SatQuMA: Satellite Quantum Modelling and Analysis
Software architecture, description and timeline (v0.1)

D. McArthur, J. S. Sidhu, T. Brougham, S. Mohapatra, R. Gonzalez Pousa, and D. K. L. Oi
CNQO, SUPA Department of Physics, University of Strathclyde, Glasgow G4 0NG, UK

6th November 2020

1 Software mission statement

We provide a modelling tool for the design and analysis of free-space and satellite quantum key distribution (QKD). The main function will be to determine the amount of secret key that can be generated from finite and time dependent optical channels. Though primarily designed for satellite QKD, the software architecture will be sufficiently flexible to model arbitrary time dependent channel characteristics. This could include static free-space installations where the channel may be subject to fluctuations (e.g. turbulence) or during single or multiple satellite overpasses of a ground station, taking into account finite block size effects. This is performed using a numerical key rate analysis based on expected channel transmission, system performance, and source-receiver geometry together with an optimisation routine to determine protocol parameters. A modular structure will enable users to define custom source and receiver models, following prescribed templates, and to include various security protocols. The initial implementation is for asymmetric BB84 weak coherent pulses with two-decoy states (3 intensities). 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 as well as the other free-space QKD deployment.

2 Software overview

The software will have a modular architecture where the constituent modules can be grouped broadly into three main groups: core functionality, physical systems, and ancillary functions. Each of the modules in these groups are described in the following section, in Secs. 3.1, 3.2, and 3.3 respectively. A simple flowchart showing the general calculation path, highlighting interfaces between modules, is shown in Fig. 1. A more detailed flowchart, showing the core functionality, is presented in Fig. 2. An example of the type of output the software generate is then given in Fig. 3. In Sec. 4 we discuss the proposed data structures, and in Sec. 6 we give a briefly documented timeline for the projected future releases of the software.

3 Module descriptors

In this section we provide descriptors of the proposed modules detailing information such as their general function, proposed contents, interfaces and potential future developments.

3.1 Core functionality modules

These modules form the backbone of the code, and are responsible for converting input parameters into output data.

3.1.1 Key module

This is the most important module, around which the main software will be constructed. The key module will primarily calculate the quantum key length/rate for a given QKD system, under the constraints of a specific security protocol. The Key module will therefore be required to interface with all of the physical system modules (see Sec. 3.2). Eventually, we aim to have coded various QKD security protocols which will be stored in a repository, either within the module itself or in a separate, connected sub-module file. Each protocol will be built around a template which satisfies the constraints of the relevant data structures and module interfaces. As well as returning the quantum key length (or rate, in the

