Automatic Generation of Optimal Quantum Key Distribution Protocols

Walter O. Krawec
Iona College
715 North Ave.
New Rochelle, NY 10801
walter.krawec@gmail.com

Michael G. Nelson
Iona College
715 North Ave.
New Rochelle, NY 10801

Eric P. Geiss
Iona College
715 North Ave.
New Rochelle, NY 10801

ABSTRACT

Quantum Key Distribution (QKD) allows two parties to establish a shared secret key secure against an all-powerful adversary. Typically, one designs new QKD protocols and then analyzes their maximal tolerated noise mathematically. If the noise in the quantum channel connecting the two parties is higher than this threshold value, they must abort. In this paper we design and evaluate a new real-coded Genetic Algorithm which takes as input statistics on a particular quantum channel (found using standard channel estimation procedures) and outputs a QKD protocol optimized for the specific given channel. We show how this method can be used to find QKD protocols for channels where standard protocols would fail.

CCS Concepts

•Security and privacy → Cryptography; •Computing methodologies → *Search methodologies;*

Keywords

Quantum Key Distribution; Genetic Algorithm

1. INTRODUCTION

Today, key distribution between two parties is accomplished using public key cryptography. Unfortunately, all such schemes rely on unproven computational assumptions. Furthermore, this is a necessity, as it is mathematically impossible to construct a public key system secure against an unbounded adversary using *classical* means alone.

Quantum Key Distribution (QKD) protocols, however, by utilizing quantum resources, allow two parties Alice (A) and Bob (B) to establish a shared secret key which is secure against even an all powerful adversary Eve (E). Such a key could, for instance, be used with one-time-pad encryption (which is information theoretically secure).

One of the very interesting, and useful, properties of QKD protocols is that any attack against them must be active (one cannot copy quantum bits to attack later). Furthermore, any attack must create some *noise* in the channel which may be measured by the two parties. The more invasive an attack, the more the adversary learns, but the more noise she creates in the channel. If the noise is low enough, one may distill a secure key. If the noise is "too high" one must simply abort. The reader is referred to [11] for a general survey of QKD protocols.

Typically one constructs a QKD protocol and then mathematically derives bounds on its maximally tolerated error rate. For instance, the BB84 protocol cannot tolerate more than 11% error before the parties must abort [10] (anything less than 11% is acceptable, though with higher noise you may have to run the protocol longer to get a secure key of sufficient length).

In this paper, we take the opposite approach: instead of designing a QKD protocol and then determining which channels (e.g., noise levels) it is secure operating over, we start with a fixed channel and output, using a real-coded genetic algorithm (GA), an optimized QKD protocol to run over it. We envision the following scenario: two parties, who purchased quantum equipment, first run a *channel estimation* procedure to estimate the noise in the channel in a variety of ways (to be described later). From this, they then insert these statistics, along with a description of their hardware's capabilities, into our GA. Our GA will then determine an optimal QKD protocol, based on the limitations of their hardware (or other user-defined restrictions), and output the settings that must be used. Of course, there are some channels where QKD is impossible (e.g., entanglement breaking channels [1]). Thus, for these channels, our algorithm will report that no protocol could be found.

In this paper, we will construct a new GA which outputs either a protocol which is guaranteed to be secure over a particular channel (given as input by the user), or to output an "error" of some sort if it cannot be done.

We stress that QKD equipment is currently in production today and has even been used in several practical instances [11]. This is very much a *real-world problem.* Our algorithm has the potential to make such devices more practical; instead of having to abort if a particular channel doesn't allow for, say, BB84 style encoding, one may try to search for a new protocol specific to the particular observed quantum channel statistics. Furthermore, our algorithm is general enough to take into account potential limitations in the abil-

GECCO'17, July 15-19, 2017, Berlin, Germany
ACM ISBN TBA.
DOI: 10.1145/1235

ities of one or both users' hardware. It is also potentially a very useful tool for theorists, allowing researchers to investigate the necessary resources required for QKD over certain channels.

While several authors have considered the application of evolutionary algorithms to quantum algorithms [4, 8, 12], and to the security analysis of *fixed* QKD protocols [5, 6], we are, to our knowledge, the first to consider their use in constructing optimal QKD protocols over specified, yet arbitrary, channels.

After describing our algorithm, we will evaluate it over symmetric channels where it was proven in [1, 7] that BB84 is optimal thus serving as a useful test-case; indeed for symmetric channels, our algorithm outputs the optimal BB84-style protocol. More interestingly, we also evaluate it on non-symmetric channels. We show two example channels where BB84 would actually fail (the parties would have to abort), yet over which our algorithm finds a QKD protocol which allows for secure key distillation.

2. QUANTUM COMMUNICATION

We will now provide a general introduction to quantum communication. This section is necessarily short due to length constraints; for more information, the reader is referred to [9].

If v is a complex vector or matrix, then we denote by v^T to mean its transpose and by v^* its conjugate transpose. If $z \in \mathbb{C}$, then $\Re z$ and $\Im z$ denote its real and imaginary part respectively, while z^* denotes its conjugate.

Unlike a classical bit which is always in a deterministic state of 0 or 1 and which may be read and/or copied at any time without compromise, a *quantum bit* or *qubit* may be prepared in infinitely many possible states. Furthermore, reading, or *measuring*, a qubit potentially destroys the qubit and alters its state. Also, qubits cannot be copied with unit probability without potentially destroying it.

A qubit is a two-dimensional system and as such may be modeled as an element in a two-dimensional complex Hilbert space $\mathcal{H}$. As all finite n-dimensional complex Hilbert spaces are isomorphic to the vector space $\mathbb{C}^n$, we may view any arbitrary qubit $|\psi\rangle$ as a vector $|\psi\rangle = (\alpha, \beta)^T$. To simplify the math, we further require all quantum states to be normalized vectors; thus $|\alpha|^2 + |\beta|^2 = 1$. The notation $|\psi\rangle$ is read "ket" ψ: it is an example of Dirac's braket notation. Every "ket" represents a quantum state which is simply an n-dimensional normalized complex column vector (for qubits, $n = 2$, however we may consider other quantum systems of higher dimension).

