Bundle Protocol Key Distribution (BPKD)

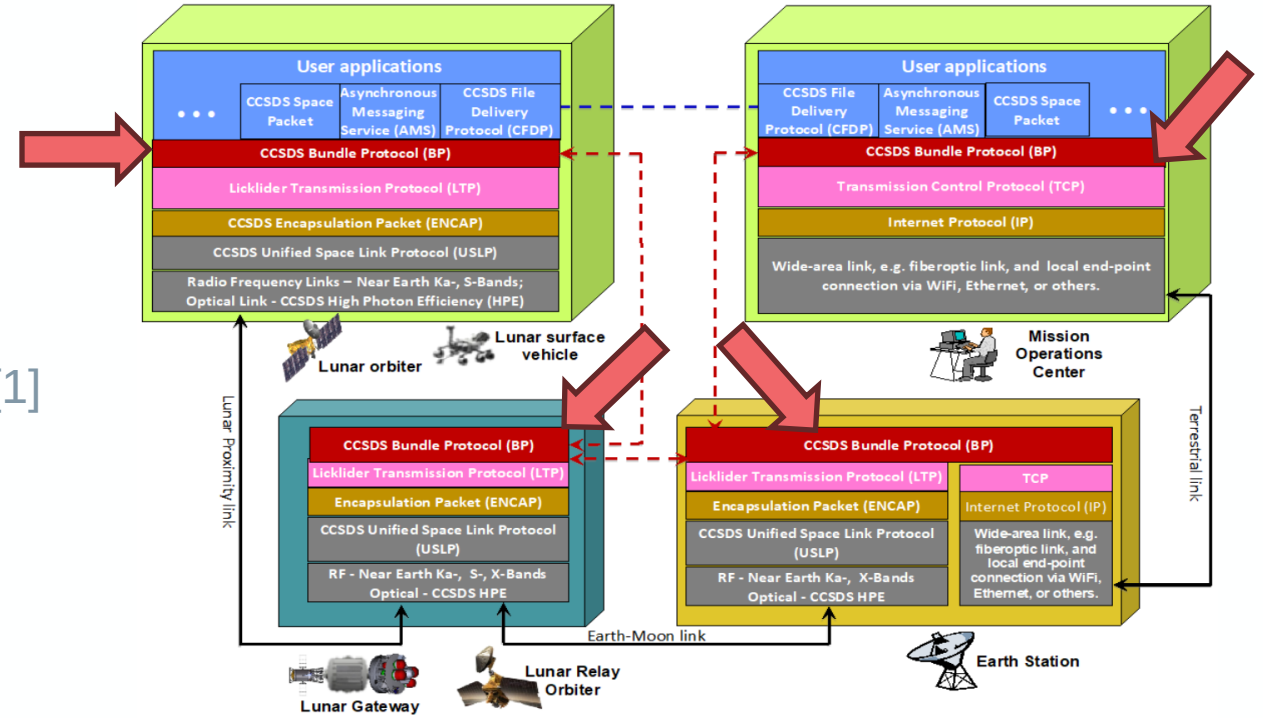
Matthes Wurbs

30/08/2024

Motivation

Bundle Protocol (BP)

- Up-and-coming standard transport layer
- Store, carry and forward architecture
- Referenced in
  - LunaNet Interoperability Specification Document [1]
  - The Future Lunar & Mars Communications Architecture IOAG Reports ([2], [3])
- Specified in RFC 9171 [4]



BP protocol stack, taken from [2] and modified

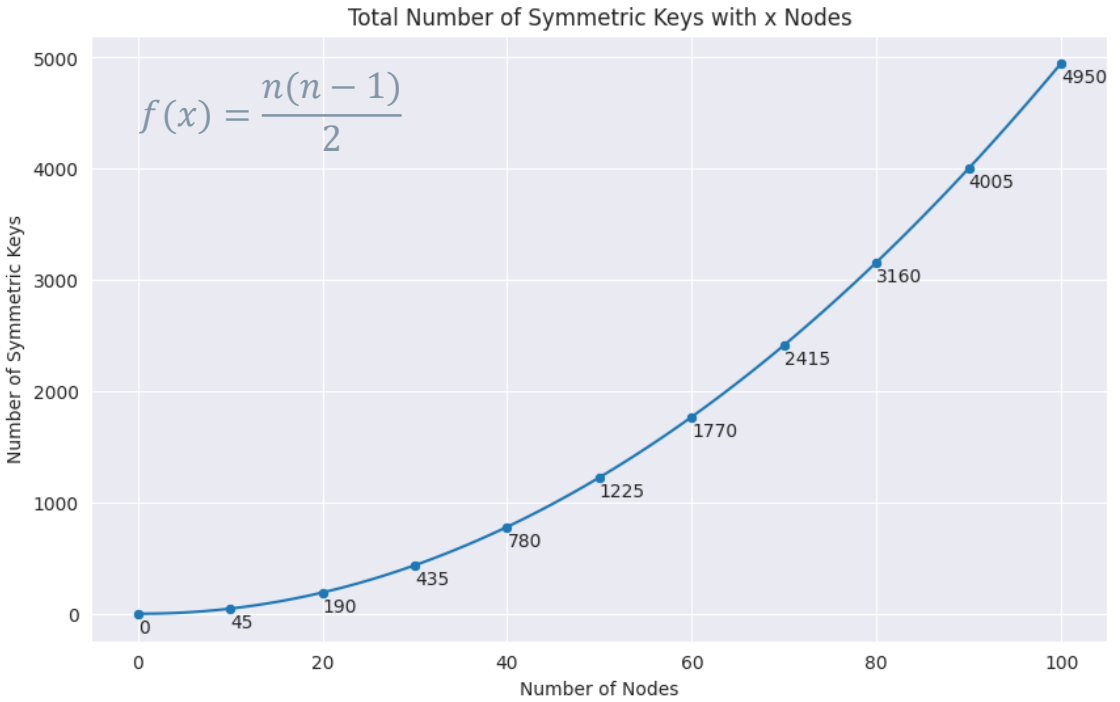
[1] LunaNet Interoperability Specification Document
[2] The Future Lunar Communications Architecture
[3] The Future Mars Communications Architecture
[4] RFC 9171

Motivation

Problem

- Multiple standards have been formalized ([1]-[3])
- [4] and [5] only propose symmetric key management
- No standard yet for public key management/distribution
  → DTN Nodes rely on pre-shared, symmetric keys

