Satellite Quantum Modelling & Analysis Software

Version 2.0.0: Documentation

D. McArthur, J. S. Sidhu, T. Brougham, R. G. Pousa and D. K. L. Oi

University of Strathclyde

Department of Physics

John Anderson Building, 107 Rottenrow East, Glasgow, G4 0NG

May 2023

Chapter 1

Introduction

1.1 Mission statement

We provide a numerical key rate analysis, which determines the amount of expected key generation in satellite-based quantum key distribution protocols. This key length analysis will help develop an intuition on the effects of different operational scenarios on the key rate and inform the development of source and receiver systems. This numerical toolkit will provide a guide to future satellite missions.

1.2 Scope of current version

The latest release of SatQuMA, version 2.0.0, represents a major release update (now using semantic versioning) to the software which involves a redesign of the structure of the code as well as some updated features. The new version has a modular structure, which segregates the core components of the software and should make future adaptations and innovations easier to implement. User specified parameters are now submitted via an input file, where advanced parameters (controlling optimisation features, tail bounds and error correction modelling for example) can be set using a secondary input file, if present, otherwise default parameters are used. Multiple unique satellite overpasses can now be included in a single set of calculations either by reading loss files named using a consistent convention between overpasses – by passing a formattable base string for the filenames – or by using the new built-in loss modelling functionality which requires parameters for the transmitter, receiver, and atmospheric transmissivity[1].

As with previous versions, we implement an optimised, asymmetric two-decoy state BB84 protocol with weak coherent pulses and an example file is provided giving elevation, and time, dependent system losses based on a zenith pass for the Micius satellite to the (up-

[1]This can be obtained from a radiative transfer suite such as MODTRAN [1] or libradtran [2].

graded) receiver at Delingha. Protocol parameters can still be either optimised or specified, collectively.

1.3 List of updates

The latest release contains many small improvements over the previous v1.1 release, listed here in no particular order:

- The software architecture has been modularised, with directories containing functions that fall under the broad categories: `channel`, `input`, `key`, `optimize`, and `output`.
- Multiple unique satellite overpasses can now be considered during a set of calculations and the maximum elevation defining these trajectories, `theta_max`, is an iterable parameter.
- Channel losses can be modelled within SatQuMA, consisting of atmospheric, diffraction and intrinsic loss contributions.
- The excess loss parameter `ls` is now a float instead of an integer and iterations are specified as `start`, `stop` (inclusive), `No. of steps` (as opposed to `start`, `stop`, `step`).
- The parameter `tCompareEC` has been removed, which was a logic flag that toggled the comparison of optimised key lengths calculated with and without error correction.
- The error correction efficiency factor `fEC` can now be specified, it was previously set (and still defaults) to 1.16.
- The error correction model named `'logM'` now returns a minimum value based on the block size and the error correction efficiency factor, this corrects anomolous behaviour for large block sizes where unfeasibly small error correction terms could be returned.
- The toggling of output data files containing only the maximum calculated key length for a range of tranmsission windows and a set of iterable parameters is now controlled by the parameter `tdtOptData` (previously `tMultiData`). The redundant output flag `tOptiData` has been removed.
- Output data is now ordered according to iterable parameters (`dt`, `ls`, `QBERI`, `Pec`, and `theta_max`), followed by the calculated values (key length, QBER, *etc.*), then the protocol parameters, and finally the fixed system parameters.
- If the output file that SatQuMA is trying to write to is open, causing a `PermissionError`, then a new filename is created using a date and time stamp so that the calculation data is not lost.

1.4 Installation

The latest version of the SatQuMA software can be found at `https://github.com/cnqo-qcomms/SatQuMA`.

1.4.1 Required packages

The current version of SatQuMA requires an installation of Python 3.* and the following standard packages:

- scipy (SciPy)
- numpy (NumPy)
- sys
- time

Chapter 2

Theoretical summary

2.1 Background

Here we present a high-level summary of the equations required to calculate the secret key length (SKL) which appear in the current SatQuMA release. We do not, however, provide any form of derivation for these relations here.

2.1.1 Protocol and statistics

In our protocol, Alice randomly prepares a state in the basis X or Z, where we usually assume $X = \{D, A\}$ and $Z = \{H, V\}$, with one of three intensities $\mu = \{\mu_1, \mu_2, \mu_3\}$, each with a probability of being selected $P_\mu = \{P_{\mu_1}, P_{\mu_2}, P_{\mu_3}\}$.

Once Alice has sent her signals to Bob and the reconciliation process, error correction, and post-processing has been completed we can define some measurement statistics from the sifted key. We define the number of events, for each basis, for each intensity Alice could prepare

$$n_{\mathrm{X},\mu} := \{n_{\mathrm{X},\mu_1}, n_{\mathrm{X},\mu_2}, n_{\mathrm{X},\mu_3}\}, \tag{2.1}$$

$$n_{\mathrm{Z},\mu} := \{n_{\mathrm{Z},\mu_1}, n_{\mathrm{Z},\mu_2}, n_{\mathrm{Z},\mu_3}\}, \tag{2.2}$$

and similarly we define the number of bit errors, for each basis, for each intensity

$$m_{\mathrm{X},\mu} := \{m_{\mathrm{X},\mu_1}, m_{\mathrm{X},\mu_2}, m_{\mathrm{X},\mu_3}\}, \tag{2.3}$$

$$m_{\mathrm{Z},\mu} := \{m_{\mathrm{Z},\mu_1}, m_{\mathrm{Z},\mu_2}, m_{\mathrm{Z},\mu_3}\}. \tag{2.4}$$

2.1.2 Secure key length

The length of the secure key is given by [3]

$$\ell = \left\lfloor s_{\mathrm{X},0} + s_{\mathrm{X},1}\left[1 - h(\phi_{\mathrm{X}})\right] - \lambda_{\mathrm{EC}} - 6\log_2\left(\frac{21}{\epsilon_{\mathrm{s}}}\right) - \log_2\left(\frac{2}{\epsilon_{\mathrm{c}}}\right)\right\rfloor, \tag{2.5}$$

where the outer brackets indicate that we should take the floor of this expression. Here, $s_{\mathrm{X},0}$ is the number of vacuum events, $s_{\mathrm{X},1}$ is the number of single-photon events, and ϕ_{X} is the phase error rate in the sifted X basis. The parameter λ_{EC} provides an estimate, or a bound, on the number of bits required for error correction although this should be replaced with the *actual* number of bits used when this is known. The security parameters ϵ_{c} and ϵ_{s} are the prescribed security parameters which define the *correctness* and *secrecy* of the resulting key respectively. The binary entropy function used above is defined as

$$h\left(x\right) = -x\log_2 x - \left(1-x\right)\log_2\left(1-x\right). \tag{2.6}$$