We denote by $\{|0\rangle, |1\rangle, \cdots, |n-1\rangle\}$ to be the *computational "Z" basis* of $\mathbb{C}^n$. The actual choice of basis vectors is largely irrelevant to our work - for simplicity we will simply assume the standard basis: $|0\rangle = (1, 0, 0, \cdots, 0)^T$, $|1\rangle = (0, 1, 0, \cdots, 0)^T$ and so on. From this, we may say that any n-dimensional quantum state may be written as a linear combination of these basis vectors, that is $|\psi\rangle = \sum_{i=0}^{n-1} \alpha_i |i\rangle$, where $\alpha_i \in \mathbb{C}$ subject to the normalization constraint $\sum_i |\alpha_i|^2 = 1$. When $n = 2$, besides the computational Z basis, we will also refer to the X basis (spanned by the two states $|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$) and the Y basis (spanned by the two states $|j_Y\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^j i\,|1\rangle)$, for $j = 0, 1$).

Measuring a quantum state (in some ways the quantum analogue of reading a classical bit) is a probabilistic process causing a qubit to collapse to the observed state. Understanding the exact mechanism of this is not too important to understand our work in this paper, so we do not go into detail. The reader should be aware, however, that one may measure in any basis - if the preparation choice and measurement choice are the same basis, the result is deterministic; otherwise the measurement outcome is random.

As mentioned, every quantum state is a vector living in some Hilbert space. Often we wish to consider two or more states in union. In our case, we will wish to consider A, B, and E's systems along with various auxiliary systems. In general, we may have n different quantum states each living in a Hilbert space $\mathcal{H}_1, \cdots, \mathcal{H}_n$. Let $d_i = \dim \mathcal{H}_i < \infty$ (all systems we consider in this paper are assumed to be finite dimensional, which, for the problem we are considering, is an assumption made without loss of generality). Then, the joint state is modeled as a vector living in the Hilbert space $\mathcal{H} = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_n$ where, given states $|\psi_i\rangle \in \mathcal{H}_i$, then $|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle \in \mathcal{H}$. If $|\psi\rangle = (\alpha_1, \cdots, \alpha_m)^T$, then the tensor product $|\psi\rangle \otimes |\phi\rangle$ is simply $(\alpha_1 |\phi\rangle, \cdots, \alpha_m |\phi\rangle)^T$ (this may then be repeated for three or more systems). Often we will write $|\psi\rangle |\phi\rangle$ or $|\psi, \phi\rangle$ to mean $|\psi\rangle \otimes |\phi\rangle$ (and similarly for three or more systems). Notice that $\dim \mathcal{H} = \prod_i \dim \mathcal{H}_i = \prod_i d_i$.

Besides measuring a quantum state $|\psi\rangle$, which causes it to irreversibly collapse to the observed state, one may also evolve a quantum state by applying a unitary operator. An operator U is unitary if $UU^* = I$, where I is the identity operator. Since $|\psi\rangle$ may be modeled as a vector, the resulting state, after applying unitary operator U, is simply the vector produced by the matrix multiplication: $U|\psi\rangle$.

For every $|\psi\rangle$, there is a corresponding $\langle\psi|$. In our case, $\langle\psi|$ is simply the conjugate transpose of $|\psi\rangle$; that is $\langle\psi| = (|\psi\rangle)^*$. It is therefore a row vector. We denote by $\langle\phi|\psi\rangle$ to be the inner-product (or dot product) of the two vectors $|\phi\rangle$ and $|\psi\rangle$.

A state represented by a vector $|\psi\rangle$ is called a *pure* state. However, often we want to model statistical ensembles of pure states. This may be done using the density operator formalism. For instance, if we have a source which prepares quantum states $|\psi_i\rangle$ with probability p_i ($\sum p_i = 1$), then this may be represented as the density matrix: $\rho = \sum_i p_i |\psi_i\rangle \langle\psi_i|$. Note that the multiplication of the vectors $|\psi_i\rangle$ with $\langle\psi_i|$ form a square matrix. Often, to simplify notation, we will write $\psi_{\mathbf{i}}$ to mean $|\psi_i\rangle \langle\psi_i|$. Also, we will write $\mathbf{0}_A$ to mean $|0\rangle_A \langle 0|_A$, where $|0\rangle_A$ is a computational basis state in $\mathcal{H}_A$ (similarly for $\mathbf{i}_X$ for any i and system X).

More generally, a *density operator* is a Hermitian positive semi-definite operator of unit trace. If ρ_{AE} is a density operator acting on Hilbert space $\mathcal{H}_A \otimes \mathcal{H}_E$ (i.e., in our case, it describes a quantum system held by parties A and E), then we write ρ_A to mean the result of "tracing out" E's system, i.e., the partial trace: $\rho_A = tr_E \rho_{AE}$ where, if $\rho_{AE} = \sum_{i,j} \rho_A^{(i,j)} \otimes |i\rangle \langle j|$, then $\rho_A = \sum_i \rho_A^{(i,i)}$.

Given density operator ρ_{AE}, we may consider the von Neumann entropy of the system (the quantum analogue of Shannon entropy). This is denoted $S(AE)$. Since all systems are finite dimensional, let $\{\lambda_i\}_{i=1}^D$ be the eigenvalues of ρ_{AE}. Then $S(AE) = -\sum_i \lambda_i \log_2 \lambda_i$. Furthermore, the conditional von Neumann entropy is denoted $S(A|E) = S(AE) - S(E)$ where $S(E)$ is the von Neumann entropy of $\rho_E = tr_A \rho_{AE}$. Von Neumann entropy is invariant to changes in basis. It is also a continuous function.

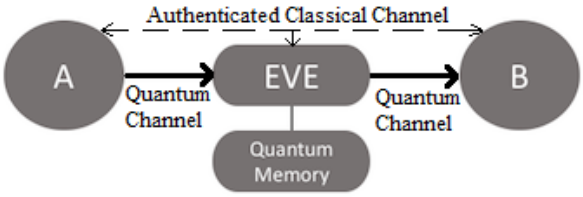

Figure 1: A typical QKD protocol

2.1 Basic Quantum Key Distribution

We consider one-way discrete variable QKD protocols in this paper. Here, a source "Alice" (A) prepares qubits randomly and sends them to a receiver "Bob" (B) who then measures the qubit in a randomly chosen basis. An all-powerful attacker "Eve" (E) sits between the users, intercepts all sent qubits, and probes them, entangling them with her perfect quantum memory, before forwarding the qubit to B. This stage of the protocol is called the *quantum communication stage* and is repeated over numerous iterations. Based on A's preparation choices, and B's measurement outcomes, they will distill a *raw key* - a string of classical 0's and 1's which is partially correlated (E's attack induces some errors) and partially secret (E may have some information on it). See Figure 1.