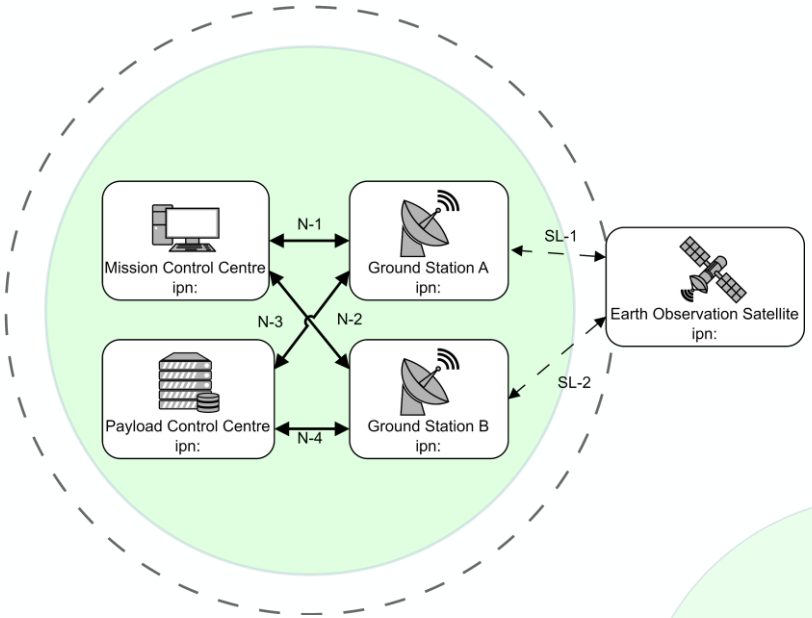
➔ Does not scale
➔ Public key management required sooner or later



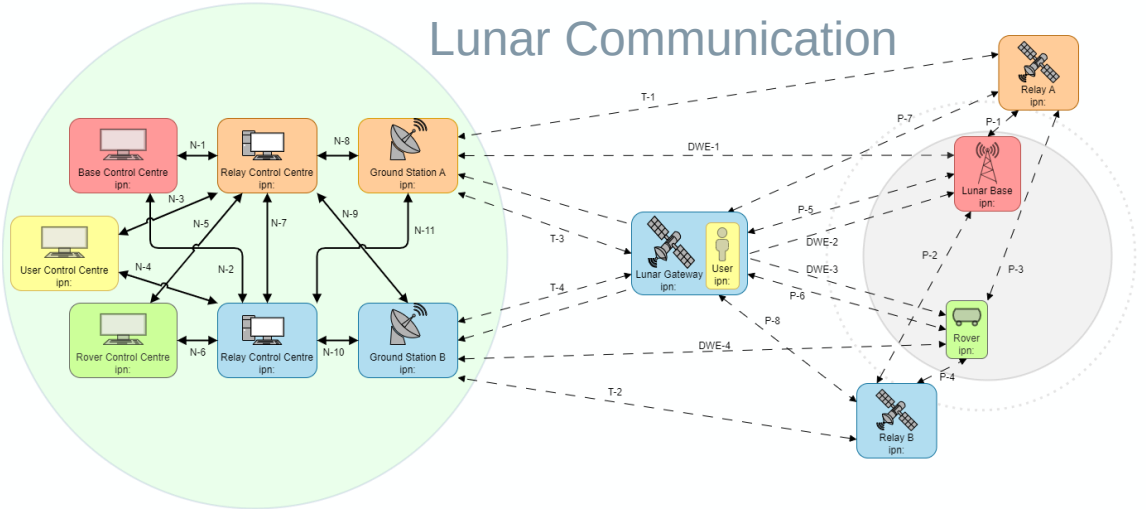
[1] CCSDS Cryptographic Algorithms
[2] RFC 9172: BPSec
[3] RFC 9173: BPSec Default Security Contexts
[4] Space Data Link Security Protocol – Extended Procedures
[5] Symmetric Key Management

Reference Scenarios

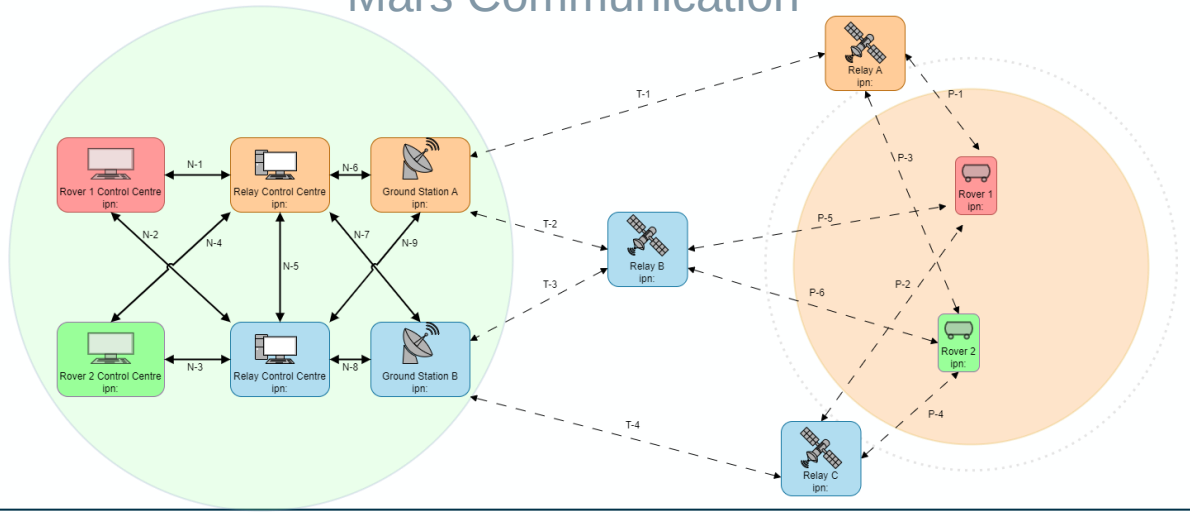
Earth Observation



Lunar Communication



Mars Communication



DTN Challenges

Delays
- Long distances in space can lead to long propagation delays
- Quickly retrieving or validating keys is infeasible

Disruptions
- Communication can be interrupted at any point
- Further increases delays
- Makes handshaking impractical

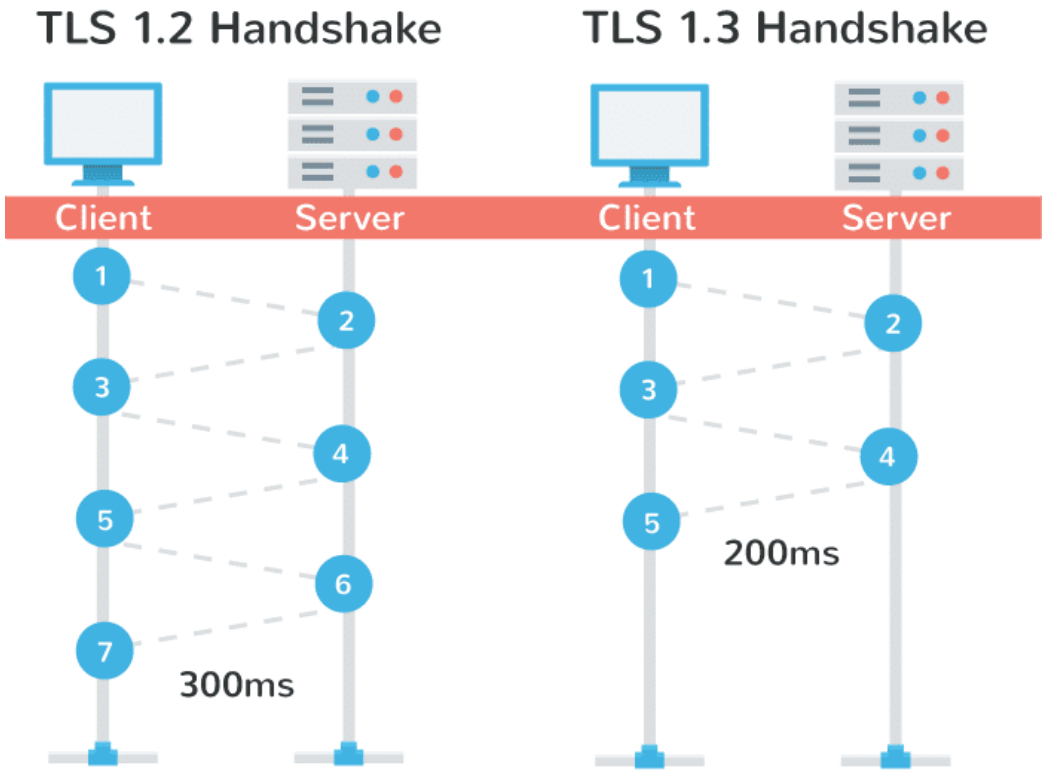
Out-Of-Order Bundles
- Bundles can arrive out of order
- This can lead to faulty states if not accounted for

