Towards an Open-source Software Platform for Numerical Key Rate Calculation of General Quantum Key Distribution Protocols

Jie Lin[1], Ian George[1], Kai-Hong Li[1], Kun Fang[1], Twesh Upadhyaya[1], Natansh Mathur[1,2], Max Chemtov[1], Shlok A. Nahar[1], Shahabeddin M. Aslmarand[1], Thomas Van Himbeeck[1], Yanbao Zhang[1,3], Christopher Boehm[1,4], Patrick Coles[1,5], Adam Winick[1], Wenyuan Wang[1,*], and Norbert Lütkenhaus[1,†]

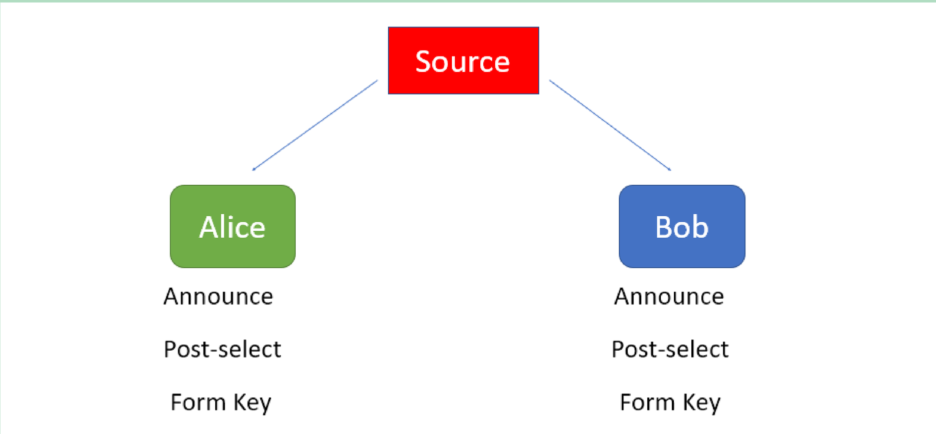
[1] *Institute for Quantum Computing and Department of Physics and Astronomy, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*
[2] *India Institute of Technology Roorkee, Roorkee, India, 247667*
[3] *NTT Basic Research Laboratories and NTT Research Center for Theoretical Quantum Physics, NTT Corporation, 3-1 Morinosato-Wakamiya, Atsugi, Kanagawa, Japan 243-0198*
[4] *University of Freiburg, Freiburg im Breisgau, Germany 79085*
[5] *Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, US*
**email: wenyuan.wang@uwaterloo.ca; †email: lutkenhaus.office@uwaterloo.ca*

IQC Institute for Quantum Computing

A **numerical approach** for the calculation of QKD key rates allows a uniform framework to be applied to **general QKD protocols**. Based on our group's previous work, we would like to build a **universal software platform that is fully modularized and user-friendly**, where one can easily swap in and out different QKD protocol descriptions, channel simulation models or experimental data, backend numerical solvers, and parameter optimization algorithms. Our goal is to build an **open-source platform** that can be both useful for theorists testing new protocols as well as experimentalists looking for optimal parameters or analyzing their experimental data.