Besides the quantum communication channel, there is also an authenticated classical channel connecting A and B. On this channel, E may only listen but not send. Thus, following the quantum communication stage, A and B will use some of their measurement results and preparation choices to estimate the noise in the quantum channel - more noise equates to a more invasive attacker in the worst case. If the noise is "low enough" they proceed to run a preprocessing stage (this is optional) [10] followed by an error correcting (EC) protocol and a privacy amplification (PA) protocol. This is all done over the authenticated channel (thus leaking more information to E). The reader is referred to [11] for more information on how EC+PA are used; preprocessing strategies will be discussed in more detail in a later section here.

In this paper we will consider *collective attacks* whereby E performs the same (possibly probabilistic) attack operation each iteration of the quantum communication stage. For the protocols we consider in this paper, security against collective attacks is sufficient to prove security against arbitrary general attacks [2].

In the case of collective attacks, E's attack may be modeled as a unitary operator U acting on the traveling qubit and E's private ancilla (both modeled as finite dimensional Hilbert spaces). Without loss of generality, we may assume E's memory is cleared to some zero state $|0\rangle_E$ and may write U's action on basis states as follows:

$$U|0,0\rangle_{TE} = |0,e_0\rangle + |1,e_1\rangle, \quad U|1,0\rangle_{TE} = |0,e_2\rangle + |1,e_3\rangle \tag{1}$$

where the $|e_i\rangle$ are arbitrary states in E's quantum memory (i.e., these $|e_i\rangle$ are complex vectors). The subscript "T" is used to denote the *transit space*: the two-dimensional Hilbert space modeling the qubit. Unitarity imposes various conditions on these states which will be important later. The above definition is sufficient to describe U's action on any arbitrary qubit due to linearity.

It was shown in [3], that, if N is the size of the raw key (in bits), then, following EC+PA, A and B may distill a secret key of size $\ell(N) \leq N$ bits, where:

$$\lim_{N\to\infty} \frac{\ell(N)}{N} = \inf_{U\in\Gamma}[S(A|E) - H(A|B)]. \tag{2}$$

Above, $H(A|B)$ is the conditional Shannon entropy (easily computed) and $S(A|E)$ is the quantum von Neumann entropy (which is more difficult to compute as A and B do not know exactly which attack E used, and E is a quantum system). The infimum is over all unitary operators U which E could have applied which induce the observed statistics (e.g., the observed error rates). We take the infimum as we assume the worst case that E chose an attack which results in the smallest secret key size $\ell(N)$. The above ratio is called the *Devetak-Winter key rate* in the *asymptotic scenario*. Clearly, the closer it is to one, the better for A and B as they are able to use more of their raw key towards their secret key; when the ratio is zero, then the users must abort (no secure key may be distilled).

2.2 Our Goal

We envision the following scenario: First, A and B, after connecting their QKD devices to a quantum channel, will perform a *quantum tomography* protocol in order to gain statistics on the channel. This protocol involves A sending to B qubits of the form $\{|0\rangle, |1\rangle, |+\rangle, |0_Y\rangle\}$. B, on receiving a qubit, measures in either the Z, X, or Y bases. A will disclose the exact state she sent, while B will disclose his measurement outcome (over the authenticated classical channel which E may listen to, but not write to). Repeating this process for a sufficient amount of iterations allows the users to estimate the values $p_{i,j}$ where $i \in \{0, 1, +, 0_Y\}$ and $j \in \{0, 1, +, -, 0_Y, 1_Y\}$ which we use to denote the probability that B measures $|j\rangle$ if A sent $|i\rangle$ (conditioning on the event that B choose the correct basis for such a measurement outcome). It turns out (see [7]) that these statistics allow the users to gain a very good understanding of certain key properties of the attack operator U employed by E. Note that, as in [7], we do not consider inaccurate estimates of these quantities - as we are working in the asymptotic scenario, we may perform this process for an arbitrary number of iterations. It should not be difficult to extend our algorithm to the case when these measurements are inaccurate, though we leave that as potential future work.

Following this channel estimation protocol, A (or B) will run our GA to find an optimal QKD protocol specific for this channel and send the protocol information (using the authenticated channel) to the other party (this does not destroy security as we are assuming an all-powerful adversary could have simulated all possible outcomes of the GA and chosen an initial attack which is the most advantageous to her for this protocol - note that we must enforce the channel statistics remain the same after tomography; otherwise, E could alter her attack!). The GA should take into account certain, if any, restrictions on the part of the two users (for instance, maybe B cannot measure in any arbitrary basis, only the Z, X, or Y basis). The GA will output "settings" needed to operate a QKD protocol, based on the given restrictions (hardware restrictions or otherwise). The GA may also output a preprocessing strategy (to be discussed later - though it was shown in [10] that such strategies may improve the key rate). The GA will also output the key-rate

(Equation 2) of the given protocol over the specified channel. Note that, for some channels, QKD is impossible - in this event the GA should output a key-rate of 0.

After the GA specifies an optimal protocol, assuming the key-rate is positive, A and B will carry out the given protocol. They will also, however, randomly choose certain iterations to run the channel estimation procedure (i.e., quantum tomography) in order to ensure that E does not alter her attack (if she does, then A and B should abort as the protocol produced by the GA is optimal only for the specified channel).

The above idea (using quantum tomography, then later running an optimal protocol) was first conceived in [1]; however there, the authors only considered a symmetric channel - they proved that BB84 style encoding is optimal in a symmetric channel and no optimizations could be done to improve its key-rate. In [7], the authors considered arbitrary channels, but only one class of protocol with few variables over which a brute-force-search was performed to find an optimal protocol. Furthermore, neither source considered preprocessing strategies which greatly complicates the optimization process.

3. ALGORITHM DESIGN

Our GA must evolve optimal QKD protocols of a specified type. The protocol type will be provided to the algorithm and will, essentially, consist of mathematical equations describing the operation of the protocol based on an arbitrary attack and certain arbitrary "free" parameters. The attack is outside the GA's control, but these free parameters (which may, for example, describe things such as what type of qubit state to prepare at which probability) specify an actual protocol. A candidate solution, therefore, will be these free parameters. The exact number of parameters, along with their domains, must be provided to the GA through the type specification.

