key associated with the identity

The Subject distinguished name field consists of the following items:

Field	**User**	**Vessel**	**Device**	**Service**	**MMS**	**Organization**
CN (CommonName)	Full name	Vessel name	Device name	Service Domain Name	MMS name	Organization Name
O (Organization)	Organization MRN					
OU (Organizational Unit)	"user"	"vessel"	"device"	"service"	"mms"	"organization"
E (Email)	User email					Organization email
C (Country)	Organization country code					
UID	Entity MRN					Organization MRN

Example: The following gives an example of the Subject distinguished name field for a vessel with identity provider idp1:

C=DK, O=urn:mrn:mcp:org:idp1:dma, OU=vessel, CN=JENS SØRENSEN, UID=urn:mrn:mcp:vessel:idp1:dma:jens-soerensen

In addition to the information stored in the standard X.509 attributes listed above, the X509v3 extension SubjectAlternativeName (SAN) extension is used to store extra information. There already exists some predefined fields for the SAN extension, but they do not match the need we have for maritime related fields. Therefore the “otherName” field is used, which allows for using an Object Identifier (OID) to define custom fields. The OIDs currently used are not registered at ITU, but are randomly generated using a tool provided by ITU (see http://www.itu.int/en/ITU-T/asn1/Pages/UUID/uuids.aspx). See the table below for the fields defined, the OIDs of the fields and which kind of entities that use the fields.

Field	**OID**	**Used by**

[1] https://www.itu.int/rec/T-REC-X.509

Flagstate	2.25.323100633285601570573910217875371967771	Vessels, Services
Callsign	2.25.208070283325144527098121348946972755227	Vessels, Services
IMO number	2.25.291283622413876360871493815653100799259	Vessels, Services
MMSI number	2.25.328433707816814908768060331477217690907	Vessels, Services
AIS shiptype	2.25.107857171638679641902842130101018412315	Vessels, Services
Port of register	2.25.285632790821948647314354670918887798603	Vessels, Services
Ship MRN	2.25.268095117363717005222833833642941669792	Services
MRN	2.25.271477598449775373676560215839310464283	Vessels, Users, Devices, Services, MMS
Permissions	2.25.174437629172304915481663724171734402331	Vessels, Users, Devices, Services, MMS
Subsidiary MRN	2.25.133833610339604538603087183843785923701	Vessels, Users, Devices, Services, MMS
Home MMS URL	2.25.171344478791913547554566856023141401757	Vessels, Users, Devices, Services, MMS
URL	2.25.245076023612240385163414144226581328607	MMS

Encoding of string values in certificates must follow the specifications defined in RFC 5280[2], and where possible it is highly recommended to use UTF-8.

5 SERVICE CERTIFICATES

Several maritime services come with requirements concerning cryptography and/or certificate formats that might make it impossible to employ MCP ID credentials directly. For example, if an identity provider issues certificates for ECDSA with 384 bits key size this will not meet the real-time requirements and low bandwidth conditions of AIS and VDES [TODO: ref Gareth's paper]. While the service must then provide its own CA the service CA can automatically issue its service certificates based on MCP ID credentials. We provide an example of how this can be done based on the concept of *certificate signing requests (CSRs)*, also known as *certification requests*. The most common format for CSRs is defined by the PKCS#10 standard [RFC 2986].

Example: In the following we show the steps carried out by an MCP entity to request a service certificate, and the steps performed by the service CA to issue the certificate respectively. The example follows the implementation of the Haptik CA from the project Haptik[3]. This functionality will also be embedded in a web service and secured by the MCP OpenID Connect/OAuth 2.0 framework.

The MCP entity ...

1. generates a fresh key pair for use with the service
2. builds a X.500 name for use in the service certificate
3. builds a corresponding PKCS#10 CSR
4. signs the CSR with their private MCP ID key
5. sends the CSR together with their MCP ID certificate to the service CA.

On receipt the service CA ...

[2] https://tools.ietf.org/html/rfc5280
[3] https://haptik.io

1. checks whether the CSR is valid
2. builds a X.509v3 certificate according to the CSR and additional information provided by the CA such as issuer, serial number and validity period
3. signs this with their CA private key
4. sends the new certificate to the requesting MCP party.

Note: This pattern is also applicable when the MCP ID keys are mainly used as enrolment keys to obtain shorter lived "working keys".

6 INTEGRATION OF OTHER PKI SYSTEMS

In the spirit of decentrality the PKI shall remain open for PKI systems other than X.509, and also agile for update of certificate formats. Care will be taken to accommodate the necessary flexibility when defining usage of certificates. More to this point is provided by example of the P3KI approach.

REFERENCES

[1] MCP Root Certificates Storage Service; Oliver Steensen-Bech Haagh