Round-Trip Times (RTTs)
- Delays and Disruptions can lead to long RTTs
- If round-trips are possible at all
- Makes interactive communication problematic

Lost Bundles
- Bundles can get lost along the way
- Can lead to missing information if not accounted for

Example: TLS Handshake

- TLS needs 3 (TLS 1.2) or 2 (TLS 1.3) round trips for a handshake
- DTN challenges lead to
  - Long handshake times
  - Failed handshakes
    → Retries



TLS 1.2 vs 1.3 handshake [1]

[1] https://kinsta.com/de/blog/tls-1-3/

What is our Focus?

Public key distribution in DTN

- Provisioning, updating and revoking of keys
- Inter-domain key exchange
- **Non-interactive**
  → No handshakes

- Exchanged are **key-derivation keys**
- Focus on **end-entities** (i.e. spacecraft, rovers, etc)
- For short-/mid-term (~50-100 nodes)

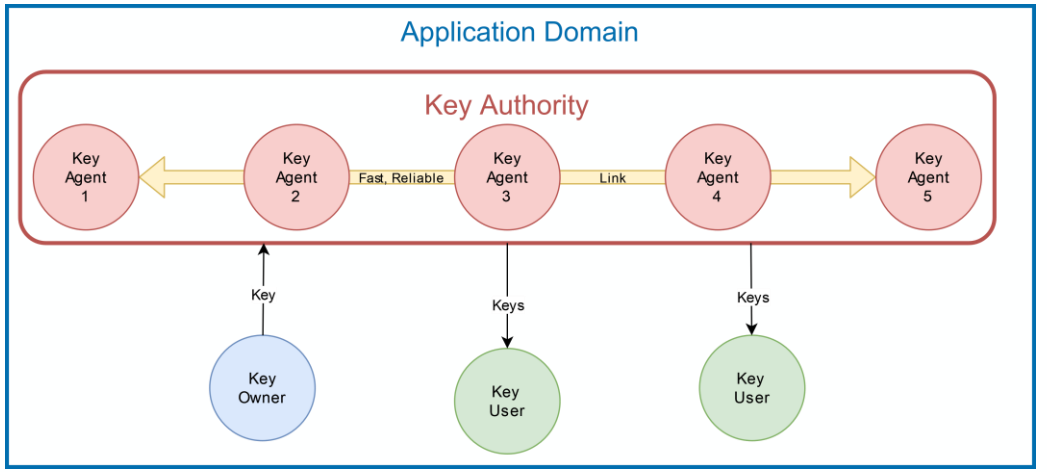
What are we NOT looking into?

- New security contexts

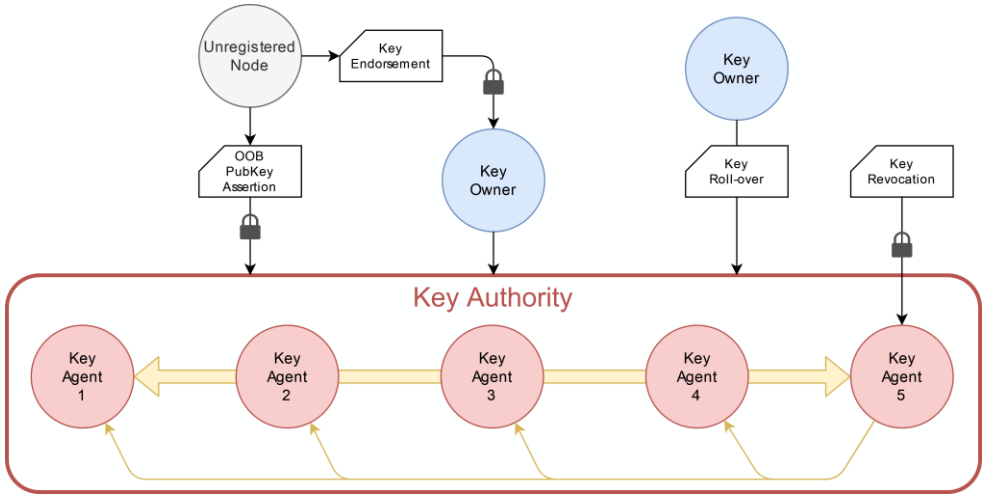
- Setting up a hierarchical PKI
- Provisioning of root certificates
- Bridging between key authorities
  → IGCA Draft

Expired DTKA Draft: Delay-Tolerant Key Administration

- *Key Authority* is comprised of multiple *Key Agents*
  - *Key Agents* require a sub-second OWLT link
- Nodes can use out-of-band assertions or endorsements to join the *Application Domain*
- *Key Owners* can roll over to new keys
- Operators can trigger revocations



- *Key Agents* periodically send *Bulletins* to all *Key Users*
  - Containing updates since last bulletin
  - Applying erasure codes to bulletin
- Missed bulletins can be re-requested



Key Differences to DTKA

Single *Key Authority* per Application Domain
- Easier & faster to deploy
- Terrestrial *Key Authority* would be
  - Easier to harden and secure
  - Able to use ground stations for wide coverage

Public Keys Only On Management Level
- No security contexts for asymmetric keys yet
- We rely on the default security contexts
  - These require symmetric keys → more later

Inter-Domain Communication
- Concept explicitly includes inter-domain functionality

No Erasure Codes
- Only one node sends a snapshot
  - Erasure coding is unnecessary

No Bulletins
- Initially, only a few nodes will be in the network
- Size of a full snapshot is small enough to be distributed directly
- Easier state management

No Multicast
- Has not been specified/standardized yet
- Would require asymmetric keys (and security context) for integrity
- With only few nodes, unicast overhead is acceptable

Terminology Used

Term	Description
PK	Asymmetric Public Key
SK	Asymmetric Secret Key
SessK	Symmetric Session Key
KA	Key Authority Central entity in the system, distributes keys to clients
Client	End-node in a domain, receives and sends its keys to KA (DTKA: Key Owner + Key User)
KD	Key Data = PK + valid from + valid until + ipn-address Could also be X.509 certificates or similar
Snapshot	KD of all clients
AD / Domain	Application Domain = KA + all its clients
OOB	Out-of-band (communication link)
DSC	Default Security Context as defined in RFC 9173