Once potential variables are found for the parameters in a candidate solution, its fitness must be evaluated. This will entail computing the Devetak-Winter key-rate expression (Equation 2). However, this expression requires an actual attack description; i.e., we must know actual vectors $|e_i\rangle$ for E's attack to substitute in to the density operator expression and thus construct an actual matrix from which the von Neumann entropy computation is straight-forward. Unlike in [5, 6], we cannot evolve these operators as we are using fixed, observed, statistics $p_{i,j}$. We will develop a mechanism to construct these vectors such that (1) they induce the correct observed statistics; (2) are physically realizable (i.e., they could be produced by an actual attack); and (3) do not limit E's power - i.e., the vectors we will construct will be sufficient to compute the key-rate in the case that E chooses the most optimal attack allowed by the observed statistics and we do not "miss" an important attack operator with our mechanism.

Finally, once all of this has been established (i.e., we have values for the protocol parameters and E's attack), we need a system that allows these values and vectors to be easily substituted into a given density operator expression. From this, various quantum information theoretic computations must be performed, namely finding the eigenvalues for the von Neumann entropy computation. To achieve this, we designed and implemented a new quantum simulator for this purpose.

3.1 Protocol Type Specification

Our algorithm will find optimal QKD protocols conforming to a particular type or class. For instance, it might be that A and B are using limited hardware in that they can only perform certain quantum operations. Or it may be that they are not limited and can perform any operation allowed by today's technology. Or, perhaps in the future, they will have access to even more powerful technology with greater quantum capabilities. Our goal is to design an algorithm that produces a QKD protocol constrained by the user's capabilities. Thus, the user must specify those capabilities by describing a density operator equation modeling the protocol using certain free parameters.

Note that the conclusion of a single iteration of the quantum communication stage of any discrete variable QKD protocol may be described by a density operator of the form:

$$\rho_{ABE} = \sum_{i,j \in \{0,1\}} \mathbf{i}_A \mathbf{j}_B \otimes \rho^E_{i,j}, \tag{3}$$

where the A and B system represent A and B's raw key while the $\rho^E_{i,j}$ is a density operator describing the state of E's quantum memory in the event A and B's raw key is i and j this iteration. The probability that A and B output a raw key bit of i and j is simply $tr\rho^E_{i,j}$.

The state of E's memory is a function of E's attack (in terms of the vectors $|e_i\rangle$ from Equation 1) and the protocol being used. Of course, the GA's goal is to discover an optimal protocol, thus this state has several "free variables" which must be evolved. These free variables may represent such things as what qubits A sends and with what probability, or what bases B measures in, etc. Thus, the user of our algorithm, when designing a new class of QKD protocol (e.g., one based on certain hardware restrictions), must write out the state of E's memory given that she used an arbitrary unitary operator U and provided that certain free variables will be substituted in later. This is not difficult to do (it is just basic algebra generally). For our evaluations, we considered two classes of protocol: the first where B is limited to measuring only in the Z basis for key distillation purposes (however A is free to prepare any type of qubit she likes); the second has no such restrictions (A may prepare any qubit she likes, and B may measure in any two bases he likes). Most common QKD protocols belong to one or both of these classes.

We call this protocol description a $\Pi - \texttt{Type}$ and it will be one of the inputs to our GA. The description consists of the following information:

1. A list of all free variables $x_i \in \mathbb{R}$ used by this $\Pi - \texttt{Type}$ which our algorithm is allowed to optimize over. These variables are given human-readable names.

2. A list of domains $\mathcal{D}_i$ which those variables live in. That is, $x_i \in \mathcal{D}_i$. We take $\mathcal{D}_i(y)$ to mean that point $z \in \mathcal{D}_i$ closest to y (standard Euclidean distance).

3. Equations $\rho^E_{i,j}$ (in density operator formalism) describing the state of E's quantum memory in the event A and B agree on a raw key of i and j respectively ($i, j \in \{0, 1\}$); see Equation 3. These equations are provided as strings using Dirac notation and are functions of E's attack vectors ($|e_i\rangle$) and also the free variables x_i.

4. Functionality that, on taking input vectors $\texttt{atk} = (|e_0\rangle, \cdots, |e_3\rangle)$ describing E's attack operator along with actual values for the free variables x_i, returns an actual density matrix for each $\rho^E_{i,j}$. Call this function `constructOps` ($\{x_i\}$,

`atk`). In our implementation, we developed a new quantum simulator allowing the user to input density operator equations as strings thus allowing the software to be used by non programmers. The simulator is able to take as input such a string, a list of the free variables (both scalars and attack vectors), along with assignments of variables to values. Having this information it will return an actual density matrix, or the result of certain computations of the matrix (e.g., its entropy).

A $\Pi-$`Type` may be considered a "blue-print" or "template" for a particular type of QKD protocol which the user desires to be optimized. In our implementation (source code to be released freely online), this is implemented as an abstract class `Protocol`. Whenever a user wishes to design a new protocol class, they must simply extend this class and pass it to the genetic algorithm.

3.2 Eve's Attack

We are given statistics $\{p_{i,j}\}$ which are the probability that, if A sends qubit $|i\rangle$, then B measures $|j\rangle$ (these are determined during the quantum tomography stage of the protocol - see Section 2.2). In particular, we are given values $\{p_{i,j}\}$ with $i,j \in \{0,1,+,-,0_Y,1_Y\}$. In order to compute the Devetak-Winter key-rate (Equation 2), we must determine actual vectors for those states $|e_i\rangle$ resulting from E's attack operator U (Equation 1). To do so, we will take advantage of work in [7] to bound certain key parameters.

Let $\Gamma = \Gamma(\{p_{i,j}\})$ be the set of all unitary operators acting on a qubit and E's private quantum memory which induce the observed statistics $\{p_{i,j}\}$. Without loss of generality, we may assume E's ancilla is at most dimension 4 (thus each $|e_i\rangle \in \mathbb{C}^4$). This follows immediately from the Stinespring Dilation Theorem [9].