2.1.3 Number of vacuum events

The number of vacuum events in a particular basis is evaluated as, for example,

$$s_{\mathrm{X},0} \geq \tau_0 \frac{\mu_2 n^-_{\mathrm{X},\mu_3} - \mu_3 n^+_{\mathrm{X},\mu_2}}{\mu_2 - \mu_3}, \tag{2.7}$$

where the probability that Alice sends an n-photon state is given by the Poisson distribution,

$$\tau_n = \sum_{j=1}^{3} \frac{\mathrm{e}^{-\mu_j}\mu_j^n p_j}{n!}. \tag{2.8}$$

In order to account for statistical fluctuations in the expected number of n-photon events, we apply the Chernoff bound and define the functions [4]

$$n^+_{\mathrm{X},\mu_j} = \frac{\mathrm{e}^{\mu_j}}{p_{\mu_j}}\left[n_{\mathrm{X},\mu_j} + \log_{\mathrm{e}}\left(\frac{21}{\epsilon_{\mathrm{s}}}\right) + \sqrt{2n_{\mathrm{X},\mu_j}\log_{\mathrm{e}}\left(\frac{21}{\epsilon_{\mathrm{s}}}\right) + \log_{\mathrm{e}}\left(\frac{21}{\epsilon_{\mathrm{s}}}\right)^2}\right], \tag{2.9a}$$

$$n^-_{\mathrm{X},\mu_j} = \frac{\mathrm{e}^{\mu_j}}{p_{\mu_j}}\left[n_{\mathrm{X},\mu_j} - \frac{1}{2}\log_{\mathrm{e}}\left(\frac{21}{\epsilon_{\mathrm{s}}}\right) - \sqrt{2n_{\mathrm{X},\mu_j}\log_{\mathrm{e}}\left(\frac{21}{\epsilon_{\mathrm{s}}}\right) + \frac{1}{4}\log_{\mathrm{e}}\left(\frac{21}{\epsilon_{\mathrm{s}}}\right)^2}\right]. \tag{2.9b}$$

Note, we could also use these expressions to determine the number of vacuum events in the Z basis by exchanging the n_{X,μ_j} terms for the corresponding n_{Z,μ_j} terms.

2.1.4 Number of single-photon events

The number of single-photon events in a particular basis is similarly evaluated as,

$$s_{\mathrm{X},1} \geq \tau_1 \frac{\mu_1\left[n^-_{\mathrm{X},\mu_2} - n^+_{\mathrm{X},\mu_3} - \frac{\mu_2^2-\mu_3^2}{\mu_1^2}\left(n^+_{\mathrm{X},\mu_1} - \frac{s_{\mathrm{X},0}}{\tau_0}\right)\right]}{\mu_1\left(\mu_2-\mu_3\right) - \mu_2^2 + \mu_3^2}, \tag{2.10}$$

where τ_1 is given by (2.8). As with the expressions for the number of vacuum events, we can use the same expression to determine the number of single-photon events in the Z basis by again exchanging the n_{X,μ_j} terms for the corresponding n_{Z,μ_j} terms.

2.1.5 The phase error rate

We evaluate the phase error rate in the X basis according to

$$\phi_{\mathrm{X}} \leq \frac{v_{\mathrm{Z},1}}{s_{\mathrm{Z},1}} + \gamma\left(\epsilon_{\mathrm{s}}, \frac{v_{\mathrm{Z},1}}{s_{\mathrm{Z},1}}, s_{\mathrm{Z},1}, s_{\mathrm{X},1}\right), \tag{2.11}$$

where we use the function

$$\gamma\left(a,b,c,d\right) = \sqrt{\frac{\left(c+d\right)\left(1-b\right)b}{cd\log_{\mathrm{e}}2}\log_2\left[\frac{c+d}{bcd\left(1-b\right)}\frac{21^2}{a^2}\right]}, \tag{2.12}$$

and the single-photon events are defined as above in Sec. 2.1.4. We have also introduced the number of bit errors associated with single-photon events in Z

$$v_{\mathrm{Z},1} \leq \tau_1 \frac{m^{+}_{\mathrm{Z},\mu_2} - m^{-}_{\mathrm{Z},\mu_3}}{\mu_2 - \mu_3}, \tag{2.13}$$

where the bounds on the number of bit errors due to statistical fluctuations, $m^{\pm}_{\mathrm{Z},\mu_j}$ are given by (2.9a) and (2.9b) where the number(s) of events in the X basis, n_{X,μ_j}, should be substituted with the number(s) of bit errors in the Z basis, m_{Z,μ_j}.

2.1.6 Estimating the amount of error correction

We estimate the number of bits that need to be sacrificed to perform the error correction in two ways: the first is more accurate but also much more complex; the second is simple to implement but provides only a lower bound.

Method 1

We can evaluate the number of bits that we need to sacrifice for error correction as [5]

$$\begin{aligned}\lambda_{\mathrm{EC}} \approx\ & n_X h\left(\mathrm{QBER}_{\mathrm{X}}\right) + \left[n_X\left(1-\mathrm{QBER}_{\mathrm{X}}\right) - F^{-1}\left(\epsilon_{\mathrm{c}};\left\lfloor n_{\mathrm{X}}\right\rfloor, 1-\mathrm{QBER}_{\mathrm{X}}\right) - 1\right] \\ & \times \log_{\mathrm{e}}\left[\frac{\left(1-\mathrm{QBER}_{\mathrm{X}}\right)}{\mathrm{QBER}_{\mathrm{X}}}\right] - \frac{1}{2}\log_{\mathrm{e}} n_{\mathrm{X}} - \log_{\mathrm{e}}\left(\frac{1}{\epsilon_{\mathrm{c}}}\right),\end{aligned} \tag{2.14}$$

where we define the quantum bit error rate in the X basis as

$$\mathrm{QBER}_{\mathrm{X}} = \frac{\sum_j m_{\mathrm{X},\mu_j}}{\sum_j n_{\mathrm{X},\mu_j}}, \text{ for } j \in \{1,2,3\}, \tag{2.15}$$

and $F^{-1}(k;n,p)$ is the inverse (or quantile function) of the binomial cumulative distribution function

$$F(k;n,p) = \sum_{i=0}^{\lfloor k\rfloor}\binom{n}{i}p^i\left(1-p\right)^{n-i}.$$

