Towards Minimal Certificates for Federated Space Public Key Infrastructure

Alin-Petru Roșu
(Delft University of Technology)

Oana-Alexandra Graur
(European Space Agency)

Security for Space Systems (3S) 2025
ESTEC, The Netherlands
November 05, 2025

Agenda

01
Introduction
Context
Problem Statement
Objectives

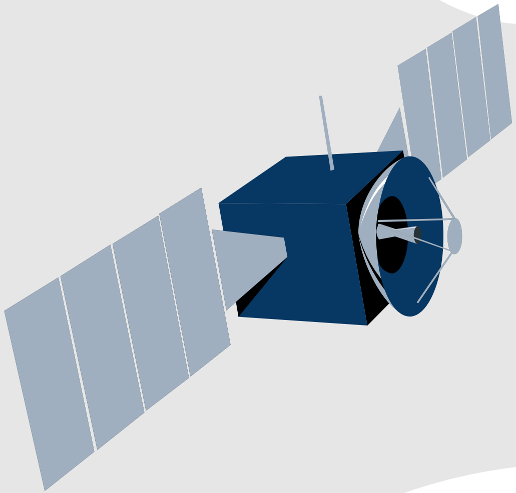
02
Preliminaries
Background

03
Methodology
PQ Certificates
Federal Profiles
C509 Tooling

04
Results
X.509 vs. C509
Comparative Analysis

05
Discussion
Implications

06
Conclusion
Future Work
Summary



01 Introduction

Context

Problem Statement

Objectives

Context

- Space missions increasingly require **international collaboration** (e.g., Artemis)
- **Interoperability** becomes critical and harder to achieve
- ECSS & CCSDS key management limited to **symmetric cryptography** which **lacks scalability**
- **PKI** deployment in space is challenging; **federated PKI**, even more
- CCSDS **Intergovernmental Certification Authority** (IGCA) aims to enable federated, trusted cooperation

Problem Statement

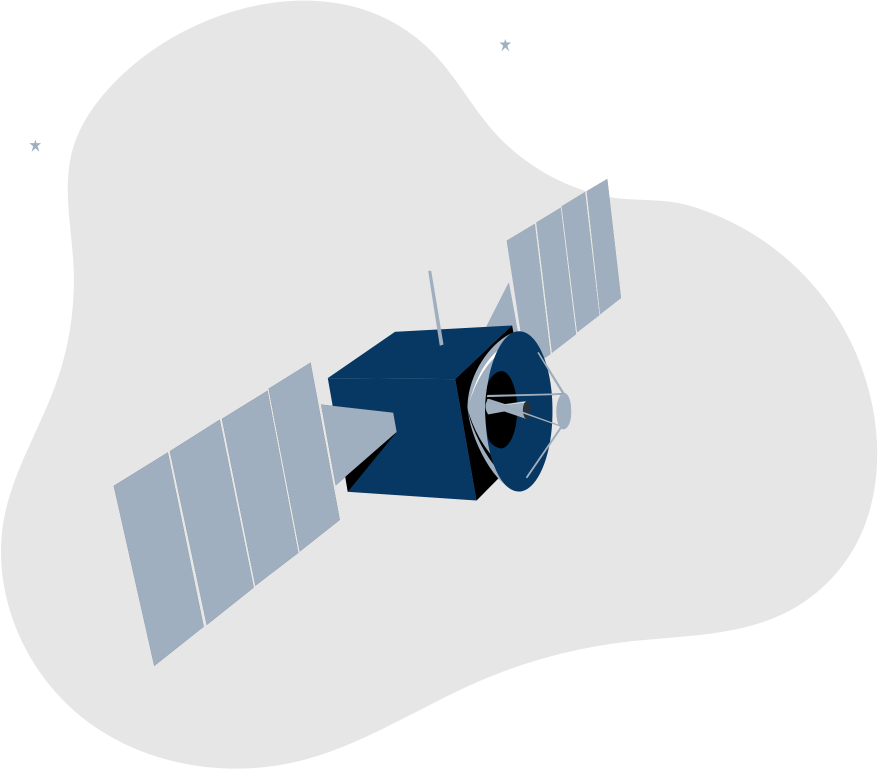
- Space standards adopt the X.509 **Internet Profile** for **interoperability**
- X.509 certificates are verbose and complex (extension mechanism)
- "**200 different extensions** exist in real life" – *[1]*
- *"**11M X.509 certificates** (…) **21.5% are syntactically incorrect**" – [2]*
- Improper parser implementation are linked to multiple attacks– *[3]*
- Post-Quantum (PQ) Cryptography complicates the matter

Ebalard et al., ANSSI, Journey to a RTE-free X.509 parser [1]
Barenghi et al., Systematic Parsing Of X.509: Eradication Security Issues with a Parse Tree [2]
Shi et al., X.509DoS: Exploiting and Detecting Denial-of-Service Vulnerabilities in Cryptographic Libraries using Crafted X.509 Certificates [3]

Objectives

"What is the minimal, interoperable certificate profile capable of bridging traditional and PQ cryptography while supporting cross-domain space federation?"

1) Analyse PQ certificate formats for federated space PKI
2) Review extension configurations from terrestrial federations to design minimal profiles for space
3) Compare X.509 and C509 for space deployment suitability



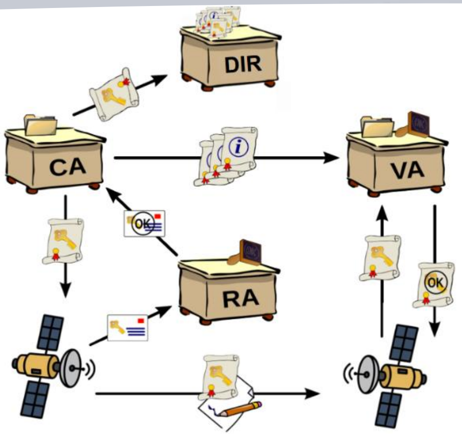
02

Preliminaries

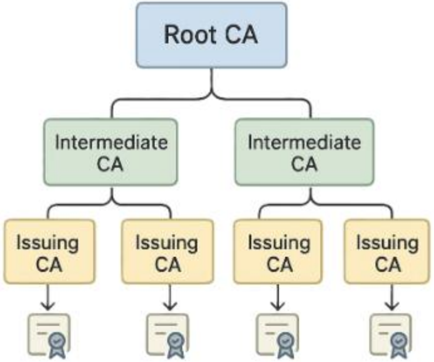
Background
Foundational Work

Public Key Infrastructure

- **PKI:** Trust framework using digital certificates to bind identities to public keys
- **Core roles:** Certification Authority (CA), Registration Authority (RA), Validation Authority (VA), Certificate Repository (DIR), End Entities (EE)
- **Single-tier PKI:** One CA, simpler but less scalable (first image)
- **Hierarchical PKI:** Root CA → Intermediate CAs → Issuing CAs; enables scalable segmented trust (second image)