For ease of notation, we write Q_X to mean the probability of a $|+\rangle$ being measured as a $|-\rangle$ (i.e., $p_{+,-}$) and Q_Y to mean the probability of a $|0_Y\rangle$ flipping to a $|1_Y\rangle$. Furthermore, we will write a subscript Y whenever we mean 0_Y (thus $p_{1,Y}$ is the probability of a $|1\rangle$ being measured as a $|0_Y\rangle$ - in a symmetric attack this should be 1/2, but in arbitrary channels this may not be the case).

It was shown in [7] that the following quantities may be directly computed from these statistics alone:

$$\langle e_i|e_i\rangle\,(\forall i=0,\cdots,3), \langle e_0|e_2\rangle, \langle e_0|e_1\rangle, \langle e_1|e_3\rangle, \langle e_2|e_3\rangle, \quad (4)$$

(both the real and imaginary parts of the above inner products may be computed). Furthermore, the following may be computed:

$$\Re\langle e_0|e_3\rangle, \Re\langle e_1|e_2\rangle. \quad (5)$$

Computing the above 10 quantities involves the simple evaluation of linear functions of the observed $\{p_{i,j}\}$ values. For space reasons we cannot print these equations here, though they can be found in [7] (see Equations 10 - 19 in the arXiv version of that source). For any $U \in \Gamma$, the above quantities are always the same. The only unknown quantities are the imaginary part of $\langle e_0|e_3\rangle$ and $\langle e_1|e_2\rangle$. However, the Cauchy-Schwarz inequality may be used to bound these:

$$\Im^2\langle e_0|e_3\rangle \leq \langle e_0|e_0\rangle\langle e_3|e_3\rangle - \Re^2\langle e_0|e_3\rangle \quad (6)$$

$$\Im^2\langle e_1|e_2\rangle \leq \langle e_1|e_1\rangle\langle e_2|e_2\rangle - \Re^2\langle e_1|e_2\rangle \quad (7)$$

Any valid choice of the above imaginary parts *potentially* describes an attack operator $U \in \Gamma$ (there are other restrictions on these quantities which show up later). In the following, we will have to consider all possible imaginary parts of the above inner-products.

Of course, [7] stopped at this point since that paper was interested in analyzing fixed protocols - we will require actual vectors for $|e_i\rangle$ using this information. We now show an iterative method to construct these vectors.

Fix $U \in \Gamma$ whose action we write as in Equation 1. By the Orthogonal Decomposition Theorem, we may write:

$$\begin{aligned}
|e_0\rangle &= a_0|E_0\rangle \quad (8)\\
|e_3\rangle &= a_3|E_0\rangle + b_3|E_1\rangle\\
|e_1\rangle &= a_1|E_0\rangle + b_1|E_1\rangle + c_1|E_2\rangle\\
|e_2\rangle &= a_2|E_0\rangle + b_2|E_1\rangle + c_2|E_2\rangle + d_2|E_3\rangle,
\end{aligned}$$

where $\{|E_0\rangle, \cdots, |E_3\rangle\}$ is some orthonormal basis of $\mathcal{H}_E \cong \mathbb{C}^4$ (here, $\mathcal{H}_E$ is the Hilbert space modeling E's quantum memory). Without loss of generality, we may assume a_0, b_3, c_1, and d_2 are non-negative real numbers (any phase change may be absorbed into the corresponding basis vector).

By considering the inner-product $\langle e_0|e_0\rangle$ we may solve for a_0. Indeed, we have $\langle e_0|e_0\rangle = a_0^2 \Rightarrow a_0 = \sqrt{p_{0,0}}$ (where we used the obvious identity $\langle e_0|e_0\rangle = p_{0,0}$).

We proceed in a similar manner, using the inner-products of the various $|e_i\rangle$ vectors (Equations 4 and 5), along with the fact that U is unitary, to solve for the various coefficients (in order, top to bottom):

$$\begin{aligned}
a_3 &= \frac{\langle e_0|e_3\rangle}{a_0},\ a_2 = \frac{\langle e_0|e_2\rangle}{a_0},\ a_1 = \frac{\langle e_0|e_1\rangle}{a_0}\\
b_3 &= \sqrt{\langle e_3|e_3\rangle - |a_3|^2},\ b_1 = \left(\frac{\langle e_1|e_3\rangle - a_1^* a_3}{b_3}\right)^*\\
c_1 &= \sqrt{\langle e_1|e_1\rangle - |a_1|^2 - |b_1|^2},\ b_2 = \left(\frac{\langle e_2|e_3\rangle - a_2^* a_3}{b_3}\right)^*\\
c_2 &= \frac{\langle e_1|e_2\rangle - a_1^* a_2 - b_1^* b_2}{c_1}\\
d_2 &= \sqrt{\langle e_2|e_2\rangle - |a_2|^2 - |b_2|^2 - |c_2|^2}.
\end{aligned}$$

Now, if $a_0 = 0$, then in fact $|e_0\rangle \equiv 0$ and so E's memory is really dimension three (or less) in which case we may set $a_3 = a_1 = a_2 = 0$ (i.e., if $|e_0\rangle$ never shows up, then after the decomposition, we may write $|e_3\rangle = b_3|E_1\rangle$ and so on, using - possibly different - orthonormal basis $\{|E_1\rangle, |E_2\rangle, |E_3\rangle\}$). A similar argument follows if $b_3 = 0$ or $c_1 = 0$. Finally, if the quantity inside one of the square-roots in the above expressions is negative, then this violates unitarity of U and so our choice of either $\Im\langle e_0|e_3\rangle$ or $\Im\langle e_1|e_2\rangle$ is invalid. Recall we must consider all possible imaginary parts of these two quantities allowed by Equations 6 and 7 - however not all such values produce legal attacks. Thus, when constructing attack vectors, we consider all imaginary values in those bounds which do not cause a negative value in the square-root equations above.

All that remains to be shown is how to construct basis vectors for $|E_i\rangle$. We claim, without loss of generality, that we may use the standard computational basis: $|E_i\rangle = |i\rangle$ (with $|0\rangle = (1,0,0,0)^T$ and so on). This is a consequence of the fact that von Neumann entropy is invariant to changes in basis. Thus, we may simply apply the change-of-basis operator $V = \sum_i |i\rangle\langle E_i|$ (which is clearly unitary) to any attack $U \in \Gamma$. This will not alter the Devetak-Winter key-rate equation. Algorithm 1 summarizes this entire process.

