Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 192 (2008) 39–58
www.elsevier.com/locate/entcs

Classical Knowledge for Quantum Cryptographic Reasoning

Vincent Danos [1]

University of Edinburgh, U.K.

Ellie D'Hondt[2]

Vrije Universiteit Brussel & FWO, Belgium

Abstract

We prove that quantum key distribution is secure against several types of attacks within the framework of classical knowledge knowledge for quantum systems, a formal model which was developed in [8]. In particular we rephrase security as a logical property and use meta-logic reasoning on the finite state machine corresponding to the quantum key distribution protocol. While these security issues have been studied before, it is the logical-based approach that is original here.

Keywords: Quantum cryptography, security, formal verification, meta-logic reasoning.

1 Introduction

Quantum computation proved to be very influential in the area of quantum cryptography, with two of the earliest and best-known applications in this area: Shor's algorithm for factoring prime numbers [23] and quantum key distribution (QKD) [2]. In this paper we initiate the investigation of quantum cryptography through formal methods. In particular we exhibit a preliminary analysis of the security of QKD against several types of attacks through meta-logic reasoning. This type of cryptanalysis is much needed since, similar to the situation with classical security protocols, there have been quantum protocols designed and proved to be safe in theory, only for their attacks to be found and their security proofs violated later, either by further theoretical analysis or by taking implementation issues into account. The problem is that while cryptography provides precise mathematical definitions

[1] Email: vdanos@ inf.ed.ac.uk
[2] Email: Ellie.DHondt@vub.ac.be

doi:10.1016/j.entcs.2008.10.026

of particular protocols and their security, these definitions are usually at a low level of abstraction. This makes it hard to analyse different protocols systematically and see how properties evolve when protocols are combined. Also, it is difficult to gauge how well these definitions capture the realities of an actual implementation. Formal methods, on the other hand, are based on well-defined protocol languages which support a systematic presentation of protocols and properties. Most importantly, they provide tool support for automated proof techniques. Such tools are particularly useful for analysis of distributed protocols (and thus cryptography), for which proofs are particularly tedious and error-prone if done by hand.

Formal analysis is a necessity still for QKD, even though the quantum community considers its security analysis as a solved problem. This is because a mathematical proof on the theoretical security of QKD does not guarantee actual security in a practical setting. Hardware issues are not captured in theoretical proofs, and these are important in security, as they can result in backdoors for breaking the system (an example is the so-called timing attack for RSA). Formal methods are more flexibly adapted to these situations and hence can cover a range of different situations.

Concretely we rely on a two-level approach. At the base level the protocol to be analysed is specified in the language of quantum networks [4], a protocol specification language for distributed quantum computations with evaluation rules and a formal semantics. On top of this we have a epistemic-temporal logical framework suitable for meta-logic analysis of the protocol under consideration, first developed in [8]. The idea of developing formal models to reason about knowledge has proved to be very useful for distributed systems [16,15,12]. Epistemic logic provides a natural framework for expressing the knowledge of agents in a network, allowing one to make quite complex statements about what agents know, what they know that other agents know, and so on. Moreover, combining epistemic with temporal logic, one can investigate how knowledge evolves over time in distributed protocols, which is useful both for program analysis as well as formal verification. Whereas the aforementioned earlier developments were defined independent of cryptographic issues, an important realisation is that cryptographic properties can be recast in terms of *knowledge*. Indeed in cryptography many issues, such as security and identification, are all about knowledge. For example, can an agent learn about the presence of an eavesdropper, or can an agent be sure the protocol's goal has been achieved? Reasoning about knowledge is a platform from which one can investigate these and other cryptographic issues in an intuitive yet systematic way.

Research into formal methods for distributed quantum computation is relatively recent and hence not very prolific. Several formal languages for protocol specification were developed, either in the flavour of classical process calculi [17,13,4] or in terms of dynamic logic [1]. Some of these frameworks have been augmented with logical tools for protocol analysis: both [14] and [8] use meta-logic reasoning to assert correctness properties on top of non-logical base language, though only the latter defines epistemic operators. In fact knowledge for quantum distributed systems was defined earlier in [24] independently of a protocol specification language.

However, as we explain in more detail in Sec. 3, a more suitable notion of knowledge, which is essentially *classical*, is the one developed in [8]. Epistemic extensions of models based on dynamic logic are evidently more natural: here there is the work in [21,22], which mostly concerns the analysis of classical security protocols but has been applied to quantum systems as well. Note however that this algebraic 1-level approach is very different from the 2-level model-checking techniques for protocol analysis mentioned above. While any model that allows protocol analysis can in principle be applied in a cryptographic setting, none of the works mentioned above proposes an applicable formal framework for automatic cryptanalysis of quantum protocols.