Features

Scalability

- Terrestrial key authority
  - Quick to deploy
  - Can use existing network of GS
- Can be upgraded if the need arises
  - Distributed, off-world key authority
- Bulletins
- Multicast-ready when it has been specified

Revocations

- Triggered manually by an operator
  → More on that later

Default Security Context (RFC 9173) compatibility

- Required symmetric keys derived
- Two-layer architecture allows using public keys while still working with the DSC

Crypto-Algorithm agnostic

- For now, we use ECC on Curve25519
  - Fast, small keys
- When the need arises, algorithms can be exchanged to be PQ-safe (e.g. TripleKEM)
  - Preferably a non-interactive algorithm
- Derived AES sessions keys of DSC PQ-safe

Inter-domain key exchange

- Supports key exchange between application domains for *Clients*
- “Domains” could be space agencies, companies, governments, …

Quick Facts

Asymmetric Layer	Curve25519
Symmetric Layer - KDF	HKDF (RFC 5869 / NIST SP80056Cr2)
BPSEC - Confidentiality	AES-GCM (256 bit)
BPSEC - Integrity	HMAC-SHA2 (384 bit)
Key Data size per node	Depending on factors like endpoint ID and used algorithms. Raw CBOR: ~64 byte X.509 DER: ~256 byte

Prerequisites

Initial State

- Key management client application loaded on node
- Initial trust between *KA* and *client* in same application domain

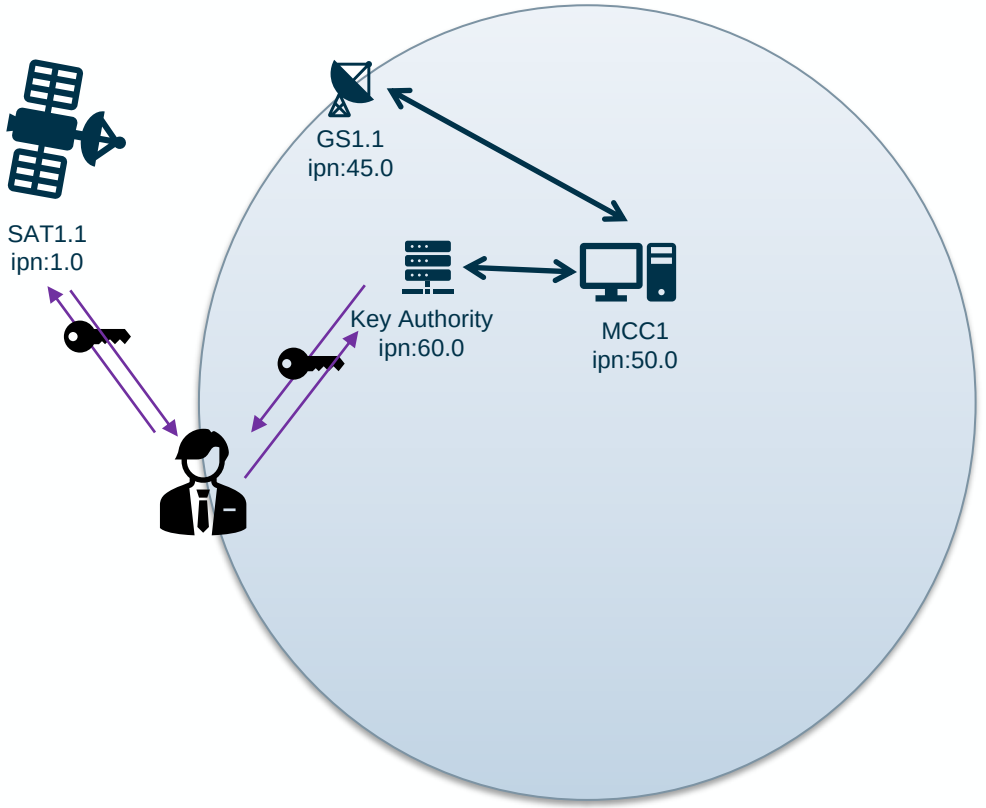
Onboard Functions

- Node needs to be able to generate new key pairs → RNG required
  - Alternatively: Operator sends secret key material via secure OOB channel → Less secure
- Key Derivation Function, possibly with hardware acceleration

Infrastructure

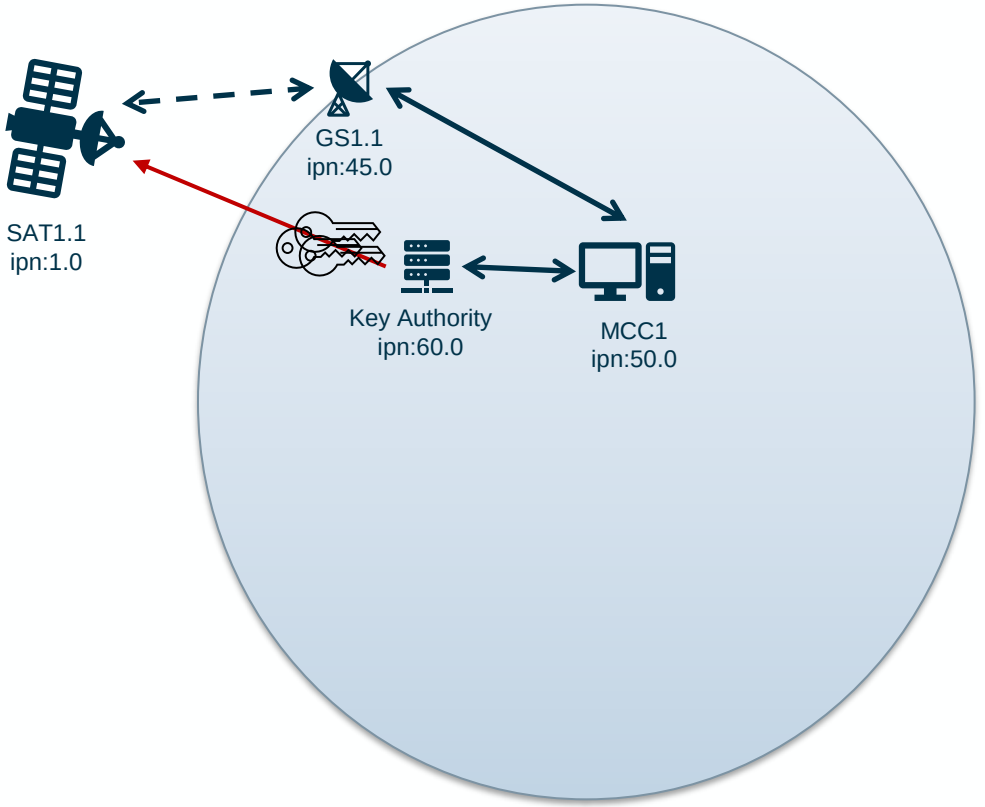
- Secure OOB channel to *client*
  - E.g. SDLS protocol
  - As backup, e.g. when a node needs to be re-added to the domain, resetting its *SK*, reestablishing *KA* trust, etc.