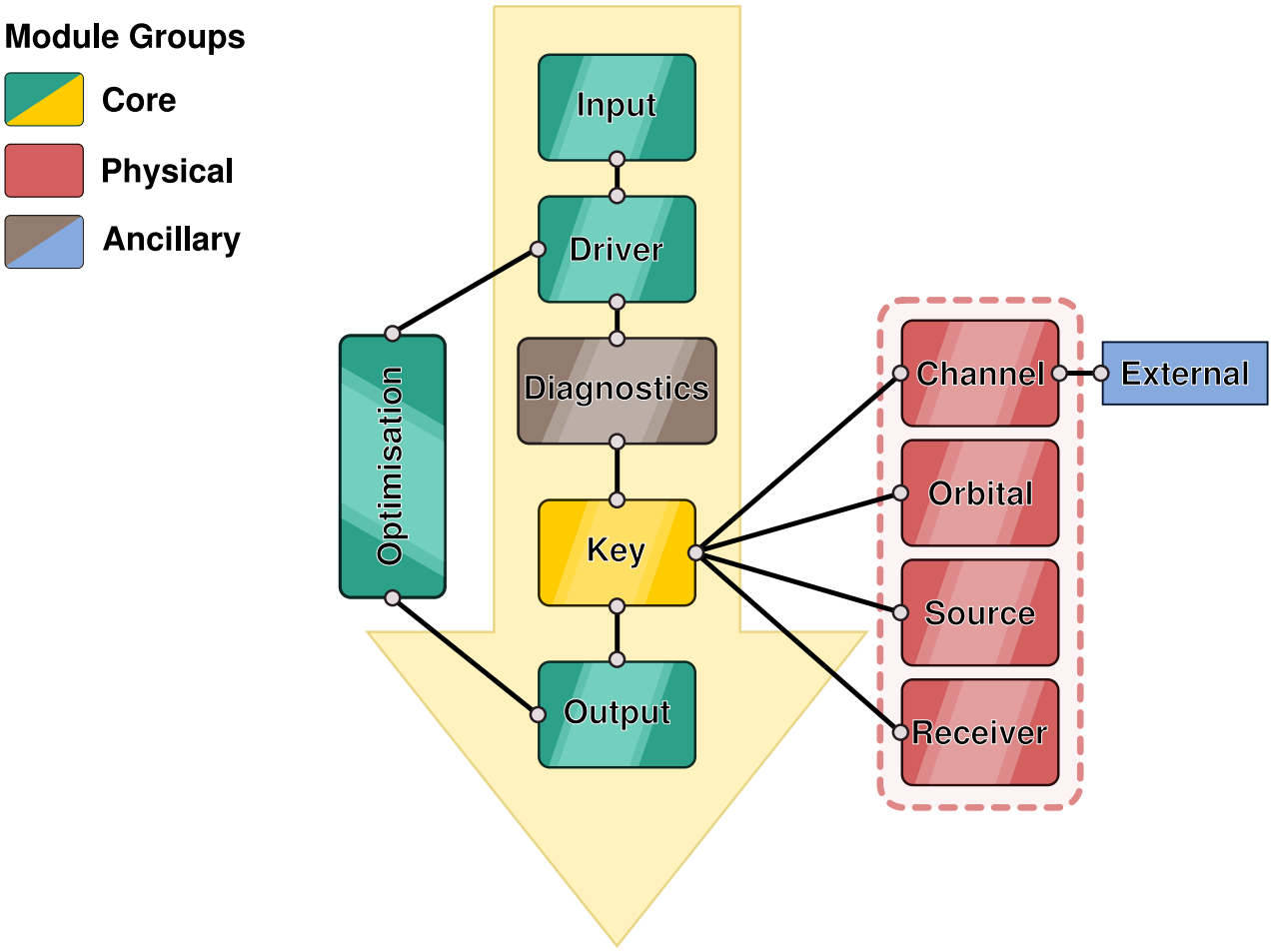

Figure 1: A flowchart showing the proposed modules, their connections, and the general calculation path. The Input and Driver modules control the modelling and analysis, determining the options and data to be used. The Key module forms the heart of the software and has sub-modules that produce the data that can be fed into a Key Length or Rate estimation procedure. The Key Length estimation will incorporate the finite block size statistics that is a result of the limited amount of time available during an passage over the optical ground station (OGS) by the satellite. Alternatively, for static terrestrial installations, this could be determined by operational constraints such as the key refresh interval or buffer lifetime. The optical channel characteristics can be determined by built-in functions or can accept external input if desired, e.g. more sophisticated channel models are available, e.g. AOtools or MODTRAN. For space applications, the orbital sub-module calculates the range and elevation of the satellite with respect to the OGS which is combined with the channel characteristics to determine the channel loss. Source and Receiver sub-modules define the signal states for given protocol parameters and the detector characteristics which together with the channel loss gives the sifted key rates and statistics. This is fed into a key length and rate estimator that then outputs the amount of secret key that could be extracted from the overpass. An optimisation routine varies the protocol parameters (basis bias, decoy state intensities and probabilities) to determine the maximum achievable secret key for a given overpass and system characteristics. The results are presented by the Output module.