The structure of this paper is as follows. In Sec. 2, the protocol of quantum key distribution is explained. This is at the same time a leading example to familiarise the reader with distributed quantum networks, a specification language with formal semantics for distributed quantum computations which we use throughout this paper. Next, we discuss the logical framework operating on top of this, including modal operators for time and knowledge, in Sec. 3. We note that though we define knowledge in terms of quantum networks, our definition holds more generally in any agent-based system for describing distributed quantum computations. In Sec. 4 we analyse QKD and its security through meta-logic reasoning, covering several types of attacks. Finally, we conclude in Sec. 5. This paper assumes some familiarity with quantum computation and reasoning about knowledge – for the reader not familiar with these domains, we refer to the excellent [19] and [12]. The present paper is a continuation of earlier work by the authors [4,7,8]. We present only the basics of this earlier work here, and refer the reader to the references cited above for more detailed explanations.

2 Quantum key distribution

The goal of quantum key distribution is to establish a shared secret key between two parties, traditionally called Alice and Bob. There are three versions of the standard QKD protocol, all of which are equivalent [2,10,3]. We will adhere to Ekert's incarnation of the QKD protocol [10] throughout this paper.

The protocol.

QKD is a private key distribution protocol that is secure against eavesdroppers [18]. It relies crucially on the properties of the *Bell state* $|\Phi\rangle$ given by

$$|\Phi_{\mathbf{AB}}\rangle = \frac{|0_{\mathbf{A}}0_{\mathbf{B}}\rangle + |1_{\mathbf{A}}1_{\mathbf{B}}\rangle}{\sqrt{2}} \ , \tag{1}$$

and distributed among agents **A** (Alice) and **B** (Bob) as indicated by the indices. The Bell state has the property that when measured in arbitrary but identical bases by Alice and Bob, both of them are guaranteed to obtain identical measurement outcomes. For the rectangular basis $\{|0\rangle, |1\rangle\}$ this follows from the expression above.

In QKD in particular we also measure in the diagonal basis $\{|+\rangle, |-\rangle\}$. When expressing $|\Phi\rangle$ with respect to this basis we obtain

$$|\Phi_{\mathbf{AB}}\rangle = \frac{|+_{\mathbf{A}}+_{\mathbf{B}}\rangle + |-_{\mathbf{A}}-_{\mathbf{B}}\rangle}{\sqrt{2}} \ , \tag{2}$$

from which follows that also in this basis Alice and Bob will obtain identical measurement outcomes. QKD is based on this property, along with the fact that no-one — including Alice and Bob — can predict which of both measurement outcomes, 0 (when collapsing to $|0\rangle$ or $|+\rangle$) or 1 (when collapsing to $|1\rangle$ or $|-\rangle$) they will obtain.

While the above explanation suffices quantum-mechanically, we need a more formal notation to investigate the protocol computationally. In our case we denote the full specification of one step of the protocol as follows.

$$\begin{aligned} QKD = &\mathbf{A}(a) : \{1\}.[(\mathsf{c}!a)(\mathsf{c}?b).M_1 H_1^a] \\ &|\mathbf{B}(b) : \{2\}.[(\mathsf{c}?a)(\mathsf{c}!b).M_2 H_2^b] \\ &\| E_{12} \ . \end{aligned} \tag{3}$$

We call such a specification a *network*. The concept of distributed quantum networks, as well as the accompanying syntax and semantics, was defined in [4,6]. The syntax for networks is a mix of classical process calculi syntax, the measurement calculus (an assembly language for one-way quantum computations [5]), and newly defined syntax for notions particular to the setting of distributed quantum computations. For classical process calculi concepts we use familiar notation: concurrency $|$ and classical message receiving $\mathsf{c}!$ and sending $\mathsf{c}?$. Note that agents in a network need to have different names, since they correspond to different parties that make up the distributed system. In other words, concurrency comes *only* from distribution; we do not consider parallel composition of processes in the context of one party. The measurement calculus is used for local quantum operations: in the above, measurement of the i-th qubit M_i, Hadamard operation on the i-th qubit H_i; note that operations are read right to left. We overload that notation slightly here by writing H_i^a, which means that execution of the Hadamard gate is conditioned on the value of a Boolean variable a (if $a = 0$, do nothing, otherwise apply H). This is also reflected in the name of the agents, for example $A(a)$ means that A is parameterised by this Boolean variable a. Finally, we need to add syntax for shared quantum resources such as the Bell state in the above. These resources are supposed to be available to each agent at the start of the computation, and because of entanglement cannot be incorporated in each agent's specification. The notation we use is $\| E_{12}$, which should be read as *given* the Bell state over qubits 1 and 2. Furthermore, our agents are *typed*. Agent types, denoted in curly braces, specify how quantum resources are distributed amongst agents. In the above, the fact that Alice owns the first qubit of the Bell state is denoted by $\mathbf{A}(a) : \{1\}$, likewise Bob owns the second as specified by $\mathbf{B}(b) : \{2\}$. So to summarise, one can read the network QKD as follows. At the network level, we have two agents $\mathbf{A}$ and $\mathbf{B}$, acting in parallel on a shared quantum state given by E_{12}. At the agent level, agent $\mathbf{A}$