Single CA PKI. Adapted from [1] under CC BY-SA 3.0

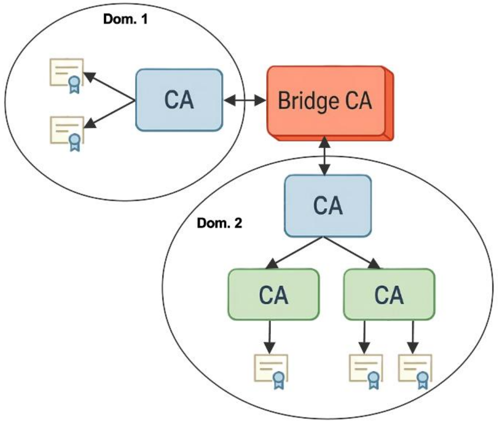


Layered PKI Architecture

https://commons.wikimedia.org/wiki/File:Public-Key-Infrastructure.svg [1]

Federated PKI

- **Federated PKI** enables trust across independent domains without a shared root
- **Bridge CA model:** one central CA cross-certified by all domains → centralised but scalable (see image)
- **IGCA (Intergovernmental Certification Authority)**: bridge-CA-based PKI for space missions, balancing interoperability and organisational autonomy



Bridge PKI Model

X.509 Certificates

- **X.509 standard** (by **ITU-T**) defines public-key certificate syntax using ASN.1 and encoded with **DER**
- **X.509 Internet profile** (by **IETF**) restricts features and defines validation rules for Internet interoperability
- **Main fields:** version, serial number, issuer, subject, public key, validity, signature, extensions
- **Extensions:** additional information (critical / non-critical)

```
Data:
① Version: 3 (0x2)
② Serial Number:
      2b:9b:61:9f:01:76:3c:6f:71:2a:40:cc:49:a9:db:8a:8e:2b:39:8c
③ Signature Algorithm: ecdsa-with-SHA256
④ Issuer: CN=root
⑤ Validity
      Not Before: Dec 30 14:16:52 2024 GMT
      Not After  : Dec 30 14:16:52 2025 GMT
⑥ Subject: CN=x509dos.com, emailAddress=test@x509dos.com
⑦ Subject Public Key Info:
   ⑧ Public Key Algorithm: id-ecPublicKey
         Public-Key: (256 bit)
      ⑨ pub:
            04:f7:1c:6e:dc:c9:ad:9a:85:c8:2f:ca:06: 53: e1:
            c7:59:64:30:2a:a8:72:a8:94:69:f6:7a:72:40: 9f:
            eb:d3:30:72:76:2b:92:b0:43:f9:a2:53:ce:a1:d3:
            f5:9a:d0:f8:d1:39:29:27:11:29:6f:af:b5: a4: a6:
            6f:a9:00:5d:98
      ⑩ ASN1 OID: secp256k1
⑪ X509v3 extensions:
   ⑫ X509v3 Subject Alternative Name:
         DNS:a.x509dos.com, DNS:b.x509dos.com
   ⑬ X509v3 Name Constraints: critical
         Permitted:
          DNS:permitted@x509dos.com
         Excluded:
          DNS:excluded@x509dos.com
   ⑭ X509v3 Certificate Policies:
         Policy: 1.2.3.4
         Policy: 1.2.3.5
   ⑮ X509v3 Policy Mappings:
         1.2.3.4:1.2.3.5, 1.2.3.5:1.2.3.4
   ...
```

tbsCertificate

Signature Algorithm: ecdsa-with-SHA256

```
30:45:02:21:00:f8:07:50:9e:00:70:11:21:c9:d5:68:82:16:
c6:3e:00:43:46:4b:a0:3a:ba:62:8b:a8:97:5d:20:16:85:12:
55:02:20:52:52:86:0f:6d:ac:45:24:2a:c5:b7:ac:a3:7d: bb:
8a:40:5d:97:9c:86:d8:42:c8:c9:74:5e:78:13:ae:f1:1d
```

Signature Value

X.509DoS: Exploiting and Detecting Denial-of-Service Vulnerabilities in Cryptographic Libraries using Crafted X.509 Certificates

C509 Certificates

C509: a subset of X.509 encoded with CBOR; optimised for constrained devices

"CBOR encoding can reduce the size of (...) certificates with over 50% while also significantly reducing memory and code size compared to ASN.1"

— *CBOR Encoded X.509 Certificates (C509 Certificates),* COSE working group

Natively Signed

- **Signature:** computed on the CBOR-encoded TBS structure
- **Encoding:** CBOR only
- **Compatibility:** backwards incompatible to X.509-only clients

Re-encoded

- **Signature:** computed on the DER-encoded TBS structure
- **Encoding:** parsing CBOR serialising CBOR/DER
- **Compatibility:** backwards compatible (via re-encoding)

C509 vs. X.509 Signature

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 128269 (0x1f50d)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: CN=RFC test CA
        Validity
            Not Before: Jan  1 00:00:00 2023 GMT
            Not After : Jan  1 00:00:00 2026 GMT
        Subject: CN=01-23-45-FF-FE-67-89-AB
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:b1:21:6a:b9:6e:5b:3b:33:40:f5:bd:f0:2e:69:
                    3f:16:21:3a:04:52:5e:d4:44:50:b1:01:9c:2d:fd:
                    38:38:ab:ac:4e:14:d8:6c:09:83:ed:5e:9e:ef:24:
                    48:c6:86:1c:c4:06:54:71:77:e6:02:60:30:d0:51:
                    f7:79:2a:c2:06
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage:
                Digital Signature
    Signature Algorithm: ecdsa-with-SHA256
        30:46:02:21:00:d4:32:0b:1d:68:49:e3:09:21:9d:30:03:7e:
        13:81:66:f2:50:82:47:dd:da:e7:6c:ce:ea:55:05:3c:10:8e:
        90:02:21:00:d5:51:f6:d6:01:06:f1:ab:b4:84:cf:be:62:56:
        c1:78:e4:ac:33:14:ea:19:19:1e:8b:60:7d:a5:ae:3b:da:16
```

X.509

```
2,
h'01f50d',
0,
"RFC test CA",
1672531200,
1767225600,
48(h'0123456789AB'),
1,
h'02B1216AB96E5B3B3340F5BDF02E693F16213A04525ED44450
  B1019C2DFD3838AB',
1,
h'EB0D472731F689BC00F5880B12C68B3F9FD38B23FADFCA2095
  0F3F241B60A202579CAC28CD3B7494D5FA5D8BBAB4600357E5
  50AB9FA9A65D9BA2B3B82E668CC6'
```

Natively Signed

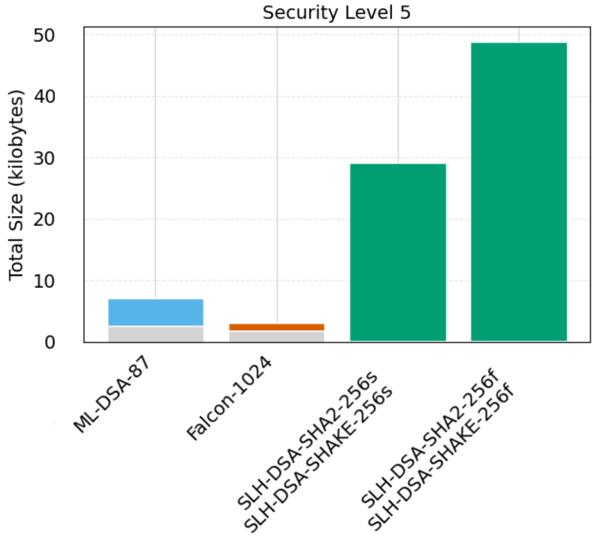
≠