Method 2

Another method, based upon the block size, estimates the lower bound on the error correction as

$$\lambda_{\mathrm{EC}} \geq f_{\mathrm{EC}} \sum_{j=1}^{3} n_{\mathrm{X},\mu_j} h\left(\mathrm{QBER}_{\mathrm{X}}\right), \tag{2.16}$$

with, typically, $f_{\mathrm{EC}} = 1.16$ the error correction efficiency factor.

Method 3

We can simply estimate the lower bound on the error correction based upon the total number of bit errors in the X basis

$$\lambda_{EC} \geq f_{\mathrm{EC}} \sum_{j=1}^{3} m_{X,\mu_j}. \tag{2.17}$$

Chapter 3

Satellite overpass geometry

In this chapter we will discuss the way in which we define a satellite overpass, and how SatQuMA expects a transmission window to be specified for the purposes of a key length calculation.

In SatQuMA, the transmission window is assumed to be symmetric in time (and elevation) about the local zenith by default and discretized into time-slots. An illustration of an ideal, zenith satellite overpass is shown in Fig. 3.1. To define the overpass geometry we set a minimum elevation angle, $\theta_{\min}$, for which signals can be transmitted between the satellite and OGS/receiver and choose a transmission window half-width, Δt. The software will attempt to generate secret key over each time-slot while $-\Delta t \leq t \leq \Delta t$.

We can also consider non-ideal satellite overpasses, where the orbital geometry can be defined in terms of either the maximum elevation of that orbit, relative to the local horizon of the receiver, or the orbit rotation angle ξ, relative to the centre of the Earth, required to transform an ideal zenith overpass into the non-ideal overpass specified. The relation between these two angles is illustrated in Fig. 3.2. The maximum elevation is related to the zenith orbit rotation angle ξ as

$$\theta_{\max} = \cos^{-1}\left\{ \frac{(R_{\mathrm{E}} + h_{\mathrm{sat}})\sin\xi}{\left[\begin{array}{l} (R_{\mathrm{E}} + h_{\mathrm{sat}})^2 + (R_{\mathrm{E}} + h_{\mathrm{OGS}})^2 - \\ 2\,(R_{\mathrm{E}} + h_{\mathrm{sat}})\,(R_{\mathrm{E}} + h_{\mathrm{OGS}})\cos\xi \end{array}\right]} \right\}, \tag{3.1}$$

where R_{E} is the radius of the Earth, h_{sat} is the orbital altitude of the satellite, and h_{OGS} is the altitude of the receiver/ground station.

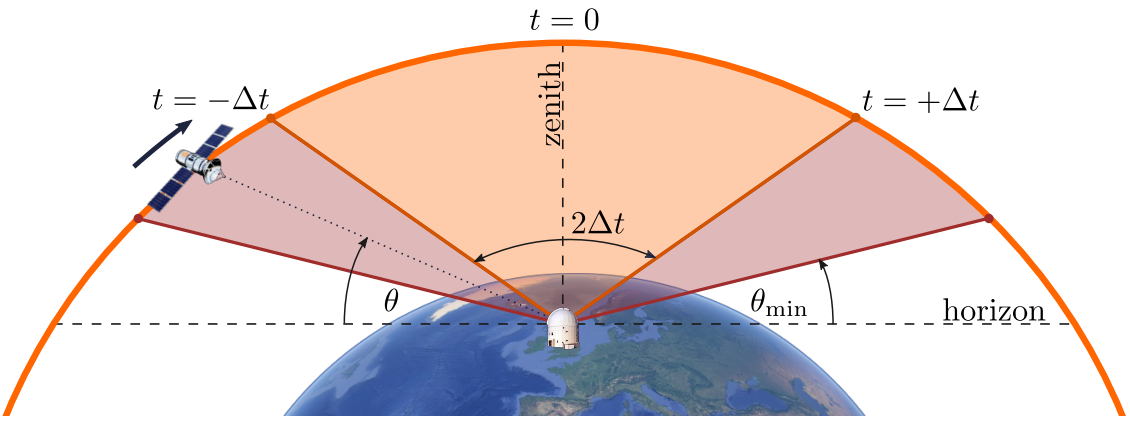

Figure 3.1: Satellite overpass geometry. A satellite passes over an OGS, where the satellite elevation angle $\theta \in [0°, 90°]$ is measured from the local horizon of the OGS. The satellite and OGS can only close a link when the satellite is above the minimum elevation angle $\theta_{\min}$. We assume that the actual transmission window is symmetric about the local zenith, which we label as $t = 0$, and that signals are sent while $-\Delta t \leq t \leq \Delta t$.

OGS photo: ESA
Globe: Google, Data SIO, NOAA, U.S. Navy, NGA, GEBCO, Landsat/Copernicus IBCAO U.S. Geological Survey

3.1 Channel loss: ADI model

We implement a model for the channel losses where the contributions to the total system loss are attributed to Atmospheric, Diffraction or Intrinsic (ADI) losses

$$\eta_{\text{tot}}(\lambda; \theta, R) = \eta_{\text{atm}}(\lambda; \theta, R) + \eta_{\text{diff}}(\lambda; R) + \eta_{\text{int}}, \tag{3.2}$$

with λ the transmission wavelength in vacuum, θ is the elevation angle, and R is the range between transmitter and receiver.