(**B**) is parameterised by a Boolean variable a (b) and owns qubit 1 (2) , to which it applies the local quantum operation $M_1 H_1^a$ ($M_2 H_2^b$). After this **A** and **B** exchange the values of a and b by classical message passing.

A further part to the protocol, which is at the moment not captured in our network definition above, is that Alice and Bob only keep their measurement outcomes after checking that $a = b$. Only in this case it is guaranteed that Alice's measurement outcome, which we denote s_1, equals that of Bob, denoted s_2. These values are then kept as part of a secret key shared between Alice and Bob. A secret key of adequate length is established by iterating this protocol many times. We note that we could easily express these extra parts of the protocol in our framework; however, we chose conciseness over completeness, as it is the core functionality of one step of the protocol which we wish to analyse in this work.

Formal semantics.

A formal semantics for quantum networks — small-step, operational and denotational — was worked out in [4,7]. It is only the small-step semantics that is of concern to us here. The small-step transitions for distributed computations essentially describe how agents, and the network with them, evolve over different time steps. It is specified by a set of rules that apply consistently to any network definition. An arbitrary network $\mathcal{N}$ is of the following form,

$$\mathcal{N} = \mathbf{A}_1 : Q_1.\mathcal{E}_1 \mid \mathbf{A}_2 : Q_2.\mathcal{E}_2 \ldots \mid \mathbf{A}_m : Q_m.\mathcal{E}_m \parallel \sigma \ , \tag{4}$$

where each event sequence is a composition of measurement patterns $\mathcal{P}$ and communication primitives `c!`, `c?`, `qc!` and `qc?`(classical and quantum). As a network is executed agents carry out local operations and send out messages, affecting the network quantum state as well as each agent's *local state* Γ_i, which is a classical memory recording measurement outcomes and classical variable bindings. The state of the complete network at a particular point in its execution is captured in a *configuration* C, which is given by a quantum state σ together with a set of agent programs and their states, specifically

$$C = \sigma \parallel \Gamma_1, \mathbf{A}_1 : Q_1.\mathcal{E}_1 \mid \Gamma_2, \mathbf{A}_2 : Q_2.\mathcal{E}_2 \mid \ \ldots \ \mid \Gamma_m \mathbf{A}_m : Q_m.\mathcal{E}_m \ . \tag{5}$$

Note that the the quantum and classical states σ and Γ_i, qubit types Q_i and event sequences $\mathcal{E}_i$ in the above are generally not equal to those in the network definition (4), as it is precisely these elements that change during execution. The initial configuration, however, is in fact almost identical to the network definition, differing only in the instantiation of empty local states $\varnothing$ for each agent.

To state the rules of the semantics we adopt a shorthand notation for agents, as

follows,

$$\begin{aligned} \mathbf{a}_i &= \mathbf{A}_i : Q_i.\mathcal{E}_i \\ \mathbf{a}_i.E &= \mathbf{A}_i : Q_i.[\mathcal{E}_i.E] \\ \mathbf{a}^{-q} &= \mathbf{A} : Q\backslash\{q\}.\mathcal{E} \\ \mathbf{a}^{+q} &= \mathbf{A} : Q \uplus \{q\}.\mathcal{E}[q/x] \ , \end{aligned} \tag{6}$$

where E is any event, and $\mathcal{E}_i$ and $\mathcal{E}_i'$ are event sequences. The small-step rules for configuration transitions, denoted $\Longrightarrow$, are specified below; we give some explanations afterwards.

