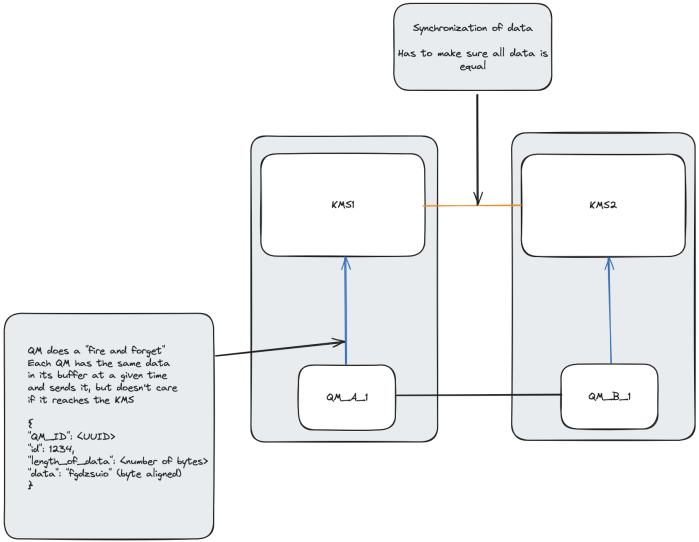
Bulk Key Sync via in-memory distributed database

Why we need our own distributed in-memory database:

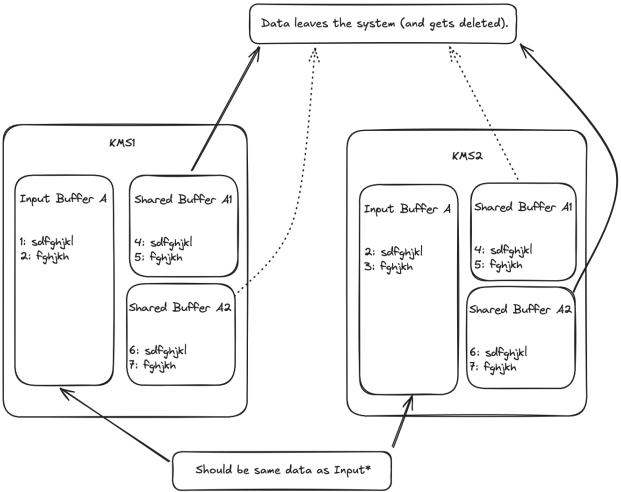
We need a distributed in-memory database with special requirements, because of the specialties of QKDN (no time to talk about this now):

- Special security considerations apply to guarantee unconditionally secure communitation
- Exactly two nodes
- Data loss is not a big problem, as long as it happens on both nodes
- Inconsistencies are not a problem in some constellations
- Reboot of one nodes deletes everything on both nodes
- Data comes into the database from one direction
- Data leaves the database into another direction and gets deleted
- We have to prevent that the same data data is read on both sides at the same time

Overview I


Synchronization of data
Has to make sure all data is equal
KMS1
KMS2
QM_A_1
QM_B_1
QM does a "Fire and forget"
Each QM has the same data
in its buffer at a given time
and sends it, but doesn't care
if it reaches the KMS
{
"QM_ID": <UUID>
"id": 1234,
"length_of_data": <number of bytes>
"data": "fgdzsuio" (byte aligned)
}

Overview II



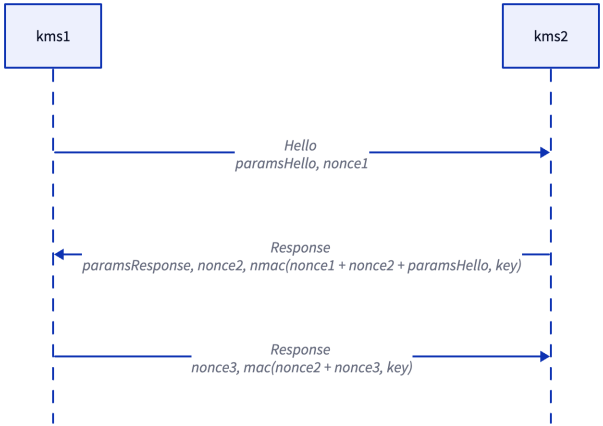
* But data can be lost or in wrong order, etc.

(QKD) KMS Handshake

(QKD) KMS Handshake

- Authentication and cipher suite agreement before distributing keys via KMS required
- QKD key material should be available on both communication partners $\longrightarrow$ can be used as pre-shared key for authentication
- We worked on a handshake protocol using a challenge-response method including a capability exchange between two KMS
- KMS can then forward secrets in a secured way using the previously negotiated crypto algorithms, etc.

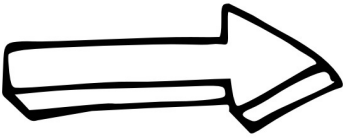
Protocol


kms1
kms2
Hello
paramsHello, nonce1
Response
paramsResponse, nonce2, nmac(nonce1 + nonce2 + paramsHello, key)
Response
nonce3, mac(nonce2 + nonce3, key)

To sum it up

Summary

- The QKDN technology is starting to make its way into the commercial market
- We now need to start the process of establishing common guidelines and standards
- We welcome interested collaborators



Link to our source code here...