Background

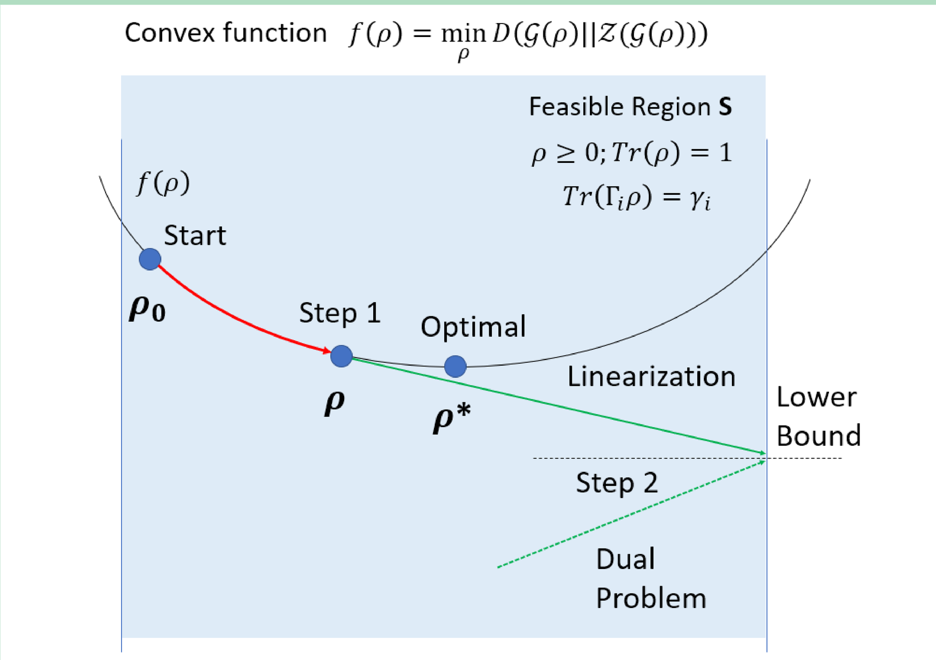
Here we first briefly introduce the **numerical approach** of calculating key rate based on our group's previous works [1,2].



A **general QKD protocol** can be described in a "prototypical" form as above. It is described by:

- a set of **POVMs** (measurements),
- **Kraus operators** (announcements and sifting), and
- **Keymap** (forming key).
- **Constraints** from observations: the *expectation values* of POVMs.

Once the description is given, the key rate (privacy amplification part) takes the form of $\min f(\rho)$, where one needs to **minimize f depending on ρ (Eve's optimal attack)**.