Algorithm 1 `Gen-Attacks`

Input: `Stats` : Channel Statistics of the form $\{p_{i,j}\}$
Output: Γ: a set containing elements of the form $(|e_0\rangle, \cdots |e_3\rangle)$ where each tuple represents a valid attack E could perform based on the given channel statistics.

Process:
1. $\Gamma \leftarrow \emptyset$
2. Discretize the interval in which the imaginary parts of $\langle e_0|e_3\rangle$ and $\langle e_1|e_2\rangle$ live based on Equations 6 and 7 (note the real parts are computed directly)
3. **Repeat** $\forall$ possible $\langle e_0|e_3\rangle$ and $\langle e_1|e_2\rangle$ in the discretized space:
   1. Compute all inner-products allowed by the quantum tomographic process (Equations 4 and 5)
   2. Compute the following coefficients in order left-to-right: $a_0, a_3, b_3, a_1, b_1, c_1, a_2, b_2, c_2, d_2$.
   3. If one of the above coefficients produce an illegal value (as described in the text) discard this iteration and move to the next possible $\langle e_0|e_3\rangle, \langle e_1|e_2\rangle$.
   4. Otherwise, create vectors $|e_i\rangle$ from Equation 8 using the computed coefficients and standard basis vectors for $|E_i\rangle$. Add the tuple $(|e_0\rangle, \cdots, |e_3\rangle)$ to Γ.
4. **Return** Γ

3.3 The Genetic Algorithm

Now that we have discussed the various foundational subcomponents, we turn our attention to the genetic algorithm itself. It requires as input a $\Pi -$ `Type` and a set of channel statistics `Stats` $= \{p_{i,j}\}$. A candidate solution (CS) is a list of free variables (x_i) (the number of free variables, along with their domains, is specified by the given $\Pi-$`Type`). Specific values for these variables define a QKD protocol as determined by $\Pi -$ `Type` (e.g., they may specify the state of the qubits to prepare). We denote by $\mathcal{D}_i$ the domain in which x_i lives; also recall that we write $\mathcal{D}_i(x_i)$ to mean the number $y_i \in \mathcal{D}_i$ which is closest to x_i. Each x_i is a double precision floating point value.

For selection we use tournament selection with a tournament size of 3. We also carried over the best solution from the previous generation to the next.

Crossover was simple one-point crossover, choosing a random point each time. To mutate a CS, we simply perturb 50% of the (x_i) variables by a small randomly chosen $\epsilon \in [-0.1, 0.1]$. Of course, after the perturbation, we ensure the variable remains in $\mathcal{D}_i$. That is, if we mutate variable i, we set $x_i \leftarrow \mathcal{D}_i(x_i + \epsilon)$. We choose a mutation rate of 75%. These settings seemed to produce very good results (we tested other settings, particularly lower mutation rates of 25% and 50%, however they produced inferior results in our experiments).

Finally, to evaluate the fitness of a candidate solution, we must compute the protocol's key rate (Equation 2) over all attacks permitted by the given statistics. A higher key-rate implies a more fit solution (as such a protocol allows A and B to distill more secret key bits after EC+PA). This fitness calculation is done as follows (for a CS denoted $\{x_i\}$):

Let $\Gamma =$ `Gen-Attacks(Stats)` *and set* `rate` $= 1$ *(this will be the returned fitness value). For each atk*$= (|e_0\rangle, \cdots, |e_3\rangle) \in \Gamma$*, do the following:*

1. *Set* $(\rho^E_{00}, \cdots, \rho^E_{11}) = \Pi-$`Type.constructOps` $(\{x_i\}, atk)$
2. *Compute* $S(AE)$ *by computing the eigenvalues of the matrix:* $\mathbf{0}_A \otimes (\rho^E_{0,0} + \rho^E_{0,1}) + \mathbf{1}_A \otimes (\rho^E_{1,0} + \rho^E_{1,1})$*; also compute* $S(E)$ *by computing the eigenvalues of the matrix:* $\sum_{i,j} \rho^E_{i,j}$*. Set* $S(A|E) = S(AE) - S(E)$.
3. *Compute* $\mathtt{key}_{i,j}$*, the probability that A's raw key bit is* i *and B's raw key bit is* j*:* $\mathtt{key}_{i,j} = tr(\rho^E_{i,j})$ *and use these to compute the Shannon entropy* $H(A|B) = H(\{\mathtt{key}_{i,j}\}_{i,j}) - h(\mathtt{key}_{0,0} + \mathtt{key}_{1,0})$.
4. *Set* `rate` $= \min($`rate`$, S(A|E) - H(A|B))$ *and* ***Repeat*** *at (1) for next atk* $\in \Gamma$

Return `rate`

Note that the computation of the density operators, and the eigenvalues, is all done using our simulator. The entire GA, along with the interaction of each module, is depicted in Figure 2.

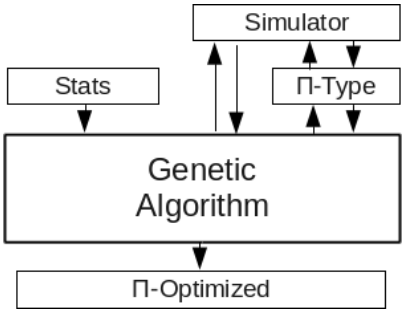

Figure 2: Showing the required input, dependencies, and output of our GA. A set of channel statistics are first inputted into the algorithm along with a $\Pi-$`Type` description. A candidate solution is a list of free parameters (x_i), required to define the QKD protocol specified by $\Pi-$`Type`. Fitness is evaluated by passing those free parameters, along with a set of all possible attack vectors, to the $\Pi-$`Type`. The $\Pi-$`Type` contains functionality to compute density matrices for the various $\rho^E_{i,j}$ - this computation requires the use of our simulator which builds the required matrix. The resulting matrices are passed back to the GA which then uses the simulator to compute the entropy of the system. Finally, the output of the GA is a QKD protocol, of a type specified by $\Pi-$`Type`, optimized for the given channel.