3.1.1 Atmospheric loss

For the atmospheric loss, effectively given by scattering and absorption within the atmosphere, we calculate the transmissivity, T_{atm}, for a given wavelength using a radiative transfer modelling suite such as MODTRAN or libradtran. This value is dependent upon the wavelength, elevation angle, transmitter-to-receiver range as well as any goegraphical and atmospheric conditions and parameters. The atmospheric loss is calculated from the transmissivity as

$$\eta_{\text{atm}} = -10 \log_{10} (T_{\text{atm}}) . \tag{3.3}$$

3.1.2 Diffraction loss

We estimate the diffraction losses by using the Fraunhoffer approximation to the Rayleigh-Sommerfeld diffraction integral to determine the power at the receiver, P_R, which is nor-

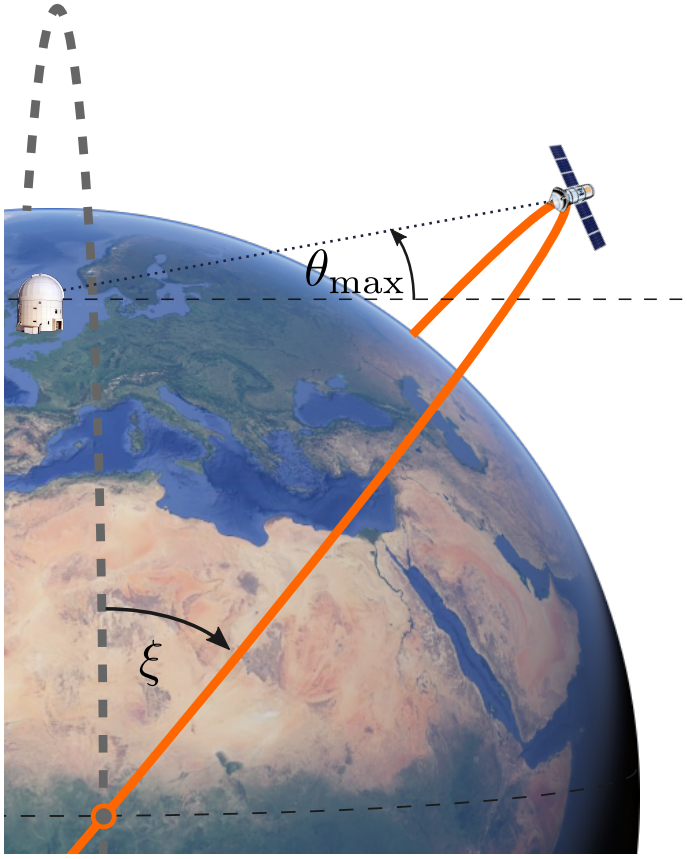

Figure 3.2: Non-zenith satellite orbits. The satellite orbit can be defined in terms of the maximum elevation that the satellite will reach, $\theta_{\max} \in [0°, 90°]$, above the local horizon and also the angle with respect to the centre of the Earth, ξ, that an ideal zenith orbit needs to be rotated to have equivalent geometry.

OGS photo: ESA
Globe: Google, Data SIO, NOAA, U.S. Navy, NGA, GEBCO, Landsat/Copernicus IBCAO U.S. Geological Survey

malised by the power at the transmitter, P_T. We define an arbitrary cylindrical coordinate system, assuming that the beam axis coincides with the centre of both apertures, with $\hat{\mathbf{z}}$ the vector aligned with the beam axis and z the distance between apertures. For clarity, at the two aperture planes we introduce different notation for the transverse radii and azimuthal angles as (ρ_0, ϕ_0) at the transmitter plane and (ρ, ϕ) at the receiver plane.

Assuming a truncated Gaussian field distribution the tranmsitted power for a given beam waist (at focus), w_0, is

$$P_T = 2\pi \int_0^{T_\mathsf{X}/2} \rho_0 \left| E_0(\rho_0, w_0) \right|^2 d\rho_0, \tag{3.4}$$

where the real electric field amplitude in the transmit plane is

$$E_0(\rho, w_0) = \sqrt{\frac{1}{2\pi}} \frac{1}{w_0} \exp\left(-\frac{\rho^2}{w_0^2}\right). \tag{3.5}$$

The power at the receive aperture (under the Fraunhoffer far-field approximation) is then,

$$P_R = 2\pi \frac{k^2}{z^2} \int_0^{T_\mathsf{X}/2} \rho \left| \int_0^{R_\mathsf{X}/2} \rho_0 E_0\left(\rho_0, w_0\right) J_0\left(\frac{k\rho_0\rho}{z}\right) d\rho_0 \right|^2 d\rho, \tag{3.6}$$

where $k = 2\pi/\lambda$ is the angular wavenumber and $J_n(x)$ is a cylindrical Bessel function of the first kind, of order n and argument x.

The diffraction loss is calculated from these power values as

$$\eta_{\text{diff}} = -10 \log_{10}\left(\frac{P_R}{P_T}\right). \tag{3.7}$$

3.1.3 Intrinsic loss

We typically take the intrinsic loss to be a constant value, encompassing system dependent loss mechanisms such as telescope and detector efficiencies, and enable a simplistic method of system scaling.

Chapter 4

Example of use

One of the main changes in this latest version of SatQuMA is the introduction of input files to set and control calculations – in previous versions parameters were set in the main script directly. The software looks for two distinct input files when run, although it only requires one, where the first contains the standard calculation parameters and the second optional file contains the so-called 'advanced parameters' (textiti.e. parameters that general users will not need to change) that control things such as the optimisation method, tail bounds and error correction functions. When the advanced parameter input file is not present in the run directory the software applies a set of default advanced parameters instead.

When run `SatQuMA_2.0.0.py` looks for input files named '`input.txt`' and '`input-adv.txt`' in the directory in which it was executed. The default filenames and path can be edited by the user in lines 21-27 of `SatQuMA_2.0.0.py`:

```
19 # Set the inpufile path and filename(s);
20 # Path to input files, empty string means location of execution
21 inputpath    = ''
22
23 # File name for main input parameters
24 filename     = 'input.txt'
25
26 # File name for advanced/optional input parameters (optional)
27 filename_adv = 'input-adv.txt'
```

The parameters to run a calculation in SatQuMA v2.0.0 are largely the same as in previous versions. Example input files, covering the three basic calculation set ups, are provided in the subdirectory inputfiles. Here we go through the process of setting up an optimisation calculation using SatQuMA, with line excerpts taken from `inputfiles/input.txt` as indicated by the specified line numbers.

