AIT QKD Post Processing and Network Software

Oliver Maurhart, Stefan Petscharnig, Thomas Grafenauer, Bernhard Ömer, Michael Hentschel, Philipp-Sebastian Vogt, Christoph Pacher

AIT Austrian Institute of Technology GmbH, Center for Digital Safety & Security, 1210 Vienna, Austria.

Contact: oliver.maurhart@ait.ac.at, Stefan.Petscharnig@ait.ac.at

Introduction

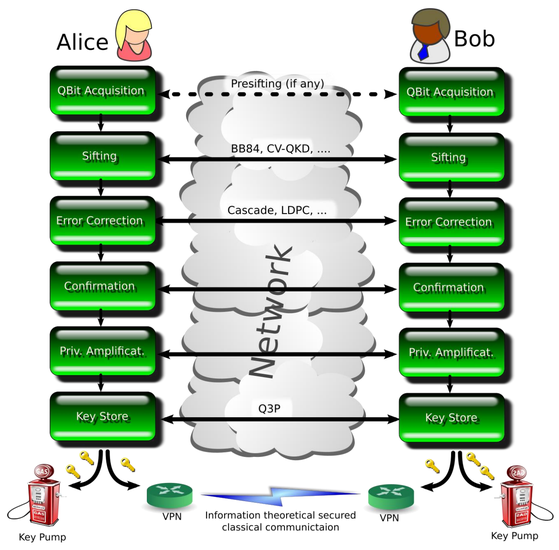
Quantum Key Distribution (QKD) involves in a first step a physical exchange of quantum signals between a pair of devices. Whatever the realization of this physical layer of QKD is, it outputs a pair of correlated values. In the second step these correlated values are transformed by a classical post-processing protocol into Information Theoretically Secure (ITS) keys. The AIT QKD framework (e.g. used in [1] and [2]) sees each QKD post-processing step as a dedicated process, a QKD module. Each QKD module has the built-in capability to run concurrently in parallel within a QKD post processing pipeline.

The AIT QKD software [3] is available under **different licence models** (research/commercial) and AIT welcomes suggestions from academic groups and industry to co-operate.

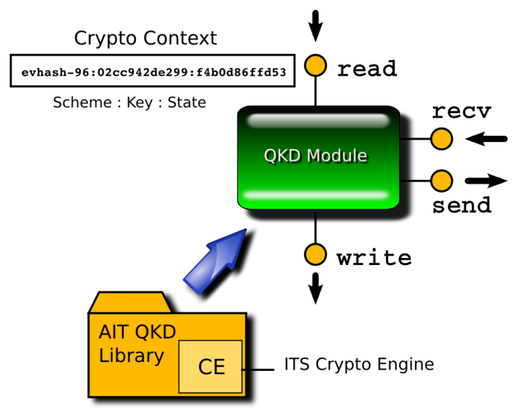
Acknowledgements: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 820466 (CiViQ).

QKD Pipeline

- A **QKD Pipeline** realizes quantum key distillation ranging from the detected signals up to a shared secret series of bits: the secure key. Starting with quantum signal acquisition each **QKD Module** concentrates on a dedicated task.
- Finally the shared secure keys are stored inside a **key store** which alongside a Crypto Engine provides means to directly extract keys from the system via "key pumps".
- Additionally a **VPN** can be fed by secure keys.