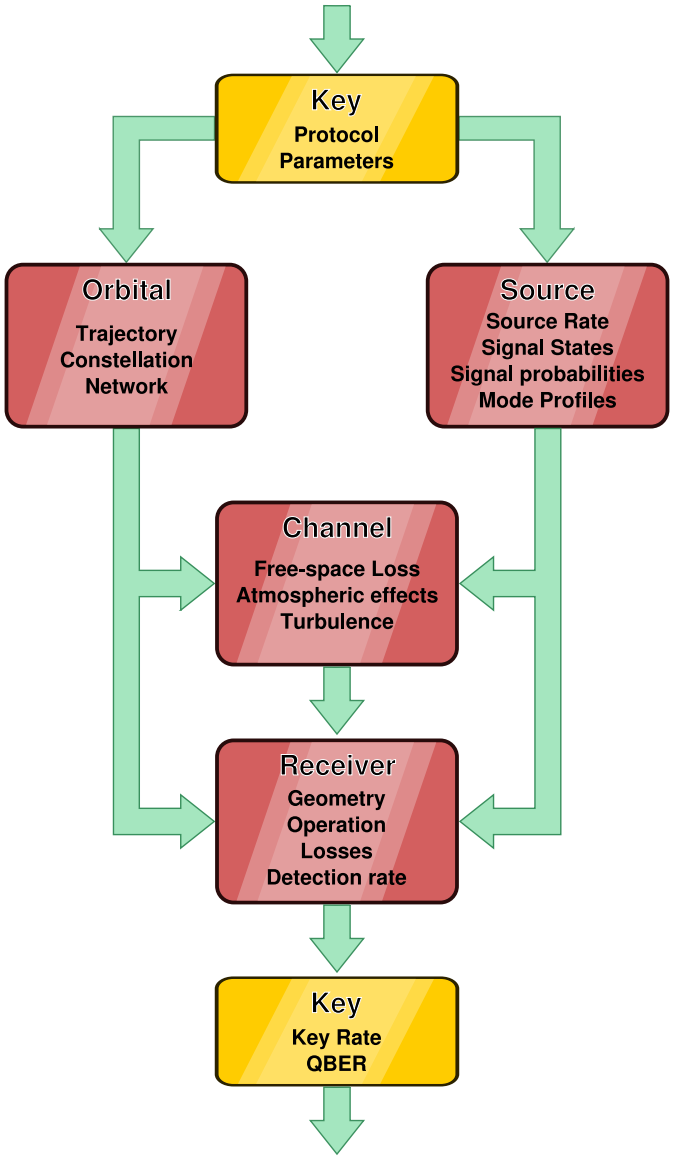

Figure 2: A flowchart outlining detail of the main function of the software, the generation of the key rate. The Key module initiates the calculation by selecting the requested protocol, defining the relevant parameters and then passing them to the Orbital and Source modules which define the signal state and network properties. Both of these modules interface with the Channel and Receiver modules to define the system losses. The Channel module then applies these losses to the signal and the Receiver module returns the detection rate. Finally the Key module determines the secure key length, and associated QBER.

asymptotic case), the Key module will determine various other important metrics related to security including the various error rates associated with a protocol, *i.e.* the Quantum Bit Error Rate (QBER).

The first working version will be programmed to calculate the finite key length for a Weak Coherent Pulse (WCP), with two Decoy States (DS), under the BB84 security protocol [1], which is a Discrete Variable (DV) QKD Prepare and Measure protocol.

As we develop the software we may include more advanced source and detector models such as entangled photon, or single photon states, along with the possibility of Dense Wavelength Division Multiplexing (DWDM). Naturally, the key module will require that the security protocols are updated in tandem with any of these types of developments.

Further, at some future point we may seek to include the option to simulate Continuous Variable (CV) QKD protocols. Or at the least, to ensure that the code architecture is flexible enough to accommodate them by allowing additional data structures to represent CV states, sources, receivers, and the key length/rate estimators.
Version 1.0: DV-QKD, BB84-WCP-DS, asymptotic key rate and finite key length.
Future development: CV-QKD, other Prepare and Measure protocols, Single photon states, and entanglement based protocols.

3.1.2 Input module

The input module will retrieve simulation parameters from the user and seed the calculation. The user will provide the initial simulation parameters in the form of an ASCII text inputfile. This will enable the user to run multiple calculations via batch scripts. When the software is run from a directory, the input module will attempt to locate the input file in that directory. If it fails it will instead prompt the user to provide the parameters via the command line. After receiving the user specified parameters, the input module will perform sanity checks to ensure that said parameters are reasonable, before passing them on to the main driver module to initialise and run the calculation.

In addition, a secondary input file will be employed, allowing advanced users to fine tune select calculation parameters. The software will again search for this locally when run and default to a predefined set of parameters if this file is not present.

The input parameters themselves will consist of numerical values, logical flags, and keywords.

Once we have a working code, and a finalised input file format, we will consider developing a Graphical User Interface (GUI) as an additional input procedure. Once the software is sufficiently developed that it supports different types of QKD simulations a GUI would need to be sufficiently developed to dynamically determine the parameters to request based on the subsequent parameters received.
Version 1.0: Read input file
Future development: Graphical User Interface (GUI)

3.1.3 Driver module

The driver module will initialise the simulation based on parameters passed from the input module. From this, it will determine the calculation parameters and initialise the data structures to be used by the key module. Depending on how the complexity of the data structures change as we introduce greater functionality within the software (varying protocols, sources and detectors, *etc.*) the driver module may need to interface with the physical systems modules to ensure that the various parameters can be simultaneously applied to a particular system. This could be achieved by maintaining a compatibility lookup table either stored within the driver module, or by creating a separate management module later on.