The first parameter that must be set is the name of the desired security protocol, for which SatQuMA v2.0.0 currently only has one accepted value; the asymmetric BB84 security protocol with weak coherent pulses and 3 signal states (often referred to as 2 decoy

states) [3] is selected via the tag 'aBB84-WCP'.

```
#*****************************************************************************
# Select security protocol
#*****************************************************************************
'aBB84-WCP' # protocol = QKD security protocol. Only 'efficient-BB84' available.
```

4.1 Optimisation parameters

SatQuMA allows the main protocol parameters to be either optimised or specified, collectively. To enable the optimisation of the protocol parameters we must set the relevant boolean flag to `True`.

```
#*****************************************************************************
# Select SKL calculation type via tOptimise
#*****************************************************************************
#    True:  Optimise over the main protocol parameter and provide intial value.
#    False: Specify the main protocol parameter.
#*****************************************************************************
True  # tOptimise
```

We note that when SatQuMA parses input strings for boolean flags it accepts a range of values for both True (`True, T, Yes, 1`) and False (`False, F, No, 0`) which are case-insensitive.

Next, we need to specify the optimisable protocol parameters which for `aBB84-WCP` are the polarisation bias(es) P_{X}, the probabilities of sending the two signal pulses P_{μ_k}, and the signal intensities μ_k. If the protocol parameters are to be optimised then we need to specify the lower and upper bounds for each, which for the probabilities are $P \in (0, 1)$ and for the pulse intensities are $\mu \in (0, 1]$. Additionally, we can specify an initial value within these bounds to use for each optimised parameter – omitting this third value results in a random intial value being allocated. For example, if we consider an optimised calculation we might use the following values:

```
#*****************************************************************************
# Initialise protocol parameters
#*****************************************************************************
#     tOptimise = True: lower bound, upper bound [, initial value]
#     tOptimise = False: value
#*****************************************************************************
0.3, 1.0, 0.5     # PxA_i = Asymmetric polarisation probability (0:1)
0.6, 0.9999, 0.7  # pk1_i = Probability Alice prepares intensity 1 (0:1)
0.0, 0.4          # pk2_i = Probability Alice prepares intensity 2 (0:1)
0.3, 1.0, 0.8     # mu1_i = Intensity 1 (0:1]
0.1, 0.5, 0.3     # mu2_i = Intensity 2 (0:1]
```

In the above example, we have given the lower and upper bounds for each parameter in turn, and specified an initial value for each parameter except P_{μ_2} (`pk2_i`). While instead

for a non-optimised calculation we might specify the protocol parameter values:

```
0.5     # PxA_i = Asymmetric polarisation probability (0:1)
0.7     # pk1_i = Probability Alice prepares intensity 1 (0:1)
0.1     # pk2_i = Probability Alice prepares intensity 2 (0:1)
0.8     # mu1_i = Intensity 1 (0:1]
0.3     # mu2_i = Intensity 2 (0:1]
```

4.2 Calculation parameters

4.2.1 System parameters

We now specify the relevant parameters which characterise the performance of the system we wish to model. First, we specify the remaining protocol parameters: the intensity of the third weak coherent pulse (second decoy state) μ_3 and the prescribed errors in both correctness and secrecy, ϵ_c and ϵ_s respectively.

```
#*****************************************************************************
# Fixed system parameters
#*****************************************************************************
0.0         # mu3     = Weak coherent pulse 3 intensity
1e-15       # eps_c   = Prescribed error in protocol correctness
1e-9        # eps_s   = Prescribed error in protocol secrecy
```

Here, by setting $\mu_3 = 0$ we have chosen the third pulse in our protocol to be the vacuum state.

The probability of an after-pulse event for each detector must also be specified and also the transmission source repetition rate.

```
0.001       # Pap     = Afterpulse probability [0:1]
5e8         # fs      = Repetition rate of the source in Hz
```

Some parameters defining the transmission window are also required, such as the number of identical passes used to build keys, the minimum transmission angle, and the shift angle for the axis (at $t = 0$) of the symmetric transmsission window.

```
1           # NoPass  = Number of satellite passes
10.0        # minElev = Minimum elevation of orbit (deg)
0.0         # shift0  = Angle to shift centre of transmission window (deg)
```

We are now required to specify the iterable(sequenced) parameters, starting with the listed parameters: the maximum elevation (deg) during transmission ($\theta_{\max}$), the extraneous count probability (P_{ec}), and the intrinsic Quantum Bit Error Rate (QBER_{I}). These parameters should be specified as either single values, or a comma separated list of values for key length calculations to be looped over.

```
#*****************************************************************************
# Define listed parameters: theta_max(xi), Pec and QBERI
#*****************************************************************************
```

```
90.0, 80.0 # theta_max = Max elevation of satellite orbit wrt receiver (deg)
1e-8, 1e-7 # Pec       = Dark count probability [0:1]
0.001      # QBERI   = Intrinsic quantum bit error rate (QBER_I)
```

It is worth noting that SatQuMA uses random intialisation of the protocol parameters to find the optimal key through brute force and recycles intial values from previous successful calculations. This method has proved very successful, and at a fraction of the cost of more complex pre-optimisation schemes such as parameter space mapping via the method of steepest descent employed by programmes like Mathematica, but provides the smoothest results for iterated parameters when the better performing systems are calculated first. For this reason, when listing values for the three parameters above, it is best practice to specify `theta_max` from largest to smallest and both `Pec` and `QBERI` from smallest to largest.

4.2.2 Time window and system loss

This version of SatQuMA loops over the duration of the overpass time half-window (s) and the relative system loss (dB) by default. As such, these parameters are defined by specifying the input to sequencing functions: for `dt` (an integer) it is a `range` function, while for `ls` (a float) it is a `numpy linspace` function. The time window loop is therefore controlled by specifying the start, stop and step indices relating to the time slots (as specified in the input loss file, see 4.2.3).

```
#*******************************************************************************
# Define (inclusive) range for looped parameters: ls and dt
#*******************************************************************************
#*******************************************************************************
# dt = Index for windowing time-slot arrays, e.g. A(t)[t0-dt:t0+dt]
# In FS_loss_XI0.csv, 0 <= dt <= 346
#*******************************************************************************
200, 350, 10 # dt_range = Start, stop, step index
```

Note, these time slots are labelled relative to the overpass zenith for which we arbitrarily set $t = 0$. Here we have requested SKL calculations with overpass transmit time half-windows of, initially, up to 200 s then rising to half-windows of a duration of 350 s in 10 s increments. The minimum elevation for transmission (specified above) will override the values we have just specified, where necessary.