3.4 Preprocessing Strategies

It was shown in [10] that, following the conclusion of the quantum communication stage of a QKD protocol, A and B may improve the key rate of the protocol, by engaging in a "preprocessing" stage immediately before error correction and privacy amplification. While there are infinitely many preprocessing strategies, we will use a generalization of a particular strategy introduced in [10] which operates on each bit of the raw key identically and independently by having A change her raw key bit according to some distribution (to be evolved). In particular, if her key bit is $\mathbf{i}_A$ (for $i \in \{0, 1\}$), after preprocessing, it will be $k_A \in \{0, 1\}$ with probability $p^{A|i}_{k_A}$. Naturally they must be normalized, therefore we have $p^{A|i}_0 \in [0, 1]$, and $p^{A|i}_1 = 1 - p^{A|i}_0$, for each i.

To incorporate the given preprocessing strategy, we add to our candidate solution the values $\{p^{A|i}_{k_A}\}$ with $i, k_A \in \{0, 1\}$ normalized as required. Furthermore, we extend the

crossover operator so that, when called, besides operating on the original $\Pi - \texttt{Type}$ variables as described before, it will also perform one-point crossover on the new vector $(p_{k_A}^{A|i})_{i,k_A}$ (choosing a new crossover point). Additionally, the mutation operator will alter not only the original candidate solution variables, but also 50% of the preprocessing strategy variables by adding a randomly chosen number $\epsilon' \in (-1, 1)$ (and then re-normalizing of course).

What remains to be shown is how the new fitness value is constructed - in particular, how the key rate of the protocol, after a given preprocessing strategy was used, is calculated (furthermore, the fitness must work with any arbitrary $\Pi - \texttt{Type}$ - thus, it can only use density matrices $\rho_{i,j}^E$). As before, we do so by optimizing over all possible attack vectors produced by `Gen-Attacks(Stats)`. Let $\rho = \sum_{i,j} = \mathbf{i}_A \mathbf{j}_B \otimes \rho_{i,j}^E$ be the state of the joint quantum system (for one iteration) before preprocessing (see Equation 3). Then, after A applies her preprocessing strategy, the system becomes:

$$\sigma_{ABE} = \sum_{i,j} \left(\sum_{k_A} p_{k_A}^{A|i} \mathbf{k}_A \right) \otimes \mathbf{j}_B \otimes \rho_{i,j}^E. \quad (9)$$

Therefore, to compute $S(A|E)$ in our fitness function, we replace line (2) of the procedure (see previous section) to instead compute the eigenvalues of the above operator (tracing out B for $S(AE)$ and then A for $S(E)$ of course). To compute $H(A|B)$, we only need the value $\texttt{key}_{a,b}$ which we use to denote the probability that A's raw key is a and B's is b after preprocessing. From Equation 9, this is seen to be: $\texttt{key}_{a,b} = \sum_i p_a^{A|i} tr \rho_{i,b}$, which is easily computed, allowing for the computation of $H(A|B)$ (replacing line (3) of the fitness procedure) and thus the key rate of the protocol.

While this is a very basic preprocessing strategy, it is enough to improve the key rate of certain protocols over very noisy channels. As future work, we are investigating more powerful preprocessing strategies.

4. EVALUATIONS

We evaluate our algorithm using two different protocol types $\Pi - \texttt{Type}$. The first, which we call `One Way Simple` (`OWS`) allows A to send any qubit but B can only measure in the Z basis (though he may measure in other bases for channel tomography of course!). In particular, A will send a qubit in the state $|\psi_j\rangle = \alpha_j |0\rangle + \bar{\alpha}_j e^{i\theta_j} |1\rangle$, where $\alpha_j \in [0, 1]$ and $\bar{\alpha}_j = \sqrt{1 - \alpha_j^2}$ with probability p_j, setting her raw key bit to j. B measures a qubit only in the Z basis. It is not difficult to derive the correct density operator equations placing them in the required form specified by Equation 3. Furthermore, it is easy to see the free variables and their domains: $p_0 \in [0, 1]$ (note that $p_1 = 1 - p_0$ so is not free); $\alpha_0, \alpha_1 \in [0, 1]$; and $\theta_0, \theta_1 \in [0, 2\pi)$. Choices for these parameters, therefore, constitute a candidate solution. This protocol type includes (asymmetric) BB84 [11] as a sub-case.

The second $\Pi - \texttt{Type}$ we evaluate we call `One Way General` (`OWG`) which allows A to send any qubit state she likes (as with `OWS`) and also allows B to measure in any two bases of his choice (or possibly one basis). In particular, if, after his measurement, he observes state $|\phi_i\rangle$ he sets his key-bit to be $\mathbf{i}_B$ (there is no restriction on $\langle \phi_0 | \phi_1 \rangle$). If, after measuring, he does not observe one of those two states, the iteration is considered "inconclusive." B will inform A of all inconclusive events and they are discarded.

	No Preprocessing		Preprocessing		BB84
Noise	Avg.	σ	Avg.	σ	(Opt.)
5%	.497	1.6×10^{-6}	.497	2.1×10^{-5}	.497
10%	.152	1.5×10^{-7}	.155	1.1×10^{-5}	.152
12%	.035	7.8×10^{-8}	.053	3.6×10^{-5}	.035
13%	0	0	.017	2.0×10^{-5}	0
14%	0	0	.0002	4.7×10^{-6}	0

Table 1: Results of running our GA when the channel is symmetric. "Avg." is the average (of 50 trials) fitness (key-rate) of the best solution at the end of 100 generations, while σ is the standard deviation. Showing results with and without preprocessing (note, the preprocessing strategy we use is not guaranteed to produce a higher key-rate for all channels - particularly "low-noise" ones such as 5%). Also showing BB84's key-rate (without preprocessing) which is known to be optimal over symmetric channels (again, without preprocessing). The higher the key-rate, the better. If correct, our algorithm should be able to find a solution that works at 14% noise with preprocessing on; furthermore, without preprocessing, the key-rate (fitness) should match that of BB84. Our algorithm passes both these tests.