Once the optimisation module is operational it will be required to algorithmically update the calculation parameters, based on the output, through a loop. Defining an optimisation mode for the driver module functions might make it safer when changing calculation parameters that would otherwise be, preferably, immutable.
Version 1.0: Run a single calculation
Future development: Optimisation mode

3.1.4 Output module

The output module will simply provide an output stream to the user, either straight to the console or in files. Once the optimisation module is operational it will also be required to feed calculation data directly into it. Some optional in-built plotting functionality, using matplotlib, may be useful if sufficient data, or metrics, are produced in a simulation.
Version 1.0: Print to console or file
Future development: Feed into optimisation module, plotting data and metrics

3.1.5 Optimisation module

The numerical optimisation module will work to maximise the length of the finite block secure key, considering multiple parameters, using standard methods [2]. Some experimentation may be required to determine the optimum algorithm for finding a *global* maximum from the parameter space once the code is operational.

We initially aim to optimise over the key rate protocol parameters relating to the source and receiver modules. Next, we aim to be able to include optimisation over the orbital parameters of a single satellite, and finally, potentially, a constellation of satellites.

Version 1.0: Optimise decoy state protocol parameters

Future development: Optimise parameters for key rate from source, receiver, and orbital modules.

3.2 Physical systems modules

These modules will contain code describing the physical components of an end-to-end QKD system. These modules will act as repositories for the various system component models that we include, and interface with external programs, where necessary, to provide modelling. The functions in each module will be written following a template to allow for easy extension. These modules will likely have some sort of mutual interface, or will at least all interface with the channel module, as the components of a QKD system must be compatible.

3.2.1 Orbital module

The orbital module will enable us to simulate the trajectory and motion of a satellite, and to define a dynamic source-to-receiver geometry as the satellite makes a pass. The elevation angle of the satellite in the sky affects the amount of atmosphere that the signal must propagate through, along with the orbit height. Due to the varying error rate during a satellite pass, it becomes important to define which parts of the signal stream should be kept, and which should be discarded, in order to build the longest, secure key. Hence, it would be insightful to have the capacity to optimise over these parameters.

An interesting future development might be the simulation of a constellation of satellites, to model the network performance, as additional passes have been shown to improve the finite key that is obtained.

Version 1.0: Single satellite source-to-receiver geometry

Future development: Dynamics of a constellation of satellites

3.2.2 Channel module

The channel module will control the loss models and mechanisms particular to each QKD simulation, with particular consideration of atmospheric models for satellite based QKD systems and horizontal turbulence models for AirQKD systems.

We plan to use the proprietary software MODTRAN [3] for atmospheric modelling, which has a python Application Programming Interface (API) that the module will connect with.

Similarly, the turbulence modelling will likely require the module to connect with an external software, such as AOtools [4] (a python based software).

For portability, the module will also support simple, parameter based models for users that either do not have licences for the proprietary software(s) or do not require such advanced modelling in their simulations.

Version 1.0: Interface with MODTRAN

Future development: Interface with additional software

3.2.3 Source module

The source module will contain functions describing various signal state distributions. Initially we will consider only the polarisation degree of freedom, along with the mean photon number, for protocols that employ Weak Coherent Pulses with Decoy States (WCP-DS). More advanced source distributions might include modal structure in any of the spectral, spatial and temporal domains. Including this kind of structure will also require consideration of the receiver and how the propagation effects might destabilise that structure and disrupt the signal, which would be modelled in the channel module.

Entangled photon, and single photon sources could also be developed along with the relevant protocols and receivers.