Next, we define the start, stop and *number of steps* for a loop over the excess system losses. That is, we can add additional loss to those specified in the loss file (converting from system efficiency).

```
#*******************************************************************************
# ls = Excess system loss(es) in dB
#*******************************************************************************
0, 12, 7   # ls_range = Start, stop, No. of steps
```

Here, we have considered systems which have 0 to 12 dB of excess loss, in steps of 0.5 dB, above the losses specified in the input loss file. Note, we could also consider negative

"*excess*" loss to simulate better performing systems.

A single transmission window half-duration, or excess loss, can be achieved by setting the minimum and maximum values of the sequences to the same value (or the number of steps to 1 for `ls`).

4.2.3 Input file options

This version of SatQuMA can either read in channel losses from a time/elevation *vs* link efficiency data file (in CSV format) to be specified, hereafter referred to as the 'loss file' or it can generate them directly. However, for the latter option an elevation dependent atmospheric loss file has to be specified otherwise the user is restricted to a very small selection of precalculated data.

So, the user must first specify if the losses are to be read-in from a file, otherwise they will be calcuated.

```
58 ##########################################################################
59 # Loss options
60 ##########################################################################
61 True  # tReadLoss = Read losses from a file?
```

If the user reuqests the losses to be read-in, the name of the loss file must be specified along with the path to that file and the column within that loss file which contains the link efficiency (losses).

```
64 ##########################################################################
65 # Input file options
66 ##########################################################################
67 ''                # loss_path = Path to loss file (empty string = current
    ↪ directory).
68 'FS_loss_XI0.csv' # loss_file = File containing loss data
69 3                 # lc = Column containing loss data in file (counting from 1)
```

An example file has been supplied with the software, named `FS_loss_XI0.csv`.

Alternatively, the user may request the losses to be calculated.

```
62 False  # tReadLoss = Read losses from a file?
```

This requires a range of parameters to be specified that the define the loss calculations (see Sec. 3.1)

```
64 ##########################################################################
65 # Input file options
66 ##########################################################################
67 False    # tWriteLoss = Write losses to a file?
68 ''       # atmfile = Name of atmospheric data file ('' = use default data)
69 15       # eta_int = Intrinsic system loss
70 500e3    # hsat = Satellite altitude in m
71 0e3      # hOGS = Receiver altitude in m
72 0.15     # aT = Transmitter aperture radii (m)
```

```
0.6       # aR = Receiver aperture radii (m)
0.15      # w0 = Beam waist at focus (m)
810       # wl = Wavelength (nm)
6371e3    # R_E = Radius of the Earth (m)
```

If the atmospheric file is not specified then the software will attempt to use one of the precalcuated atmospheric transmissivity functions, which were calculated for wavelengths $\lambda \in \{785, 790, \ldots, 850\}$ for a receiver at sea level and a transmitter above the atmosphere (> 100 km).

4.2.4 Output file options

During a calculation, SatQuMA can write to two different output streams: a local file and the standard output (IDE/terminal/screen/*etc*). We need to specify each output file that we wish to generate, there is the full data output (for each value of `dt`) for each set of parameters and also the optimised data (for the `dt` that gives the maximal key) for all sets of parameters. Additionally, for calculations that use the numerical optimiser, we can request a set of optimiser metrics for each calculation to monitor behaviour and performance. To request each of these files (in CSV format) we simply set the relevant boolean flags to be `True`.

```
#############################################################################
# Output file options
#############################################################################
True  # tPrint     = Print values to StdOut during calculations? True or False?
False  # tFullData = Write output to CSV file(s)? True or False?
True  # tdtOptData = Write out all dt optimised data in a single file? True or
 ↪ False?
False  # tMetrics  = Write out optimiser metrics? True or False?
```

Finally, we choose the path and base filename for the output.

```
''    # out_path   = Path for output file (empty string = current directory)
'out' # outbase    = Basename for output file(s) (excluding .csv)
```

4.2.5 Advanced parameters

We can also specify some of the advanced parameters using a secondary input file; these parameters have been set to default values which should only be changed by more advanced users looking for additional control over the calculations as they may strongly affect the software performance and resulting SKL.

Here we go through these parameters, with line excerpts taken from `inputfiles/input-adv.txt` as indicated by the specified line numbers.

We will specify that the protocol uses the Chernoff bounds, for all tail bounds, by selecting the default option from the list of bound functions.

```
#############################################################################
```

```
# Advanced/additional parameters
#*********************************************************************************
# Select the type of tail bounds to use for estimating statistical fluctuations
# in the count statistics.
#   'Chernoff'   = boundFunc[0] => Different upper and lower bound terms
#   'Hoeffding'  = boundFunc[1] => Same upper and lower bound terms
#   'Asymptotic' = boundFunc[2] => No tail bounds. This choice forces also
#                                  errcorrFunc = 'block' below.
'Chernoff' # boundFunc = Select an option from the list above.
```

Next we can select the method used to approximate the number of bits required for error correction (EC). Again, we select the default option from the list of EC functions, which are presented in decreasing order of complexity (*i.e.* difficulty to optimise) and accuracy.

```
#*********************************************************************************
# Select the method for estimating the number of bits sacrificed for error
# correction, listed below in order of decreasing precision (increasing
# smoothness).
#   'logM'  = logM(nX, QBERx, eps_c) = errcorrOpts[0]
#   'block' = 1.16 * nX * h(QBERx)   = errcorrOpts[1]
#   'mXtot' = 1.16 * mXtot           = errcorrOpts[2]
#   'None'  = 0                      = errcorrOpts[3]
#*********************************************************************************
'logM'      # errcorrFunc = Select a method from the list above.
```

The error correction efficiency factor can now also be specified.

```
#*********************************************************************************
# Set the error correction maximum efficiency factor
#*********************************************************************************
1.16        # fEC = Error correction tail bound efficiency factor
```

The security protocol sometimes needs to approximate zero as an arbitrarily small number which can also be specified

```
1e-10       # num_zero = Numerical value to use when denominators are ~zero
```