Functionality: OOB Initial Registration



- Initial *KD* needs to be exchanged manually
  → Only operators can register new nodes
  → Via secure OOB channel
    - Before launch: Direct access
    - After launch: SDLS, …
- After initial trust has been established: *KA* and *Client* can derive shared secret
- Optional: *Snapshot* of all domain *KD*s at deployment

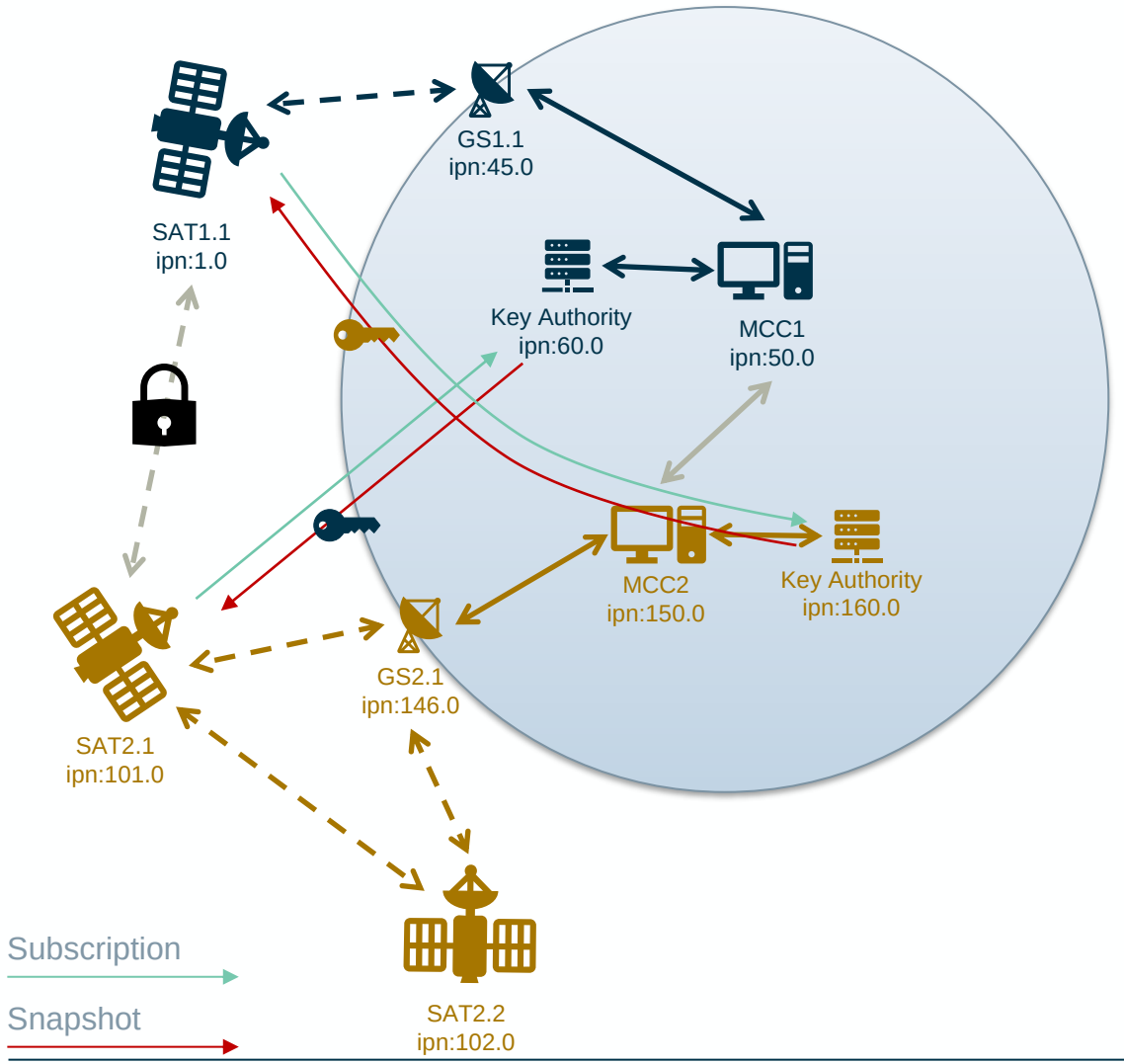
Functionality: *Key Authority*



- *Key Authority* collects updates
  - New *Client* registrations
  - Roll-Overs/*Client's* key renewal
  - Revocations
- *KA* sends snapshots back to *clients*
  - Periodically, e.g. once per week
  - Currently unicast, but multicast would decrease overhead
- System of clients + *KA* => *Application Domain*