```
3,                    / version and certificate type /
h'01f50d',            / serialNumber /
0,                    / signatureAlgorithm /
"RFC test CA",        / issuer /
1672531200,           / notBefore /
1767225600,           / notAfter /
48(h'0123456789AB'),  / subject, EUI-64 /
1,                    / subjectPublicKeyAlgorithm /
h'FEB1216AB96E5B3B3340F5BDF02E693F16213A04525ED44450
  B1019C2DFD3838AB',
1,                    / single extension:
                        non-critical keyUsage
                        digitalSignature /
h'D4320B1D6849E309219D30037E138166F2508247DDDAE76CCE
  EA55053C108E90D551F6D60106F1ABB484CFBE6256C178E4AC
  3314EA19191E8B607DA5AE3BDA16'
```

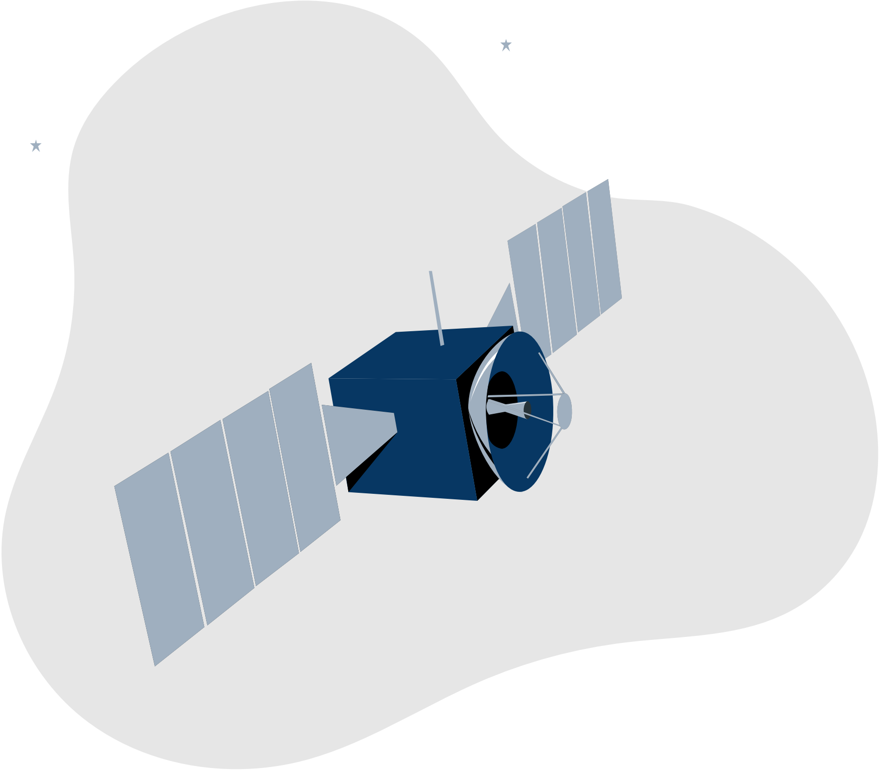
Re-encoded

Post-Quantum Cryptography

- **PQC standardisation:** NIST selected ML-DSA, SLH-DSA; Falcon (FN-DSA),ML-KEM, HQC
- **Deployment challenges:** large key/signature sizes, intensive operations
- **Hybridisation debate:**
  - **Europe (BSI, ANSSI, EU):** hybrid recommended
  - **USA (NSA):** pure PQ allowed
- **Interoperability need:** flexible certificate profiles supporting both hybrid and standalone PQ deployments



Level 5 Signature and Public Key Sizes for NIST Standardized Signature Schemes



03

Methodology

PQ Certificates
Federal Profiles
C509 Tooling

Post-Quantum Certificates

Post-Quantum Certificate Formats

Pure

one post-quantum component

Hybrid Certificate Formats

Catalyst

alternative public key and signature extensions

Composite

unique OID for each hybrid combination

One certificate chain

Chameleon

Delta (certificate differences) extension

Bound

related (linked) certificate extension

Separate certificate chains

NIST SP 1800-38C, Migration to Post-Quantum Cryptography Quantum Readiness: Testing Draft Standards

Post-Quantum Certificate Formats

Pure

- A single post-quantum component; same format as X.509
- **Backwards compatibility:** Incompatible with **legacy systems**, (must recognise the new OIDs)
- **Security:** Based on the security of the post-quantum algorithm
- **Use case:** Used in quantum-safe PKIs; the transition end goal

```
Certificate
├── TBSCertificate
│   ├── version
│   ├── serial number
│   ├── signature algorithm : PQC SIG OID
│   ├── issuer
│   ├── validity
│   ├── subject
│   ├── subject public key info
│   │   ├── algorithm : PQC PK OID
│   │   └── subject public key : PQC PK value
│   └── extensions
├── signature algorithm : PQC SIG OID
└── signature value : PQC SIG value
```

Post-Quantum Certificate Formats

Hybrid Catalyst

- PQ component stored in alternative algorithm extensions.
- **Status:** ITU-T X.509 standard; not adopted by IETF

("ISARA Dedicates Four Hybrid Certificate Patents to the Public")

- **Backwards compatibility:** Extensions marked as non-critical
- **Security:** Either traditional or PQ component (not both)
- **Use case:** Gradual transition to quantum-safe PKI (simplified certificate management)
- **Disadvantage:** Potential bandwidth waste (PQ component is not used recognized)

```
Certificate
├── TBSCertificate
│   ├── version
│   ├── serial number
│   ├── signature algorithm : traditional SIG OID
│   ├── issuer
│   ├── validity
│   ├── subject
│   ├── subject public key info
│   │   ├── algorithm : traditional PK OID
│   │   └── subject public key : traditional PK value
│   └── extensions
│       ├── subjectAltPublicKeyInfo : PQ PK
│       ├── altSignatureAlgorithm : PQ SIG OID
│       └── altSignatureValue : PQ SIG value
├── signature algorithm : traditional SIG OID
└── signature value : traditional SIG value
```

ITU-T, X.509, pg. 58

Multiple Public-Key Algorithm X.509 Certificates

Post-Quantum Certificate Formats

Hybrid Composite

- PQ and traditional keys/signatures are concatenated; each combination has a unique OID ; same format as X.509
- **Status:** IETF **drafts** for ML-KEM and ML-DSA composite OIDs
- **Backwards compatibility:** Incompatible with legacy systems
- **Security:** Based on both PQ and traditional components.
- **Use case:** The component algorithms cannot be trusted alone (prohibits separability to increase security).