Due to the strong influence of the initial parameters on the final optimised output we include an option to set the minimum number of optimisations performed for a given set of parameters. Similarly, we can set the maximum number of optimisations to use.

```
10          # NoptMin =  Minimum No. of optimisations to strive for
1000        # NoptMax = Maximum No. of optimisations (not used)
```

The actual number of optimisations performed is also partially determined by the number of function evaluations reported by the optimisation algorithm. There are some additional flags which will stop the repeat optimisation loops for a given parameter set which are best employed when the user expects that there will be regions of zero SKL or where the optimised SKL varies little with the initial parameters. The optimisation behaviour can be further controlled by the following boolean flags.

```
True        # tStopZero = Stop optimizing if the first NoptMin return SKL = 0?
```

```
False       # tStopBetter = Stop after NoptMin optimizations if SKL improved?
```

The first flag above allows the user to specify if SatQuMA should exit the optimisation loop if the optimised SKL is consistently returned as zero. The second flag controls whether the optimisation loop should be exited if the optimised SKL has improved during the loop.

We can choose the SKL optimisation method from the three constrained optimisation algorithms provided with `scipy` however we strongly recommend that users select the default option `'COBYLA'` to ensure robust performance.

```
#*******************************************************************************
# opt_methods = ['COBYLA','SLSQP','trust-constr']
#*******************************************************************************
'COBYLA'    # opt_method = Select a optimisation method
```

For each optimisation method there are additional parameters that contro the specific algorithm (see `input/optimiser_params.py: default_params_opt()` for default values for each). For the `'COBYLA'` method there are three control parameters which are named according to the `scipy` documentation.

```
#*******************************************************************************
# Optimiser options: minimize, method='COBYLA'
#*******************************************************************************
1200        # Nmax = Max No. of iterations
1.0e-12     # ctol = Constraint absolute tolerance.
0.002       # rhobeg = Reasonable initial changes to the variables.
```

4.3 Visualising data

Once the software has successfully completed the calculation we can visualise the output. Here we will focus on the optimised, sorted output data file `out_opt.csv`. For example, we can plot the secret key length as a function of the total system loss, both including and excluding the error correction estimation, using the following (minimum working) python code.

```
import numpy as np
data = np.loadtxt('out_opt.csv',skiprows=1,delimiter=',')
x   = data[:,32]       # Total system loss
y1  = data[:,5]        # SKL
y1_ = y1 + data[:,11] # SKL + lambda_EC
import matplotlib.pyplot as plt
fig1, ax1 = plt.subplots(1,1,figsize=(5,5))
ax1.semilogy(x,y1,'-',x,y1_,'--')
```

The resulting graph is shown in Fig. 4.1(a). We may also extend our plotting code to visualise the error rates associated with these finite keys, as shown in Fig. 4.1(b).

```
fig2, ax2 = plt.subplots(1,1,figsize=(5,5))
y2  = data[:,6:8]     # QBERx and phi_x
ax2.plot(x,y2[:,0],'-',x,y2[:,1],'--')
```

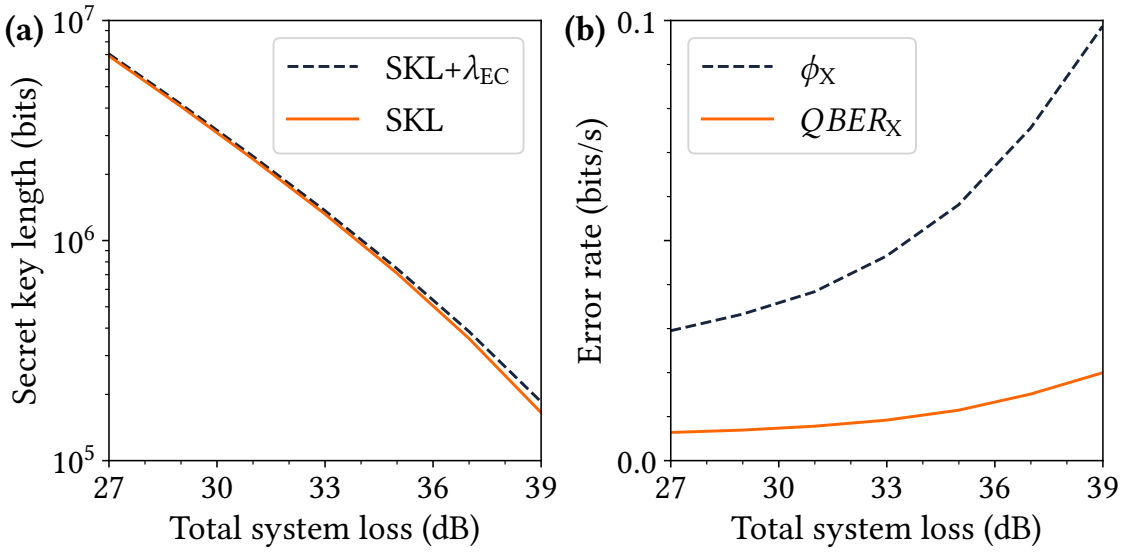

Figure 4.1: Total system loss in decibels against (a) secret key length, with and without error correction estimation, and (b) the phase and quantum bit error rates for the X basis.

Finally, we may also plot the change in the optimised protocol parameters, as shown in Fig. 4.2, using the following code.

```
fig3, ax3 = plt.subplots(1,1,figsize=(5,5))
y3  = data[:,17:25]  # Protocol parameters
ls  = [':',':','-','-','-','--','--','--'] # Linestyles
for ii in range(0,8,1):
    ax3.plot(x,y3[:,ii],ls[ii])
```

The various other parameters and output data from this file are listed in Table 4.1.