Version 1.0: Polarisation controlled WCP

Future development: Modal structure, entangled photons

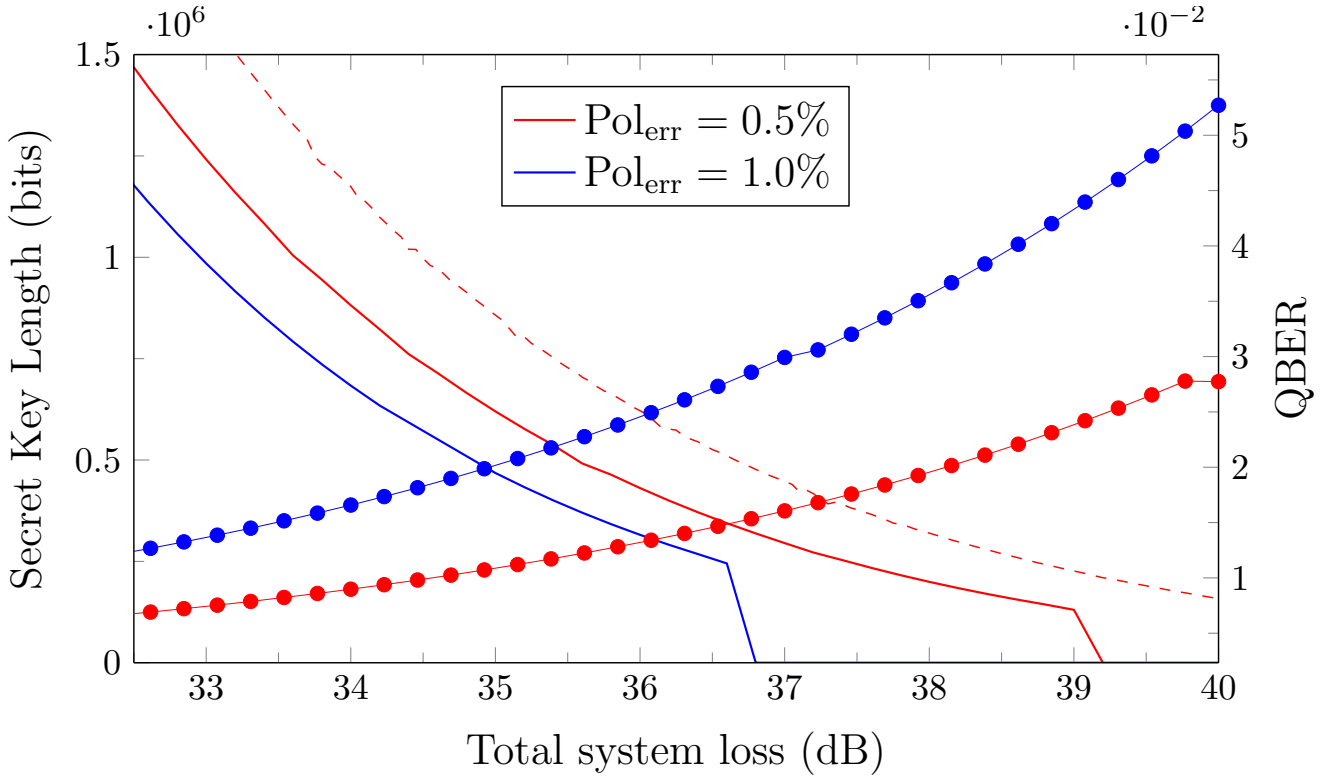

Figure 3: Example Output Data. Here is an example of the type of output data that we should expect to be able to produce from the modelling software. These results come from test code produced during the development of the core modules. The output data has been plotted with an external program. For different transmission efficiencies, indicated by the source-receiver loss at zenith, we plot the total secret key length (solid lines) that can be obtained from a single overpass (altitude $500km$, maximum elevation $90°$), optimised over protocol parameters, as well as the average QBER for the entire block (circles). The source rate was $100MHz$ and the combined dark count and background rate was 10^{-7} per pulse. The different curves represent the effect of different intrinsic source QBERs. For reference, the key length under the asymptotic parameter estimation assumption is also plotted for source error of 0.5% (dashed line).

3.2.4 Receiver module

The receiver module will contain functions describing the geometry, operation, and associated losses for various detection systems. These functions will be used by the key module to provide count statistics for detection events and to determine the key rate, or length.

A suitably advanced receiver model must include the many loss mechanisms and considerations, not limited to the collection area, including obscuration loss, internal losses, basis misalignment error, dark count rate, jitter, and dead-time. Ideally, a ground based receiver should account for the local environment by including background light, information on lunar phases and cloud reflectivity for example.

Once we have developed/included a sophisticated turbulence model, adaptive optics could be modelled to correct signal aberrations.

Once the capacity to select the signal state wavelength has been developed we may include advanced source/detection setups, such as Dense Wavelength Division Multiplexing (DWDM).

Version 1.0: Count statistics

Future development: Adaptive optics, DWDM

3.3 Ancillary functions modules

The ancillary functions modules are superfluous to the operation of the main software, but will provide improved functionality for later releases.