Snapshot

Functionality: Inter-Domain Communication



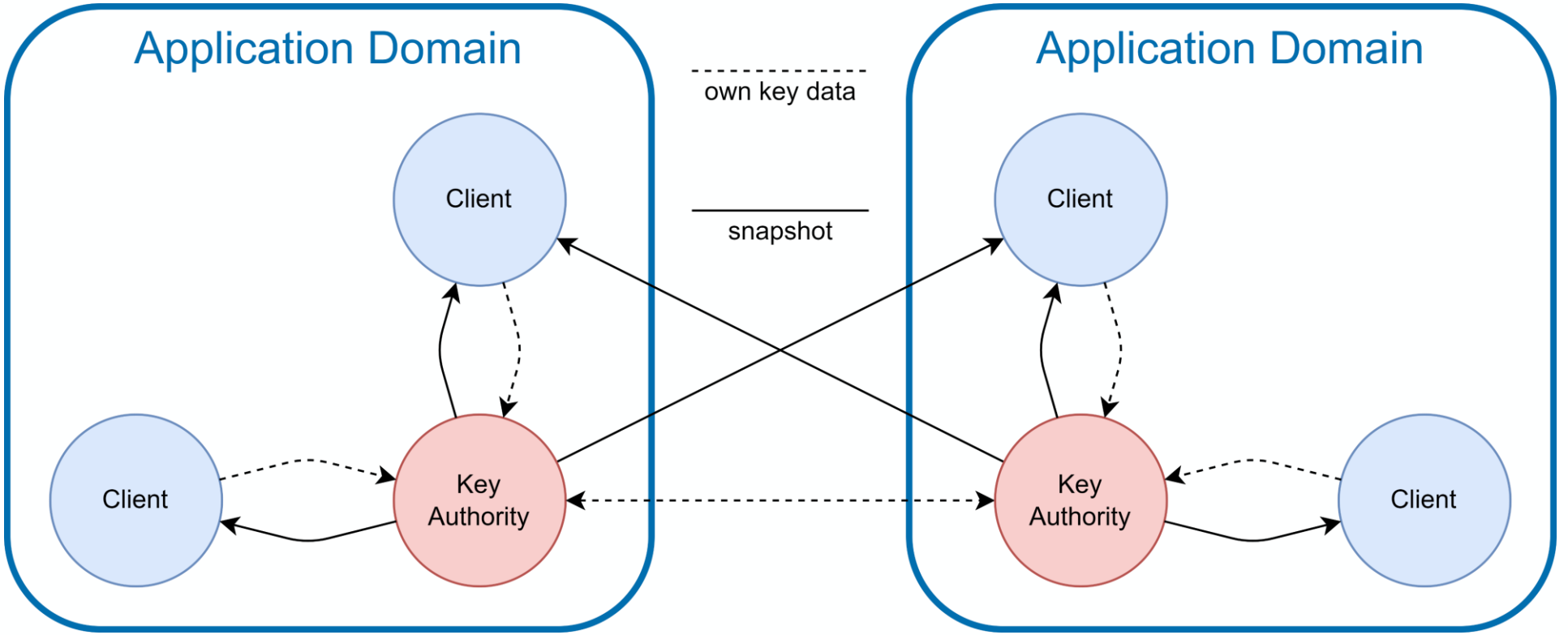
Inter-domain

- Key management within one *Application Domain*
- *Clients* can subscribe to another domain's *KA*
- KA controls which keys are shared with which client of which *AD*
  - Limited view of network for outsiders
- A client's *KA* controls to which foreign *KAs* its client can subscribe to

Requirements/Constraints

- To trust the other *KA*, clients need to know its *PK*
  → *KAs* need each other's *KD*
  → IGCA

Formalized


Application Domain
Client
Client
Key Authority
Application Domain
Client
Key Authority
Client
own key data
snapshot

Revocations

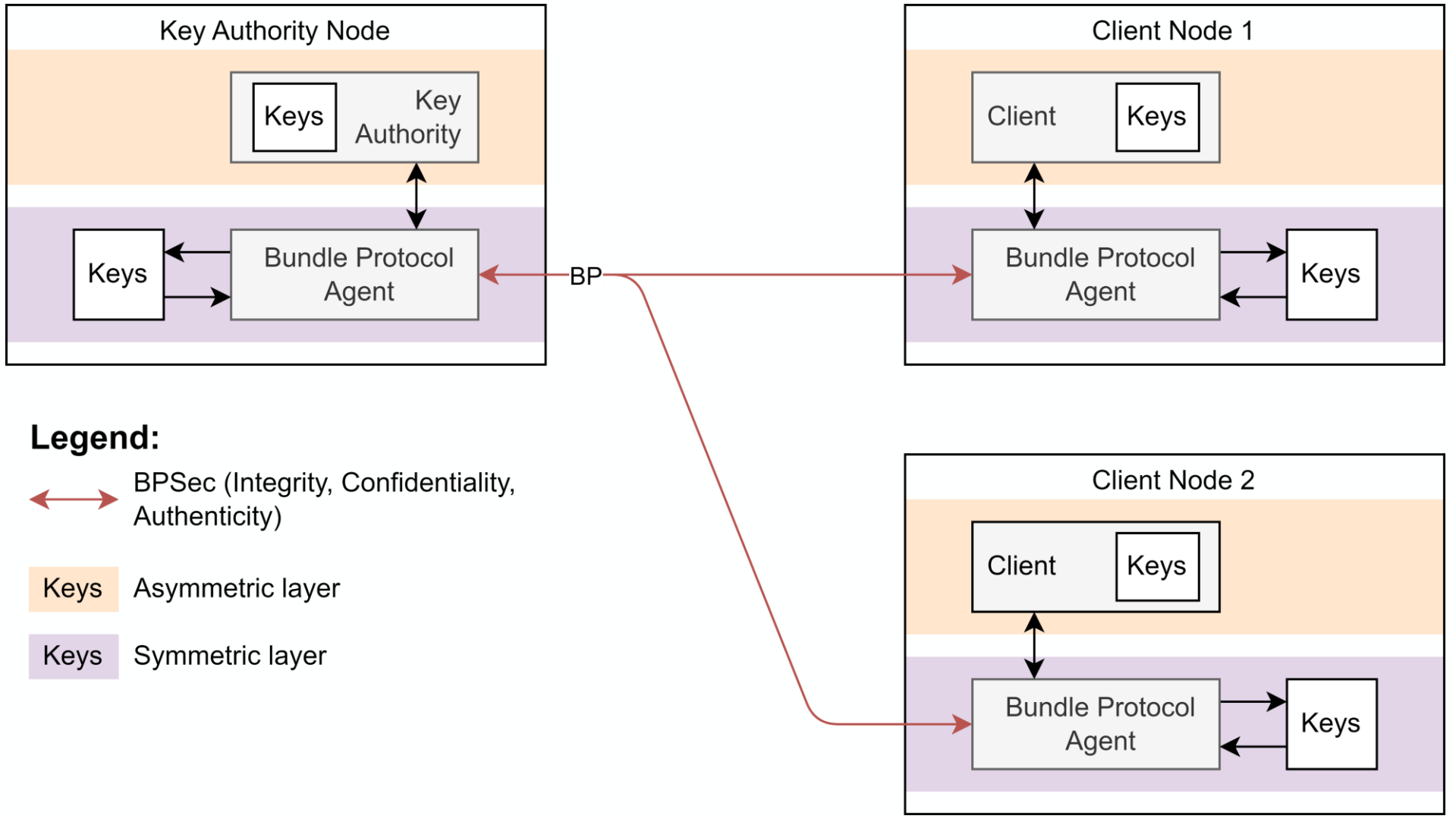
- Different proposals investigate revocations in DTN (for example: [1], [2])
- All are laid out for large numbers of nodes/certificates
  - [1]: 10K – 1M nodes
  - [2]: 250 nodes, 20K – 140K revoked certificates
- We assume near- to mid-term, so only a few nodes (50-100)