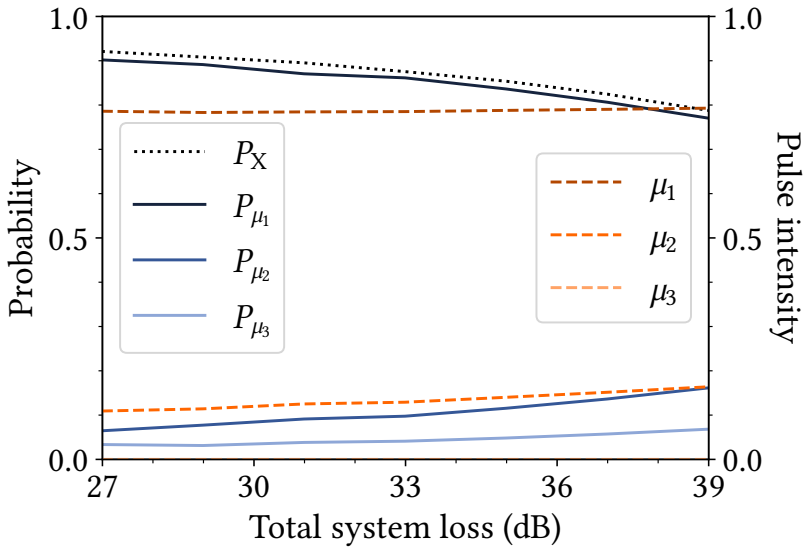

Figure 4.2: Optimised protocol parameters (for asymmetric BB84) as a function of the total system loss in decibels. Note, $\mu_3 = 0$.

Index	Variable	Symbol	Description
0	`dt`	Δt	Transmission half-window duration (s).
1	`ls`	ls	Additional, constant loss added to system loss (dB).
2	`QBERI`	$\mathrm{QBER_I}$	Intrinsic quantum bit error rate (%).
3	`Pec`	P_{ec}	Extraneous count probability.
4	`max_elev`	$\theta_{\max}$	Maximum elevation of satellite overpass (deg).
5	`SKL`	ℓ	Secret key length (bits).
6	`QBERx`	$\mathrm{QBER_X}$	Quantum bit error rate for X basis (bits/s).
7	`phi_x`	ϕ_{X}	Phase error rate for X basis (bits/s).
8	`nX`	n_{X}	Number of events in the X basis.
9	`nZ`	n_{Z}	Number of events in the Z basis.
10	`mX`	m_{X}	Number of errors in the X basis.
11	`lambdaEC`	λ_{EC}	Estimate of number of bits used for error correction.
12	`sX0`	$s_{\mathrm{X},0}$	No. of vacuum events for X basis.
13	`sX1`	$s_{\mathrm{X},1}$	No. of single photon events for X basis.
14	`vz1`	$v_{\mathrm{Z},1}$	No. of bit errors associated with single-photon events in Z basis.
15	`sZ1`	$s_{\mathrm{Z},1}$	No. of single photon events for Z basis.
16	`mpn`	$\sum_j \mu_j/3$	Mean transmitted photon number.
17	`PxA`	$P_{\mathrm{X,A}}$	Probability Alice sends a signal in the X basis.
18	`PxB`	$P_{\mathrm{X,B}}$	Probability Bob receives a signal in the X basis.
19	`P1`	P_{μ_1}	Probability of sending pulse 1.
20	`P2`	P_{μ_2}	Probability of sending pulse 2.
21	`P3`	P_{μ_3}	Probability of sending pulse 3.
22	`mu1`	μ_1	Intensity of pulse 1.
23	`mu2`	μ_2	Intensity of pulse 2.
24	`mu3`	μ_3	Intensity of pulse 3.
25	`eps_c`	ϵ_{c}	Correctness parameter.
26	`eps_s`	ϵ_{s}	Secrecy parameter.
27	`Pap`	P_{ap}	Probability of after-pulse event.
28	`NoPass`	M	Number of satellite overpasses.
29	`fs`	f_{s}	Source repetition rate (Hz).
30	`min_elev`	$\theta_{\min}$	Minimum elevation of satellite for transmission (deg).
31	`shift_elev`	θ_{shift}	Angle to shift transmission window central axis (deg).
32	`ls+sysLoss`	$\eta_{\mathrm{loss}}^{\mathrm{sys}}$	Total system loss (dB).

Table 4.1: Table of data written to main output file(s) giving the index (data column), python variable name, associated mathematical symbol and a brief description.

Chapter 5

Bugs and future releases

SatQuMA is still very much under development with improved versions planned for release in the near future. The current version employs out-of-the-box optimisation algorithms from the scipy package.

5.1 Known bugs

Unfortunately, as the function being minimized does not have derivatives that can be continuously defined for all regions of the relevant parameter space the scipy optimiser can report errors when it encounters locally flat regions of the function space, typically where the SKL drops to zero.

5.2 Reporting bugs

If you encounter any behaviour you consider unexpected, or anytime python raises an exception (excluding for user error), please send a copy of the main SatQuMA python file, and include any relevant output files, to the development team at the University of Strathclyde. It is always good practice to add python files to an archive (zip/tar/etc) before sending via email to avoid mail being rejected by mail scanning algorithms.

5.3 Suggested content

If there are any features that you would like to see added to this software then please feel free to contact the development team at the University of Strathclyde. Otherwise, if you are interested in collaborating on a project to either develop or utilise SatQuMA then please contact the development team!

5.4 Contact

Please contact the development team through via Dr D. K. L. Oi at daniel.oi@strath.ac.uk.

Bibliography

[1] A. Berk, P. Conforti, R. Kennett, T. Perkins, F. Hawes, and J. van den Bosch, "MODTRAN6: a major upgrade of the MODTRAN radiative transfer code," *Proc. SPIE 9088, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XX, 90880H*, June 2014. http://modtran.spectral.com

[2] C. Emde *et al.*, "The libradtran software package for radiative transfer calculations (version 2.0.1)," *Geoscientific Model Development*, vol. 9, pp. 1647-1672, May 2016. https://www.libradtran.org

[3] C. C. W. Lim, M. Curty, N. Walenta, F. Xu, and H. Zbinden, "Concise security bounds for practical decoy-state quantum key distribution," *Phys. Rev. A*, vol. 89, p. 022307, February 2014.

[4] H.-L. Yin, M.-G. Zhou, J. Gu, Y.-M. Xie, Y.-S. Lu, and Z.-B. Chen, "Tight security bounds for decoy-state quantum key distribution," *Sci. Rep.*, vol. 10, p. 14312, August 2020.

[5] M. Tomamichel, J. Martinez-Mateo, C. Pacher, and D. Elkouss, "Fundamental finite key limits for one-way information reconciliation in quantum key distribution," *Quant. Inf. Proc.*, vol. 16, p. 280, October 2017.