3.3.1 Diagnostics module

The diagnostics module will be developed to accommodate analysis functions such as regression, self-consistency testing and benchmarking. These tools will be particularly useful when additions are made to the software, ensuring that the core functionality is not compromised. Any new functions or modules should be designed with testable outcomes in mind.
Version 1.0: Analyse system performance
Future development: Regression testing, benchmarking

3.3.2 External programs

The software will be designed to have the capability to interface with external, python compatible software. We expect this to primarily involve advanced modelling tools relating to atmospheric and turbulence models (see Sec. 3.2.2). Interfaces will have to be custom designed for each new external software.
Version 1.0: Inclusion of MODTRAN
Future development: Turbulence modelling software

4 Data structures

The primary data structures, those used by various modules, will be built using the python class function. This will ensure robust structures, which can be initialised and destroyed as required, and enable the use of immutable parameters and attributes. The data structures will be custom defined to represent and store different sets of parameters and calculated values (source, receiver, key, *etc.*) which can then be edited locally as the software develops, reducing the amount of edits required when adding (or removing) functionality. The software will control the way in which the data structures are populated via a series of logical flags, keywords and values. This will ensure that any model parameters not associated to a specified system are not erroneously invoked.

5 Development notes

The software we release will only contain modules relevant to Discrete Variable QKD (DV-QKD), however the design will allow for the seamless extension to Continuous Variable QKD (CV-QKD). This will require the development of distinct source, receiver and key modules for CV-QKD which can then be swapped with, or between, their DV-QKD counterparts such that the overall structure remains unchanged. The current development plan only includes effects due to system imperfections, hacks, and side channels in a very limited sense. We are effectively assuming that we have perfect states and measurements, and simply accounting for effects due to misalignment. This could be improved by developing functions within the Source, Receiver and Key rate modules which can model effects such as non-orthogonal polarization measurements, and spectral or timing leakage for example. Similarly, we do not presently plan to include the loss of secret key arising from the classical authentication process. This could easily be included as a function within the Key module. As all of these effects are strongly dependent upon the specifics of the system being modelled we leave their development up to the user. We may extend the Key module to incorporate, or interface with, existing QKD protocol software such as that developed by the University of Waterloo [5]. The contents of this section may be subject to change as the software development progresses.

6 Projected timeline

We currently have two milestone software releases planned, here we list the projected software updates and additions for those releases.

6.1 SatQuMA v1.0 Q2 2021

SatQuMA v1.0 will be the first working version that can calculate key lengths for a limited set of circumstances. This release will see an updated documentation package with a changelog, descriptions of core modules, and user instructions. We may revise the development timeline at this point.

Module	Update
Core	Working version, may be non-interactive.
Orbital	Generate Source-Receiver geometry for limited case.
Source	Polarisation based signal states for weak coherent pulses.
Channel	Option to use MODTRAN input and elevation dependant transmission/ turbulence data.
Receiver	Generate count statistics. Will have options for DCR, Background, AP, detector efficiency, basis bias, measurement error, dead time (low loss channels).
Key	Return finite block size key length for weak coherent pulse, 2 decoy state algorithm.
Optimisation	Optimises protocol parameters for fixed input of orbit, system performance.

Table 1: List of SatQuMA software goals for version 1.0.

6.2 SatQuMA v2.0 Q4 2021

SatQuMA v2.0 will offer a refinement of the QKD simulations with an extended capability. The accompanying documentation will be updated, as will the changelog. The timeline will be revised at this point.

Module	Update
Core	GUI interface, more simulation options including event-by-event output.
Orbital	Constellations, interface with STK
Source	Add options for Single Photon Source, different wavelengths (e.g. 1550nm), new protocols (RFI).
Channel	Add in background light (night-glow, Moon), wavelength dependence, update turbulence module.
Receiver	Include adaptive optics option (with channel turbulence update)
Key	Add in new sources and protocols.
Optimisation	Include system parameters and orbits. Possibly constellations?

Table 2: List of SatQuMA software goals for version 2.0.

References

[1] 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, Feb 2014.

[2] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY: Springer, 2nd ed., 2006.

[3] 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," in *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XX* (M. Velez-Reyes and F. A. Kruse, eds.), vol. 9088, pp. 113 – 119, International Society for Optics and Photonics, SPIE, 2014.

[4] M. J. Townson, O. J. D. Farley, G. O. de Xivry, J. Osborn, and A. P. Reeves, "AOtools: a Python package for adaptive optics modelling and analysis," *Opt. Express*, vol. 27, pp. 31316–31329, Oct 2019.

[5] P. J. Coles, E. M. Metodiev, and N. Lütkenhaus, "Numerical approach for unstructured quantum key distribution," *Nature Commun.*, vol. 7, p. 11712, May 2016.