Functionality

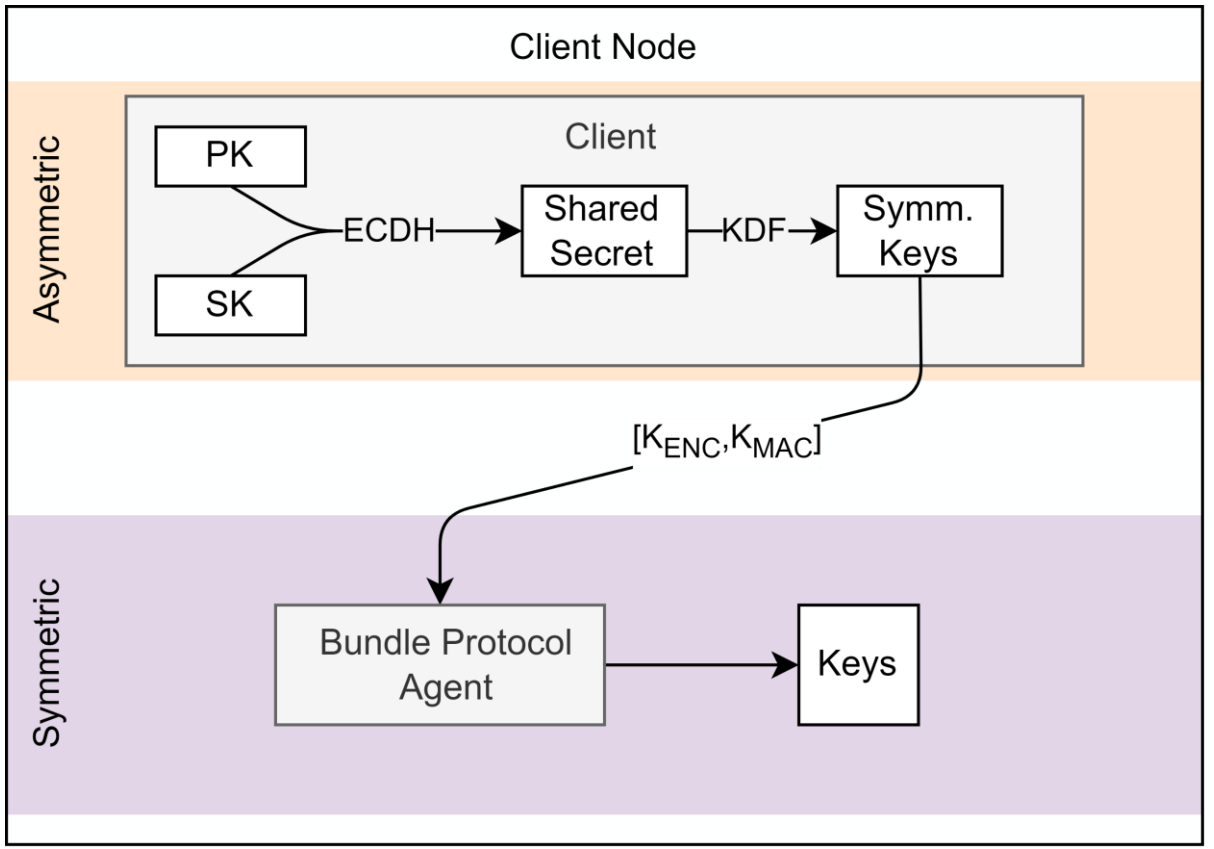
- Operator triggers revocation with *Key Authority*
- *Key Authority* immediately sends small revocation message to clients
  - Is affected by usual DTN challenges (Delays, Disruptions, …)
  - Could have QoS extension block with Information Type: Critical
    - See upcoming CCSDS BP QoS Extension orange book
- Next *snapshot* also contains this update
  - No need to keep revocation lists forever but only until the next scheduled snapshot distribution

[1] https://www.usenix.org/conference/usenixsecurity22/presentation/koisser
[2] https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/iet-ifs.2015.0438

Default Security Context Compatibility



Key Derivation



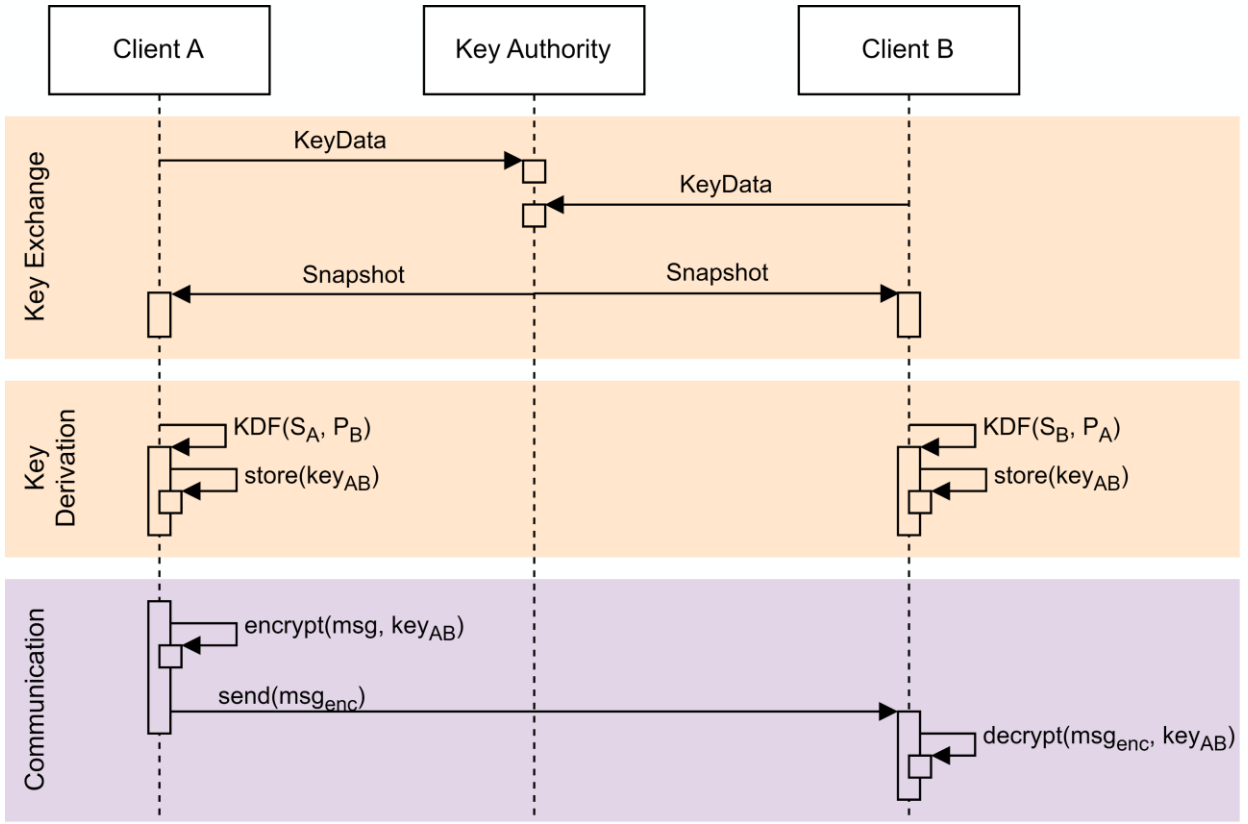
Symmetric Keys

- Are currently used for immediate use with BPSec
- Long lifetimes
  → Possible security risk

- Could instead be used as master key for key derivation or key wrapping
  - As described in [1]
  - Low level key usage needs to be investigated more

[1] Space Missions Key Management Concept

Key Exchange Sequence



- Key Exchange phase is secured by BPSec
- DSC with bundle integrity and confidentiality blocks
  (HMAC 256-512, AES 128/256)
  → PQ-protected
- Requires only two communication steps

External Limitations

Key Authority needs to derive symmetric session keys for every (subscribed) client

- We need a mechanism to verify message authenticity
- Digital signatures are not possible (missing security contexts)
- Each client receives its own unicast message from the *KA*
    - MAC'd with the corresponding symmetric key
      → Every symmetric key needs to be derived

No multicast

- Has not been specified yet
- If it was: See above
  → No way (yet) to verify message authenticity for multicast messages

No (perfect) forward secrecy

- Would be achieved by using an ephemeral *PK* for each message
    - Ephemeral *PK* is sent with the message
    - Receiver can use ephemeral *PK* and own *SK* to derive symmetric key
      → New symmetric key for each message
- Without new security contexts: Symmetric key is valid for *PK's* lifetime