```
Certificate
├── TBSCertificate
│   ├── version
│   ├── serial number
│   ├── signature algorithm : composite SIG OID
│   ├── issuer
│   ├── validity
│   ├── subject
│   ├── subject public key info
│   │   ├── algorithm : composite PK OID
│   │   └── subject public key : PQC PK value ||
│   │                            traditional PK value
│   └── extensions
├── signature algorithm : composite SIG OID
└── signature value : PQC SIG value || traditional SIG value
```

[Composite ML-KEM for use in X.509 Public Key Infrastructure and CMS]

[Composite ML-DSA for use in X.509 Public Key Infrastructure and CMS]

Post-Quantum Certificate Formats

- **Bound** (RFC9763): The PQ certificate is linked to the traditional one using an extension that contains the hash of the classical certificate. The certificates are independently managed.
- **Chameleon** (Internet draft, **no longer IETF endorsed**): Encode and embed differences between PQ and traditional certificate in an extension in the latter.
- **Backwards compatibility:** compatible and highly flexible; extensions should be non-critical.
- **Security:** Based on the capabilities of each party (**either** traditional or PQ)
- **Disadvantages:**
  - Parallel chains/multiple certificates lead to **complicated lifecycle and management**
  - *“paired certificates could have different validity periods, and the usable overlap is the subscriber’s concern” – (Bound)*
  - Increased bandwidth usage and processing for validation

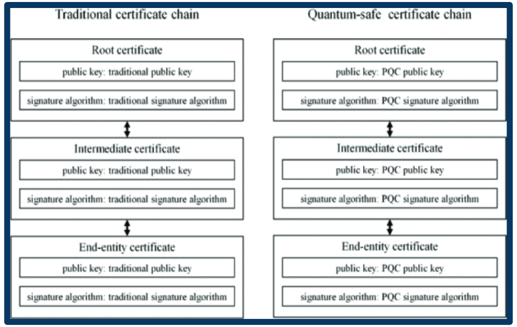
Chameleon

Delta (certificate differences) extension

Bound

related (linked) certificate extension

Separate certificate chains



Parallel certificate chains [1]

Wang et al., Integration of Quantum-Safe Algorithms into X.509v3 Certificates [1]
IETF draft, Related Certificates for Use in Multiple Authentications within a Protocol (Bound)
IETF draft, A Mechanism for Encoding Differences in Paired Certificates (Chameleon)

Space Considerations

- Current standards do not mention any format to be used for transition.
- Bandwidth impact of each format is negligible (see Table).
- Backwards compatibility should come second to security and interoperability (no space PKI deployed)
- Divergent security guidelines on hybridisation
- **Mitigation:** Enforce a single (preferably composite) format for the federation (update Cryptographic Algorithms Blue Book)

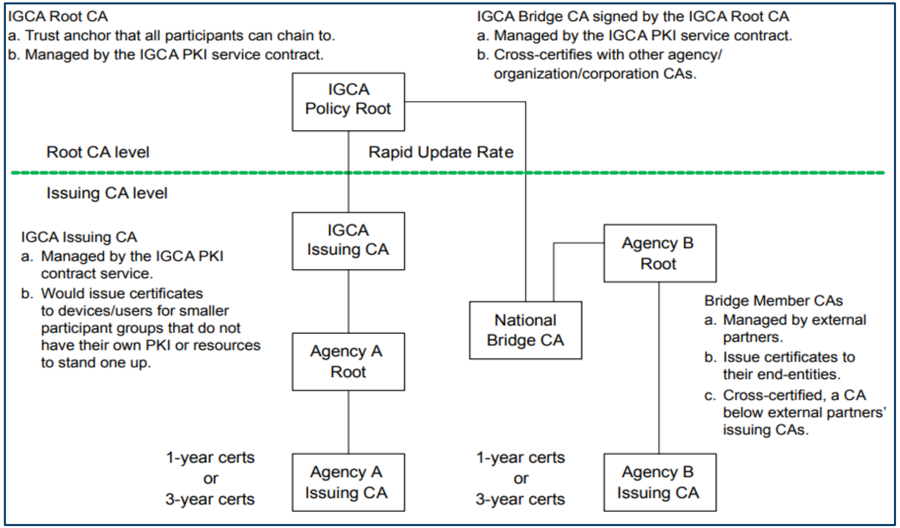
Comparison of certificate sizes (bytes) for pure ML-DSA:44 and hybrid ML-DSA:44 + ECDSA:secp256r1. Body size = total size minus key and signature. Relative Increase = size overhead (bytes) over pure PQ certificate.

Format	Cert. Size (bytes)	Body Size (bytes)	Relative Increase (bytes)
Pure[1]	3894	152	-
Hybrid Composite[1]	4045	158	6
Hybrid with Extensions[2]	4112	229	77
Hybrid Chameleon[3]	4193	310	158
Hybrid Bound (Approx.)[1]	4247	363	211

Federal Profiles

IGCA

- policy and requirements necessary for the IGCA and affiliated CAs to issue and manage trusted certificates
- certificates for **systems, software, spacecraft, instruments, ground stations, relay spacecraft, people**, and other entities

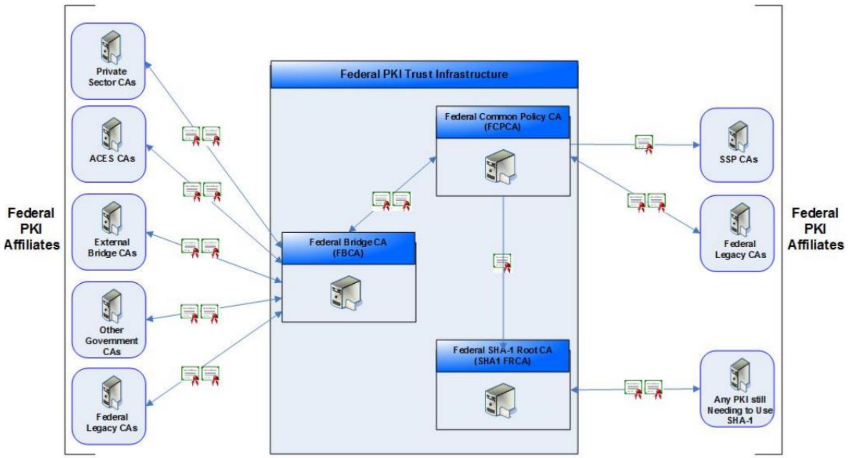


IGCA Architecture [1]

CCSDS 357.1-O-1, Intergovernmental Certification Authority – [1]

FBCA

- U.S. FPKI includes organisations that **work together to provide services for the benefit of the federal government**. – *[1]*
- Personal Identity Verification (**PIV**) and device identity **certificates**
- The **Federal Common Policy Certification Authority** (FCPCA) established trust using the **Federal Bridge Certification Authority** (FBCA) and **defines the policies and standards to be used by the affiliated CAs**



FPKI Architecture [2]

Federal Public Key Infrastructure 101 – [1]
Federal Public Key Infrastructure (FPKI) Concept of Operations (ConOps) – [2]

IGCA

Authentication Credentials Requirements – [1]

Item #	Feature	Status	Support
1	ASN1	M	
2	DER	M	
3	X.509.V3	M	
4	tbsCertificate	M	
5	Version	M	
6	Serial number	M	
7	algorithm identification	M	