QKD Module



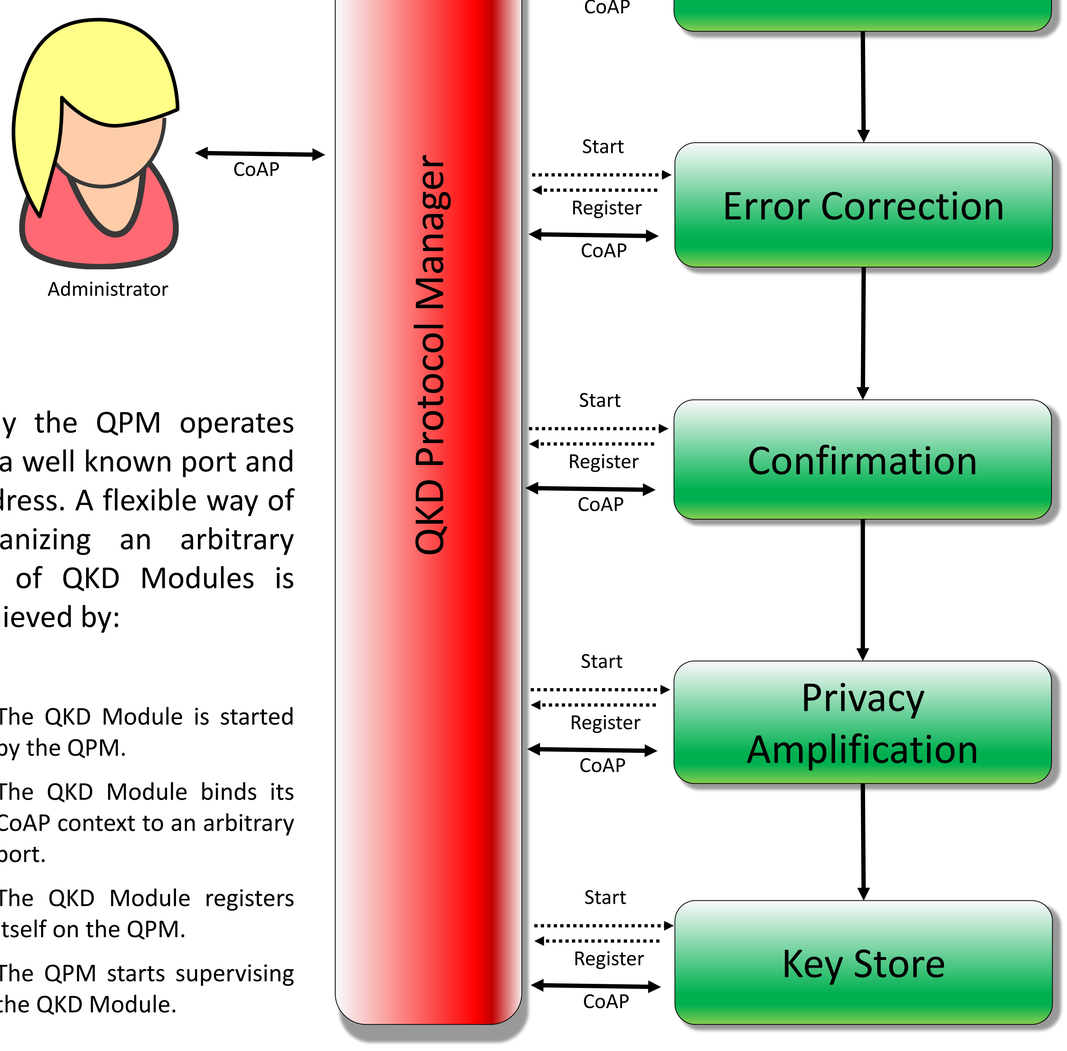
- The building block of a QKD pipeline is a single QKD Module. Such modules are incorporated as **UNIX processes**.
- The AIT QKD Library provides all functions for communication: from/to the next QKD Module, from/to the peer QKD Module. All these functions are mimicked as POSIX calls.
- With the help of ready-to-use ITS crypto functions the AIT QKD Software environment enables **ITS authenticated communication channels** with the peer QKD Module during post processing.
- Any QKD Module may change the current crypto setup during key distillation at will by instantiating new crypto contexts and finalizing the old one.

QKD Module Orchestration

The next release will provide every QKD Module with a **CoAP service context**. With this CoAP interface, a single QKD Module serves a RESTful API.

Since every QKD Module has its own service point, a central authority, the **QKD Protocol Manager (QPM)**, orchestrates all QKD Modules and therefore controls the **QKD pipeline**.

The QPM itself operates a CoAP service as well to which administrators can connect and control the whole setup.



Only the QPM operates on a well known port and address. A flexible way of organizing an arbitrary set of QKD Modules is achieved by:

1. The QKD Module is started by the QPM.
2. The QKD Module binds its CoAP context to an arbitrary port.
3. The QKD Module registers itself on the QPM.
4. The QPM starts supervising the QKD Module.

CoAP as a new IPC for QKD Modules

The next release uses the Constrained Application Protocol (CoAP) as an Inter-Process Communication Protocol (IPC) in between the QKD Modules.

CoAP is a lightweight integration of the ideas and concepts of the successful REST approach in today's network application landscape. At its simplest form CoAP consolidates only four verbs:

- GET: query QKD Module states (e.g. key bit per second)
- PUT: update QKD Module states (e.g. modify error correction parameters)
- POST: not used on a QKD Module
- DELETE: not used on a QKD Module

Since CoAP runs also primarily on UDP, we see the binary, low-overhead implementation suitable for environments with limited and tight resource budget.

Launching and Hibernating QKD Pipelines

The **QKD Pipeline** definition is comprised as **XML** file. It defines the set and order of the QKD Modules along with their configurations. These QKD pipeline files are **used by the QPM** to launch a full pipeline.

The QPM can instruct each QKD Module to **store its current state.**
The QPM can instruct each QKD Module to **load a previously saved state**.

This mechanism can be used for satellite-based QKD: the QPM on the satellite starts and maintains separate QKD pipelines with each ground-station it passes.

References

[1] M. Peev et al., ``The SECOQC Quantum Key Distribution Network in Vienna'', New Journal of Phys., **11**, 075001 (2009). https://doi.org/10.1088/1367-2630/11/7/075001

[2] T. Gehring et al., ``Implementation of continuous-variable quantum key distribution with composable and one-sided-device-independent security against coherent attacks'', Nature Communications **6**, 8795 (2015). http://dx.doi.org/10.1038/ncomms9795

[3] AIT QKD Software platform, https://sqt.ait.ac.at/software/projects/qkd