Requirements for BP Agent Implementations

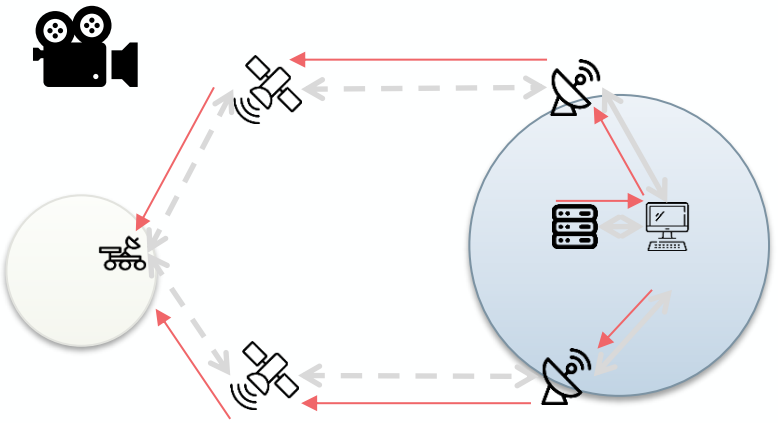
API to update keys during runtime
- To provide BP(sec) with new keys

A BPSec Policy framework
- To define rules for the technical usage of BIB and BCB
- To define key usage based on different factors like destination, key lifetime, etc.
- Dynamic policy updates required, e.g., for revocations or new *KD* from snapshots

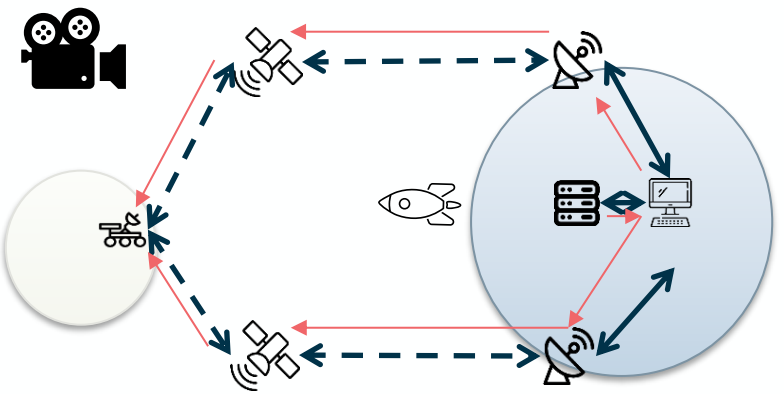
Ability to handle multiple keys for one policy
- Lifetimes may overlap each other
- BP must be able to select the valid key for the message creation time

Lifetimes for keys
- Policy framework must allow specification of key validity periods to know from/until when keys can be used
- Should be flexible enough to check creation timestamp of a message and use key based on that – no end-to-end connectivity in DTNs, thus, bundles might arrive delayed and were created with a now outdated key!

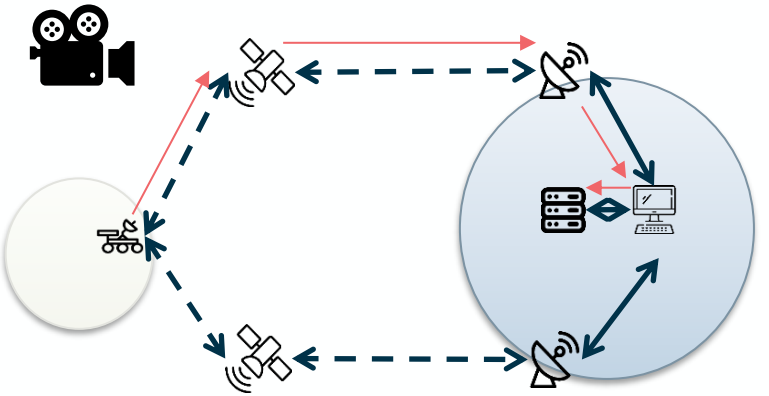
Test Cases for Evaluation



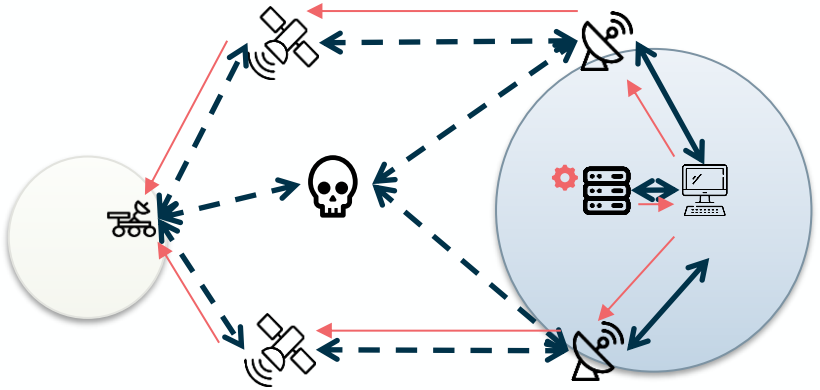
Key Distribution



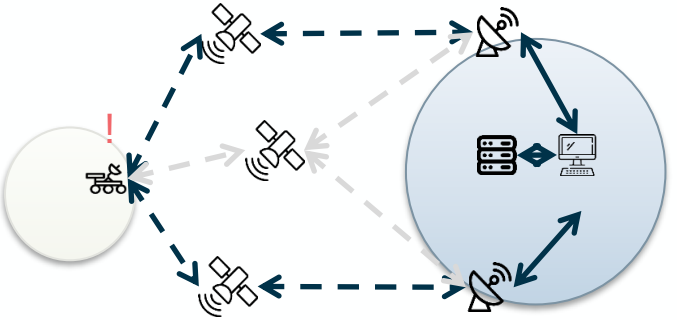
Nodes Joining



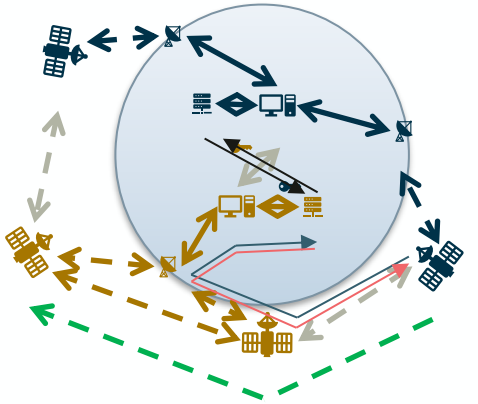
Key roll-over



Compromised Nodes/
Revocations



Expired Keys



Inter-Domain Communication

ESA BP DTN KM Demo

Future Work

Low-Level Key Management

- How are derived keys used exactly?

Asymmetric security contexts

- Allow signing of messages
  - Which in turn simplifies key exchange
    - Especially inter-domain
  - Enables integrity in multicast
- Allow the usage of ephemeral public keys for encryption
  - Greatly increase security through forward secrecy
  - Derives new symmetric key for each message

Multicast

- Reduces message overhead massively
- Instead of one unicast message per client per *KA* only one multicast message would be needed
- Requires asymmetric security context for signatures

Questions & Comments