8	Issuer Signature	M	
9	Validity from	M	
10	Validity to	M	
11	Subject	M	
12	Subject algorithm identification	M	
13	Subject public Key	M	
14	Issuer Unique ID	O	
15	Subject Unique ID Public Key Info	O	
16	Universal Time Coordinated Time Certificate	M	
17	Generalized Time	M	
18	object identifiers (OID)	O	
19	Policy Mapping	O	
20	Subject Alternative Name	O	
21	Certificate Revocation Lists distribution points	O	
22	signatureAlgorithim	M	
23	signatureValue	M	

- Uses CCSDS Authentication Credentials *[2]*
- Provides minimal guidelines on extensions
- Still an experimental specification

CCSDS 357.0-B-1, CCSDS Authentication Credentials – [1]
CCSDS 357.1-O-1, Intergovernmental Certification Authority – [2]
Federal Bridge Certification Authority (FBCA) X.509 Certificate and CRL Extensions Profile – [3]

FBCA

FBCA Certificate Profiles (subset) – [3]

Worksheet #	Profile	Description
1	Self-Signed CA Certificate	Self-signed certificate issued by CAs primarily for establishing a trust anchor.
2	Self-Issued CA Certificate	Key rollover certificate, sometimes called a link certificate, that is self-issued by a CA but not self-signed.
3	Cross Certificate	Issued by a CA in one PKI domain to a CA in another PKI domain to enable interoperability through certificate policy mapping.
4	Intermediate/Signing CA Certificate	CA certificate issued to a subordinate CA
5	Signature Certificate	Subscriber certificate used to verify signatures.
6	Key Management Certificate	Subscriber certificate used to perform key management operations (e.g., key transport using RSA or Diffie-Hellman key agreement).

- Mandatory and optional extensions of certificates and CRLs
- All fields and extensions listed should be implemented.
- Extensions that are not mandatory or optional should not be included." – [3]

Proposed Federal Profiles for IGCA

Minimal certificate profiles for federated space PKI with RFC5280 standardised extensions
(M – Mandatory, O – Optional, Empty - Disallowed)

Extension	Self-Signed	Self-Issued	Cross	Intermediate	Signature	Key Exchange
Authority Key Identifier		M	M	M	M	M
Subject Key Identifier	M	M	M	M	M	M
Key Usage	M (critical)	M (critical)	M (critical)	M (critical)	M (critical)	M (critical)
Certificate Policies		M	M	M	M	M
Policy Mappings			M			
Subject Alternative Name	O	O	O	O	O	O
Issuer Alternative Name						
Subject Directory Attributes						
Basic Constraints	M (critical)	M (critical)	M (critical)	M (critical)		
Name Constraints				O (critical)		
Policy Constraints			M (critical)	O (critical)		
Extended Key Usage					O	O
CRL Distribution Points		M	M	M	M	M
Inhibit anyPolicy			M (critical)	O (critical)		
Delta CRL Distribution Point						
Authority Information Access		M	M	M	M	M
Subject Information Access	M	M	M	M		

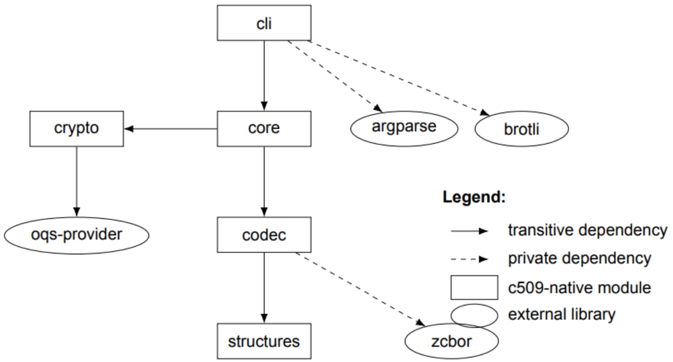
Space Considerations

- Constrained space systems often can't support full RFC 5280 validation or complex X.509 profiles
- Rigid, minimal certificate designs help meet hardware and mission-specific limitations
- Fixed algorithms, fixed-length subject/issuer fields, and minimal extensions reduce parsing complexity
- Mission-specific adaptations are sometimes unavoidable, risking federation-wide inconsistency
- **Conclusion:** Proposed profiles offer a structured foundation for IGCA, but broader alignment and standardisation are needed to address diverse mission requirements

c509-native

A Tool for Natively Signed C509

- Functional requirements:
  - Generate, sign, verify C509 certs, CSRs, CRLs
  - CLI mirroring OpenSSL workflows (subset)
  - Support ML-DSA, ML-KEM, and hybrid (ECDSA, ECDH)
- Non-functional requirements
  - Deterministic CBOR encoding, no dynamic memory
  - Minimal C++: avoid inheritance, dynamic dispatch, exceptions
  - Permissive MIT License, unit-tested core (structures, codecs)
  - Integrated schema-driven generation with zcbor
- Command-line interface (CLI)
  - Commands: genpkey, req, crl, parse
  - OpenSSL-like flags: e.g., -key, -subj, -days, -set_serial



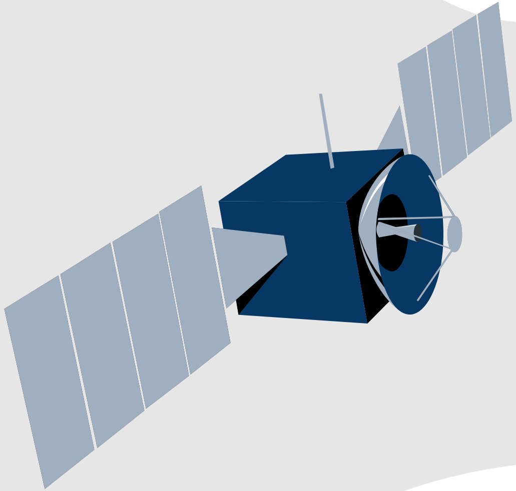
c509-native Design