4.1 Symmetric Channels

A symmetric channel is one which is parameterized by a single parameter Q representing the noise in the Z, X, and Y bases (i.e., $Q = p_{0,1} = p_{1,0} = Q_X = Q_Y$) with all other "mismatched" statistics being 1/2 (e.g., $p_{0,+} = 1/2$). It was shown in [1, 7] that, without preprocessing, the key-rate of the BB84 protocol cannot be surpassed over a symmetric channel thus this serves as a useful test-case for our algorithm - our GA should be able to (and, as Table 1 demonstrates, does) discover a protocol with the same key-rate of the BB84 protocol. With preprocessing, it was shown in [10] that BB84 can tolerate at least 14% error so our algorithm should be able to find a solution (albeit one with a very low, but positive, key-rate) at such high noise levels. We tested our algorithm with a population size of 100 using 100 iterations over symmetric channels with various amounts of noise. We only considered `OWS` here (there will be no difference between the two $\Pi - \texttt{Type}$'s in a symmetric channel). The results are shown in Table 1. In all our trials that produced a positive key-rate, the optimized protocol outputted by our GA was the BB84 protocol. Thus Table 1 provides evidence for the correctness of our algorithm and approach.

4.2 Arbitrary Channels

While symmetric channels serve as a useful test case, we also evaluated our algorithm on asymmetric channels. For evaluation purposes, we randomly generated attack operators and simulated their statistics - thus these channels could potentially arise under actual quantum attacks. To do so, we chose random unitary operators, based on almost-symmetric attacks, and simulated the tomographic portion of the protocol to find the $\{p_{i,j}\}$ statistics. We evaluated several different channels, however in Table 2 we highlight two interesting ones: namely channels with a high asym-

Channel Number	$p_{0,0}$	$p_{0,1}$	$p_{1,0}$	$p_{1,1}$	Q_X	Q_Y	$p_{0,+}$	$p_{1,+}$	$p_{+,0}$	$p_{0,Y}$	$p_{1,Y}$	$p_{Y,0}$	BB84's key-rate
1	.752	.248	.055	.945	.163	.115	.435	.592	.507	.660	.357	.322	0 (Abort)
2	.790	.210	.099	.901	.121	.283	.581	.381	.373	.337	.453	.499	0 (Abort)

Table 2: Statistics of two sample non-symmetric channels we evaluated. Both channels would cause BB84 to fail (i.e., abort) yet our GA found protocols which could successfully perform QKD over them (see Table 3).

		No Preprocessing		Preprocessing	
Channel Number		Avg.	σ	Avg.	σ
`OWS`	1	.094	.006	.098	.008
	2	.042	1.4×10^{-4}	.049	.005
`OWG`	1	.126	.04	.127	.035
	2	.157	.018	.157	.028

Table 3: Results of running our GA when the channel is not symmetric. Both channels are such where BB84 would normally fail, yet our GA found protocols with positive key-rates.

$p_0 = .563$	Probability that A chooses a raw key bit of 0 (else 1)
$.102\,\|0\rangle + .995e^{4.55i}\,\|1\rangle$	The qubit that A sends if her key bit is 0
$\|0\rangle$	Qubit A sends if her key bit is 1
$.284\,\|0\rangle + .959e^{2.012i}\,\|1\rangle$	After B measures this state, his key bit will be 0
$.982\,\|0\rangle + .189e^{.629i}\,\|1\rangle$	If he observes this state, his key bit is 1
$p_1^{A\|0} = .011$	Probability that A flips her key bit if it was initially 0
$p_0^{A\|1} = .004$	Probability that A flips her key bit if it was initially 1

Table 4: Showing a sample QKD protocol (of type `OWG` over Channel 2) evolved using our GA (after 150 generations). The key-rate of this particular protocol over Channel 2 is 0.134**. Recall that BB84 always fails (i.e., has a key-rate of** 0**) over this particular channel.**

metry in noise and over which BB84 would actually fail. However, as seen in Table 3, our algorithm was able to find a protocol capable of performing QKD over both these channels. For this experiment, we used 150 generations and 25 independent trials. A sample output of our algorithm (i.e., an optimized QKD protocol) is shown in Table 4.

5. CLOSING REMARKS

In this paper, we showed how a GA may be applied to discover optimal QKD protocols, conforming to user-specified restrictions, for arbitrary quantum channels. Furthermore, our algorithm is able to develop a classical preprocessing strategy increasing the key-rate over certain channels. Full source code of our implementation is available online at: `walterkrawec.org/QKDOpt.html`.

Many very interesting open problems remain. In particular, it would be useful to consider inaccurate parameter estimates along with the finite key scenario. Other, more practical attacks, such as multi-photon attacks would also be interesting to study. Finally, it would be interesting to extend the algorithm to work with two-way quantum channels - however, this would require a new method of constructing E's attack vectors.

6. REFERENCES

[1] Joonwoo Bae and Antonio Acín. Key distillation from quantum channels using two-way communication protocols. *Physical Review A*, 75(1):012334, 2007.

[2] Matthias Christandl, Robert Konig, and Renato Renner. Postselection technique for quantum channels with applications to quantum cryptography. *Phys. Rev. Lett.*, 102:020504, Jan 2009.

[3] Igor Devetak and Andreas Winter. Distillation of secret key and entanglement from quantum states. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 461(2053):207–235, 2005.

[4] S. R. Hutsell and G. W. Greenwood. Applying evolutionary techniques to quantum computing problems. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 4081–4085, September 2007.

[5] Walter O Krawec. Using evolutionary techniques to analyze the security of quantum key distribution protocols. In *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion*, pages 171–172. ACM, 2014.

[6] Walter O Krawec. A genetic algorithm to analyze the security of quantum cryptographic protocols. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 2098–2105. IEEE, 2016.

[7] Walter O. Krawec. Quantum key distribution with mismatched measurements over arbitrary channels. *To appear: Quantum Information and Computation*, 2016.

[8] M. Lukac and M. Perkowski. Evolving quantum circuits using genetic algorithm. In *EH '02: Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware*, pages 177–185, 2002.

[9] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, MA, 2000.

[10] Renato Renner, Nicolas Gisin, and Barbara Kraus. Information-theoretic security proof for quantum-key-distribution protocols. *Phys. Rev. A*, 72:012332, Jul 2005.

[11] Valerio Scarani, Helle Bechmann-Pasquinucci, Nicolas J. Cerf, Miloslav Dušek, Norbert Lütkenhaus, and Momtchil Peev. The security of practical quantum key distribution. *Rev. Mod. Phys.*, 81:1301–1350, Sep 2009.

[12] L. Spector. *Automatic Quantum Computer Programming: A Genetic Programming Approach*. Kluwer Academic Publishers, Boston, MA, 2004.