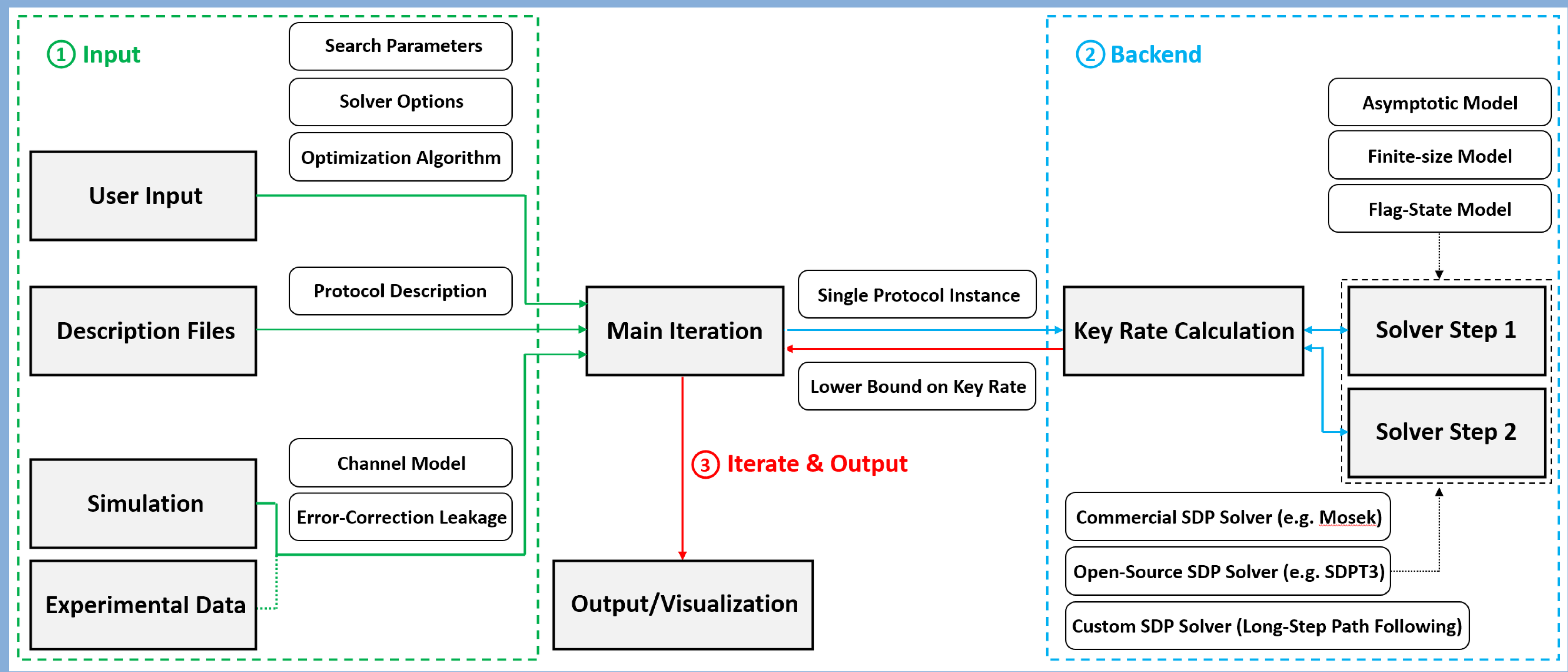


As f is a convex, we can calculate a lower bound on $\min f(\rho)$ by a **two-step procedure**:

*(1) finding a **near-optimal** ρ that gives us an **upper bound**.*
*(2) switch to a **dual problem** with a linearization of f to calculate a **lower bound**.*

The gap between upper/lower bounds reflects closeness of the solution to optimality.

Architecture



We follow **a modularized** approach, and build the software with three main types of modules, each independent from the rest and is easily swappable between different modules.

1. User supplies **input data**: a description of the QKD protocol, the channel model, parameters and solver settings
   - Description file easily caters for various QKD protocols and side-channels
   - Channel model can be from theoretical simulation, can also be from real experimental data
2. The **backend solver** follows the two-step numerical approach to calculate key rate for a given instance of protocol
   - The model can be easily swapped between e.g. asymptotic infinite-data case or finite-size case
   - Flag-state model (utilizing protocol structure symmetry) can be used to reduce computation dimensions
   - The backend SDP solver can be swapped between different software libraries (e.g. commercial/open-source solvers), as well as custom solver algorithms (such as long-step path following for step 1)
3. The **main iteration** can perform optimization of parameters
   - can be used to find highest key rate for a simulation, or optimal parameters for an experiment
   - can use different parameter optimization algorithms (brute-force, local search, global search, etc.)

Our goal is to build a **numerical software framework** that is easy to use and modularized. It will be able to cater for **various protocols and types of side-channels**. We hope it can be a testing tool for theorists and data analysis tool for experimentalists. Eventually, our goal is to build an **open-source platform** for the QKD community to access and build upon.

Applications and Ongoing Works

1. **Protocol description** - applying to various protocols & side-channels
   - DM-CVQKD (Jie, Twesh, Max) [3]
   - Detector efficiency mismatch (Yanbao, Jie) [4]
   - Unbalanced BB84 encoding (Nicky) [5]
   - Modeling of optical channel (Shahab)
2. **Backend Solver**
   - Finite-size model (Ian) [6]
   - Speedup from structure of protocols e.g. using flag-states (Yanbao, Nicky) [4,5]
   - High efficiency custom solver (Kun, Cunlu)
3. **Main iteration**
   - Local/global search (Wenyuan, Natansh)

- Architecture & Unifying of interfaces (Wenyuan)

References

[1] Patrick J. Coles, Eric M. Metodiev, and Norbert Lütkenhaus. Nature Communications 7.1 (2016): 1-9.

[2] Adam Winick, Norbert Lütkenhaus, and Patrick J. Coles. Quantum 2 (2018): 77.

[3] Jie Lin, Twesh Upadhyaya, and Norbert Lütkenhaus. Physical Review X 9.4 (2019): 041064.

[4] Yanbao Zhang, Patrick J. Coles, Adam Winick, Jie Lin, Norbert Lütkenhaus, arXiv: 2004.04383, accepted as QCrypt 2020 talk.

[5] Nicky Kai Hong Li and Norbert Lütkenhaus. arXiv:2007.08662.

[6] Ian George, Jie Lin, Norbert Lütkenhaus, arXiv: 2004.11865, accepted as QCrypt 2020 talk.