```
Usage: c509_cli req [--help] [--version] [-in VAR] [-verify] [-new] [-c509] [-CA VAR] [-CAkey
     VAR] [-subj VAR] [-days VAR] [-set_serial VAR] [-addext VAR...] [-key VAR] [-out VAR] [-
    batch] [-compressed]

Optional arguments:
  -h, --help     shows help message and exits
  -v, --version  prints version information and exits
  -in            C509 request input file
  -verify        Verify self-signature on the request
  -new           New request
  -c509          Output an C509 certificate structure instead of a cert request
  -CA            Issuer cert to use for signing a cert, implies -c509
  -CAkey         Issuer private key to use with -CA
  -subj          Specify the subject (Distinguished Name) in OpenSSL format: "/C=XX/ST=State/
      L=City/O=Organization/OU=OrgUnit/CN=CommonName/emailAddress=email@example.com".
  -days          Number of days cert is valid for. Default: 365 days [nargs=0..1] [default:
      365]
  -set_serial    Serial number to use
  -addext        Additional cert extension key=value pair [nargs: 0 or more]
  -key           Key for signing, and to include unless -in given
  -out           Output file
  -batch         Do not ask anything during request generation
  -compressed    Use Brotli compression
```

c509-native *req* Command

https://github.com/rosualinpetru/c509-native



04

Results

X.509 vs. C509
Comparative Analysis

X.509 vs C509 – Field Comparison

- C509 prevents structural encoding overhead

Fields size (in bytes) for X.509 and C509 self-signed certificates (ECDSA P256)

X.509	C509	DER	CBOR
Certificate	Certificate	4	1
tbsCertificate		4	0
version	version	5	1
serialNumber	serialNumber	3	2
signature	signatureAlgorithm	12	1
issuer *(commonName)*	issuer *(commonName)*	28	1
validity		2	0
notBefore	notBefore	17	5
notAfter	notAfter	17	9
subject *(commonName)*	subject *(commonName)*	28	16
subjectPublicKeyInfo		2	0
algorithm	publicKeyAlgorithm	21	1
subjectPublicKey	publicKeyValue	68	67
extensions	extensions	4	1
keyUsage	*keyUsage*	16	3
basicConstraints	*basicConstraints*	17	2
subjectKeyIdentifier	*subjectKeyIdentifier*	31	22
subjectInformationAccess	*subjectInformationAccess*	58	34
signatureAlgorithm		12	0
signatureValue	signatureValue	75	66

X.509 vs C509 – Field Comparison

- C509 prevents structural encoding overhead
- C509 removes duplication
  - A self-signed certificate will mark the issuer as *null*
  - *signatureAlgorithm* is no longer duplicated

Fields size (in bytes) for X.509 and C509 self-signed certificates (ECDSA P256)

X.509	C509	DER	CBOR
Certificate	Certificate	4	1
├── tbsCertificate	│	4	0
│ ├── version	├── version	5	1
│ ├── serialNumber	├── serialNumber	3	2
│ ├── signature	├── signatureAlgorithm	12	1
│ ├── issuer *(commonName)*	├── issuer *(commonName)*	28	1
│ ├── validity	│	2	0
│ │ ├── notBefore	├── notBefore	17	5
│ │ └── notAfter	├── notAfter	17	9
│ ├── subject *(commonName)*	├── subject *(commonName)*	28	16
│ ├── subjectPublicKeyInfo	│	2	0
│ │ ├── algorithm	├── publicKeyAlgorithm	21	1
│ │ └── subjectPublicKey	├── publicKeyValue	68	67
│ └── extensions	├── extensions	4	1
│ ├── *keyUsage*	│ ├── *keyUsage*	16	3
│ ├── *basicConstraints*	│ ├── *basicConstraints*	17	2
│ ├── *subjectKeyIdentifier*	│ ├── *subjectKeyIdentifier*	31	22
│ └── *subjectInformationAccess*	│ └── *subjectInformationAccess*	58	34
├── signatureAlgorithm	│	12	0
└── signatureValue	└── signatureValue	75	66

X.509 vs C509 – Field Comparison

- C509 prevents structural encoding overhead
- C509 removes duplication
  - A self-signed certificate will mark the issuer as *null*
  - *signatureAlgorithm* is no longer duplicated
- C509 optimises extension, signature and public key encoding
  - e.g., point compression for ECC

Fields size (in bytes) for X.509 and C509 self-signed certificates (ECDSA P256)

X.509	C509	DER	CBOR
Certificate	Certificate	4	1
tbsCertificate		4	0
version	version	5	1
serialNumber	serialNumber	3	2
signature	signatureAlgorithm	12	1
issuer *(commonName)*	issuer *(commonName)*	28	1
validity		2	0
notBefore	notBefore	17	5
notAfter	notAfter	17	9
subject *(commonName)*	subject *(commonName)*	28	16
subjectPublicKeyInfo		2	0
algorithm	publicKeyAlgorithm	21	1
subjectPublicKey	publicKeyValue	68	67
extensions	extensions	4	1
keyUsage	*keyUsage*	16	3
basicConstraints	*basicConstraints*	17	2
subjectKeyIdentifier	*subjectKeyIdentifier*	31	22
subjectInformationAccess	*subjectInformationAccess*	58	34
signatureAlgorithm		12	0
signatureValue	signatureValue	75	66

X.509 vs C509 – Field Comparison

- C509 prevents structural encoding overhead
- C509 removes duplication
  - A self-signed certificate will mark the issuer as *null*
  - *signatureAlgorithm* is no longer duplicated
- C509 optimises extension, signature and public key encoding
  - e.g., point compression for ECC
- C509 defines registries for extensions, attributes and policies to replace verbose OIDs with one integer
  - ➢ **C509 saves at least 6 bytes / replaced OID**

Fields size (in bytes) for X.509 and C509 self-signed certificates (ECDSA P256)

X.509	C509	DER	CBOR
Certificate	Certificate	4	1
├── tbsCertificate		4	0
│ ├── version	├── version	5	1
│ ├── serialNumber	├── serialNumber	3	2
│ ├── signature	├── signatureAlgorithm	12	1
│ ├── issuer *(commonName)*	├── issuer *(commonName)*	28	1
│ ├── validity		2	0
│ │ ├── notBefore	├── notBefore	17	5
│ │ └── notAfter	├── notAfter	17	9
│ ├── subject *(commonName)*	├── subject *(commonName)*	28	16
│ ├── subjectPublicKeyInfo		2	0
│ │ ├── algorithm	├── publicKeyAlgorithm	21	1
│ │ └── subjectPublicKey	├── publicKeyValue	68	67
│ └── extensions	├── extensions	4	1
│ ├── *keyUsage*	│ ├── *keyUsage*	16	3
│ ├── *basicConstraints*	│ ├── *basicConstraints*	17	2
│ ├── *subjectKeyIdentifier*	│ ├── *subjectKeyIdentifier*	31	22
│ └── *subjectInformationAccess*	│ └── *subjectInformationAccess*	58	34
├── signatureAlgorithm		12	0
└── signatureValue	└── signatureValue	75	66

X.509 vs. C509 – Object Size

Certificate sizes (bytes) and Brotli compression rates (%) for traditional and PQ algorithms

Certificate	Size (bytes)			Compression (%)	
	X.509	*C509*	*Red. (%)*	*X.509*	*C509*
ECDSA/ECDH with secp256r1					
Self-Signed	424	232	45.3	16.7	-1.7
Self-Issued	578	323	44.1	20.9	10.8
Cross	631	358	43.2	16.0	12.3
Intermediate	581	334	42.5	24.6	13.2
Signature	497	290	41.6	14.3	4.5
Key Exchange	491	284	42.1	14.7	1.1
ML-DSA:44					
Self-Signed	4 019	3 855	4.1	1.3	0.0
Self-Issued	4 174	3 946	5.5	3.3	1.1
Cross	4 227	3 981	5.8	3.1	1.3
Intermediate	4 177	3 957	5.3	3.1	1.2
Signature	4 094	3 913	4.4	1.4	0.3
Key Exchange	4 076	3 945	3.2	1.2	0.3

- C509 savings stem from **CBOR encoding, OID removal, and structural optimisations**
- Compression shows **X.509 has more redundancy** than C509

Absolute size reductions (bytes) for pure PQ/hybrid composite end-entity certificates

Signature	Public Key	X.509	C509	Difference
Security Level 1/2				
mldsa44	mldsa44	4 094	3 913	181
mldsa44	mldsa44_ecdsa_p256	4 173	3 980	193
mldsa44	mlkem512	3 576	3 395	181
mldsa44	ecdh_p256_mlkem512	3 647	3 466	181
mldsa44_ecdsa_p256	mldsa44	4 181	3 998	183
mldsa44_ecdsa_p256	mldsa44_ecdsa_p256	4 259	4064	195
mldsa44_ecdsa_p256	mlkem512	3 664	3 479	185
mldsa44_ecdsa_p256	ecdh_p256_mlkem512	3 734	3 550	184
Security Level 3				
mldsa65	mldsa65	5 623	5 442	181
mldsa65	mldsa65_p256	5 702	5 509	193
mldsa65	mlkem768	4 849	4 668	181
mldsa65_ecdsa_p256	mldsa65	5 710	5 528	182
mldsa65_ecdsa_p256	mldsa65_p256	5 790	5 593	197
mldsa65_ecdsa_p256	mlkem768	4 936	4 753	183
Security Level 5				
mldsa87	mldsa87	7 581	7 400	181
mldsa87	mldsa87_ecdsa_p384	7 692	7 499	193
mldsa87	mlkem1024	6 551	6 370	181
mldsa87_ecdsa_p384	mldsa87	7 700	7 517	183
mldsa87_ecdsa_p384	mldsa87_ecdsa_p384	7 811	7 616	195
mldsa87_ecdsa_p384	mlkem1024	6 670	6 487	183

- C509 is inherently **limited by PQ** cryptographic payloads

CRL sizes (bytes) and Brotli compression rates (%) for traditional and PQ algorithms

Revocations	Size (bytes)			Compression (%)	
	DER	*CBOR*	*Red. (%)*	*DER*	*CBOR*
ECDSA/ECDH with secp256r1					
1	183	107	41.5	0.5	-3.7
10	413	197	52.3	26.2	-2.0
100	2 664	1 098	58.8	54.8	6.0
1 000	25 159	10 100	59.9	60.8	10.7
10 000	250 118	100 099	60.0	62.1	12.2
20 000	500 066	200 099	60.0	62.4	12.3
30 000	750 035	300 100	60.0	62.5	12.3
ML-DSA:44					
1	2 538	2 466	2.8	0.6	-0.2
10	2 766	2 556	7.6	3.0	0.0
100	5 017	3 457	31.1	28.7	1.0
1 000	27 512	12 458	54.7	55.6	8.6
10 000	252 471	102 458	59.4	61.5	11.8
20 000	502 419	202 458	59.7	62.1	12.1
30 000	752 388	302 458	59.8	62.3	12.2

- CBOR CRLs cut size vs. DER through **efficient time encoding**
- PQ **signature overhead is minor for large CRLs**

X.509 vs. C509 – Software Complexity

- Logical Lines of Code (LLOC)
  - Measures code size; higher → more storage, maintenance, testing.
- Cyclomatic Complexity (CCN)
  - Counts independent execution paths; higher → harder testing, more bug risk.
- Halstead Volume
  - Captures token-level cognitive load; larger volume → more code and logic.
- Halstead Difficulty
  - Estimates comprehension effort; sensitive to unique vs. total token ratio.
- Function Count
  - Counts declared functions; shows modularity, useful for context.

Corpus

Implementation	Lang.	Profile	Description
`x509-parser`[2]	C	X.509	ANSSI, open-source, custom, runtime-error-free parser-only implementation (no serialisation) including the DER decoding layer and no external dependencies, formally verified using Frama-C and ACSL annotation comments that do not affect the analysis [9].
`ASN1C (generated)`[3]	C	X.509	ASN.1 schema [6] generated parser and serialiser using the commercial Objective Systems `asn1c` compiler generating industry-grade code for BER/DER with the encoding layer delivered as a pre-compiled library.
`c509-native`[4]	(C-like) C++	C509	The custom parser and serialiser proposed in this work, relying on `zcbor` for the encoding layer, optimised for embedded systems (Chapter 5).
`zcbor (generated)`[5]	C	C509	CDDL schema [8] generated parser using the open-source `zcbor` generator producing low-footprint C encoders/decoders for CBOR.

X.509 vs. C509 – Software Complexity

Comparison of certificate and CRL parser(-serializer) implementations

Implementation	Total LLOC	Mean CCN	Total CCN	Total Volume	Median Diff	Q-95 Diff	Func Count
Setting 1: Parser-only including binary encoding layer (Tool 1: ccccc)							
`x509-parser`	8 019	7.80	1 622	243 775.46	28.67	62.28	208
`c509-native`	1 939	2.99	535	66 309.72	7.88	25.14	179
Setting 1: Parser-only including binary encoding layer (Tool 2: rust-code-analysis)							
`x509-parser`	7 432	7.69	1 630	214 955.33	26.39	56.57	212
`c509-native`	1 346	3.34	565	55 192.19	7.50	25.23	169
Setting 2: Parser-only excluding binary encoding layer (Tool 1: ccccc)							
`ASN1C (generated)`	4 611	6.73	1 090	181 210.87	12.23	52.73	162
`zcbor (generated)`	538	9.02	379	47 710.39	12.29	16.36	42
Setting 2: Parser-only excluding binary encoding layer (Tool 2: rust-code-analysis)							
`ASN1C (generated)`	4 421	6.73	1 090	161 474.18	10.80	47.62	162
`zcbor (generated)`	243	10.21	429	35 456.66	10.47	14.31	42
Setting 3: Parser and serializer excluding binary encoding layer (Tool 1: ccccc)							
`ASN1C (generated)`	6 991	7.09	1 752	277 721.04	19.72	48.92	247
`zcbor (generated)`	1 041	7.70	647	89 113.70	11.84	16.36	84
Setting 3: Parser and serializer excluding binary encoding layer (Tool 2: rust-code-analysis)							
`ASN1C (generated)`	6 389	7.09	1 752	247 644.75	17.60	45.33	247
`zcbor (generated)`	441	8.94	751	65 836.81	10.45	14.80	84

X.509 vs. C509 – Software Complexity

esa TU Delft

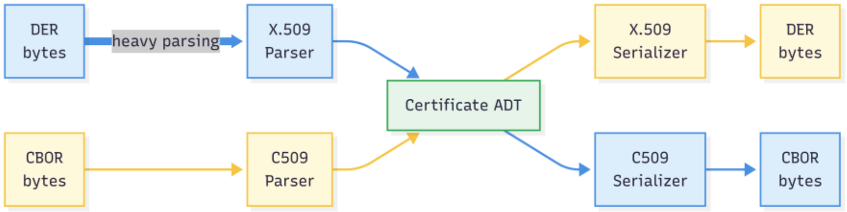
- **Smaller codebase:** C509 parsers show significantly lower LLOC and CCN vs. X.509, improving verifiability and testability.
- **Lower token complexity:** Halstead Volume and Difficulty decrease notably, reducing token-level complexity and enhancing maintainability.

Comparison of certificate and CRL parser(-serializer) implementations

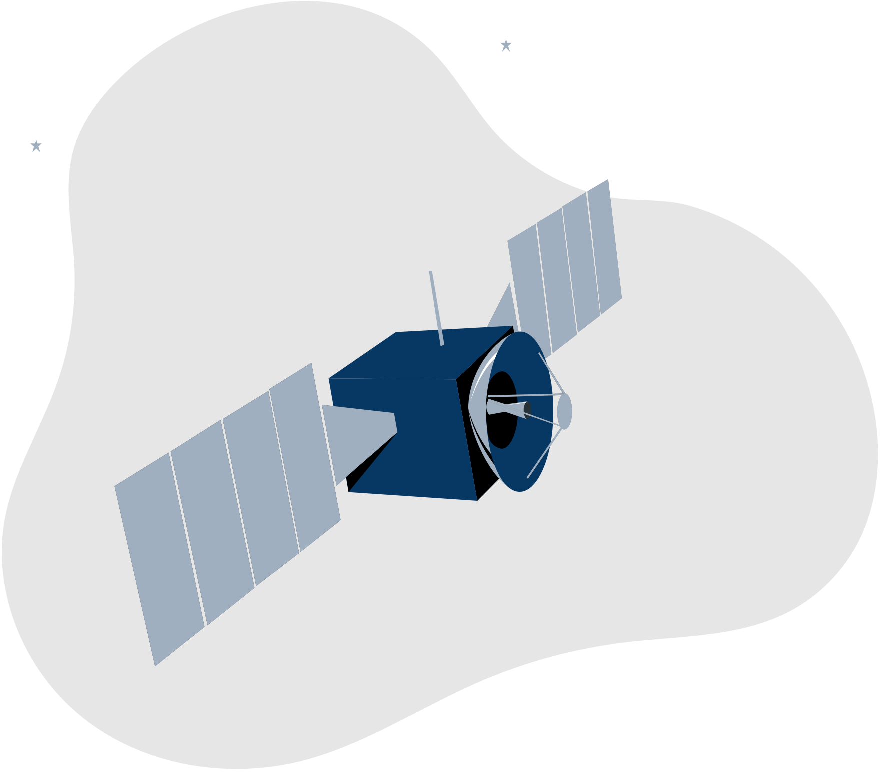
Implementation	Total LLOC	Mean CCN	Total CCN	Total Volume	Median Diff	Q-95 Diff	Func Count
Setting 1: Parser-only including binary encoding layer (Tool 1: ccccc)							
`x509-parser`	8 019	7.80	1 622	243 775.46	28.67	62.28	208
`c509-native`	1 939	2.99	535	66 309.72	7.88	25.14	179
Setting 1: Parser-only including binary encoding layer (Tool 2: rust-code-analysis)							
`x509-parser`	7 432	7.69	1 630	214 955.33	26.39	56.57	212
`c509-native`	1 346	3.34	565	55 192.19	7.50	25.23	169
Setting 2: Parser-only excluding binary encoding layer (Tool 1: ccccc)							
`ASN1C (generated)`	4 611	6.73	1 090	181 210.87	12.23	52.73	162
`zcbor (generated)`	538	9.02	379	47 710.39	12.29	16.36	42
Setting 2: Parser-only excluding binary encoding layer (Tool 2: rust-code-analysis)							
`ASN1C (generated)`	4 421	6.73	1 090	161 474.18	10.80	47.62	162
`zcbor (generated)`	243	10.21	429	35 456.66	10.47	14.31	42
Setting 3: Parser and serializer excluding binary encoding layer (Tool 1: ccccc)							
`ASN1C (generated)`	6 991	7.09	1 752	277 721.04	19.72	48.92	247
`zcbor (generated)`	1 041	7.70	647	89 113.70	11.84	16.36	84
Setting 3: Parser and serializer excluding binary encoding layer (Tool 2: rust-code-analysis)							
`ASN1C (generated)`	6 389	7.09	1 752	247 644.75	17.60	45.33	247
`zcbor (generated)`	441	8.94	751	65 836.81	10.45	14.80	84

C509 Space Considerations

- **Natively Signed:** Direct signatures over CBOR data; removes ASN.1/DER dependency but limits interoperability in federated X.509 environments.
- **Re-encoded:** DER-signed certificates re-encoded to CBOR; maintains X.509 compatibility.
- **DER Parsing Elimination:** Ground gateway parses heavy DER, converts to CBOR (blue); spacecraft parses CBOR, serialises DER for signature verification (yellow).
- **Standardisation limitation:** C509 is still an IETF draft; lacks mature revocation standards.
- **Limited bandwidth gains:** While CRLs shrink notably, PQ certificate saving are limited.
- **Hardware constraints:** Offset-based parsing does not benefit from software simplicity.



X.509 vs. C509 Parsing and Serialising.



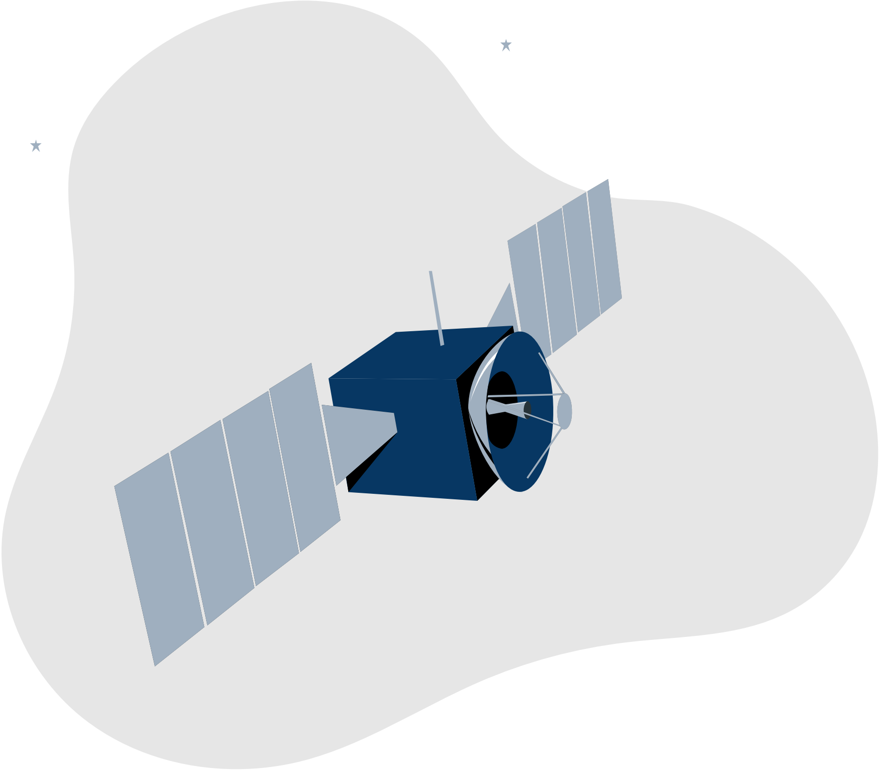
05

Discussion

Implications
Limitations

Discussion

- **Designing a unified profile is challenging**, requiring alignment of cryptography, encoding, profiles and policies.
- **Progress depends on broad multi-stakeholder agreement**; C509's promise is limited by early maturity and low adoption.
- **Current standards** could benefit from **updates and further detailing** to aid the interoperability, implementation and deployment of future federated PKI.
- **Patching X.509 often leads to over-restriction**, delivering little real compatibility.
- Diverging incompatible complicates **COTS integration** and creates technical debt if future interoperability is needed.
- `**c509-native is a prototype** and lacks support for some algorithms



06

Conclusion

Future Work
Summary

Conclusions

- Reviewed and proposed guidelines on PQ formats
- Proposed a preliminary minimal set of extension profiles tailored for CCSDS IGCA
- Developed *c509-native*, an open-source prototype
- C509 cuts size (~60% CRLs yet negligible for PQ certificates) and software complexity (~80% smaller codebase, 2–3x lower cyclomatic complexity) vs. X.509.
- Proposed a potential gateway-based re-encoding C509 deployment; C509 adoption is limited by early maturity
- *Provided insights to help standardisation bodies shape minimal, interoperable space PKI profiles for PQ migration and cross-domain operation.*

Future Work

- **Main bottleneck in space PKIs:** certificate validation
- SCVP (RFC5055) enables **delegated validation**; used in commercial and mobile networks
- Signed/MAC-protection, nonces, and client-specified time references
- **Relayed requests** (e.g., lunar-to-Earth)
- *Further research is needed on latency, security, and compatibility for space*

RFC3379
RFC5055

THANK YOU!

Questions?

Contact