$$\frac{\sigma, \mathcal{P}(V, I, O, \mathcal{A}) \longrightarrow_\lambda \sigma', \Gamma'}{\sigma \parallel \Gamma, \mathbf{A} : I \uplus R.[\mathcal{E}.\mathcal{P}] \Longrightarrow_\lambda \sigma' \parallel \Gamma \cup \Gamma', \mathbf{A} : O \uplus R.\mathcal{E}} \tag{7}$$

$$\frac{\Gamma_2(y) = v}{\sigma \parallel \Gamma_1, \mathbf{a}_1.\mathsf{c}?x \mid \Gamma_2, \mathbf{a}_2.\mathsf{c}!y \Longrightarrow \sigma \parallel \Gamma_1[x \mapsto v], \mathbf{a}_1 \mid \Gamma_2, \mathbf{a}_2} \tag{8}$$

$$\frac{}{\sigma \parallel \Gamma_1, \mathbf{a}_1.\mathsf{qc}?x \mid \Gamma_2, \mathbf{a}_2.\mathsf{qc}!q \Longrightarrow \sigma \parallel \Gamma_1, \mathbf{a}_1^{+q} \mid \Gamma_2, \mathbf{a}_2^{-q}} \tag{9}$$

$$\frac{L \Longrightarrow_\lambda R}{L \mid L' \Longrightarrow_\lambda R \mid L'} \tag{10}$$

Here $\cup$ denotes the union of outcome maps. Implicit in these rules is a sequential composition rule, which ensures that all events in an agent's event sequence are executed one after the other. The first rule is for local operations, which are specified by patterns in the measurement calculus [5]. We have written the full pattern instead of only its command sequence here to make pattern input and output explicit. Because a pattern's big-step semantics is given by a probabilistic transition system described by $\longrightarrow$, we pick up a probability λ here. The full specification of the small-step rules for patterns can be found in the appendix. Note that an agent changes its sort depending on the pattern's output O. The next rule is for classical rendezvous and is straightforward. For quantum rendezvous, we need to substitute q for x in the event sequence of the receiving agent, and furthermore adapt qubit sorts. The last rule is a metarule, which is required to express that any of the other rules may fire in the context of a larger system. L and R stand for any of the possible left-, respectively right-hand sides of any of the previous rules, while L' is an arbitrary configuration. Note that we might need to rearrange terms in the parallel composition of agents in order to be able to apply the context rule. This can always be done since the order of agents in a configuration is arbitrary. In derivations of network execution, we often do not explicitly write reductions as specified by (10), but rather specify in which order the other rules fire in the context of the network at hand. It is precisely in this last rule that introduces nondeterminism at the network level, that is, several agent transitions may be possible within the context of

a network at the same time.

Execution of QKD.

Let us now apply the formal semantics for networks to the particular case of QKD. This is done by applying the rules of the semantics to the network definition given in (3). By going through the protocol step by step one constructs all configurations that potentially occur during the execution of QKD; it is this configuration space, denoted $\mathcal{C}_{\mathcal{N}}$ for a network $\mathcal{N}$, that we reason upon for cryptanalysis.

The initial configuration for QKD is simply

$$C_0(a,b) = E_{12} \parallel \varnothing, \mathbf{A}(a) : \{1\}.[(\mathsf{c}!a)(\mathsf{c}?b).M_1 H_1^a] \mid \varnothing, \mathbf{B}(b) : \{2\}.[(\mathsf{c}?a)(\mathsf{c}!b).M_2 H_2^b]$$

Here $\varnothing$ denotes the empty local state, i.e. no variables or measurement outcomes are stored initially. Note that configurations are parameterised by the values of a and b.[3] Jumping to the configuration where both agents have carried out their local quantum operation and obtained measurement outcomes j_1 and j_2, we have

$$C_1(a,b) = \mathbf{0} \parallel [s_1 \mapsto j_1], \mathbf{A}(a).[(\mathsf{c}!a)(\mathsf{c}?b)] \mid [s_2 \mapsto j_2], \mathbf{B}(b).[(\mathsf{c}?a)(\mathsf{c}!b)] \ .$$

The network quantum state is now the null state $\mathbf{0}$ and types have disappeared because measurements in patterns are destructive. Obviously we have branching due to varying measurement outcomes, and this with specific probabilities. In the final step of the computation agents have added the values of b, respectively a in their local states; the associated configuration $C_2(a,b)$ is given by

$$C_2(a,b) = \mathbf{0} \parallel [s_1 \mapsto j_1, \overline{b} \mapsto b], \mathbf{A}(a) \mid [s_2 \mapsto j_2, \overline{a} \mapsto a], \mathbf{B}(b) \ ,$$

where $\overline{a}$ and $\overline{b}$ are variable names. Schematically we have two possible structures of configuration trees for QKD, two of each type. These are represented in Fig. 3 (ignore the boxes for now) for $a = b$ or $a \neq b$ respectively. Here we have not written the dependency of each configuration on a and b explicitly to avoid cluttering the picture. Our configuration trees are slightly cruder than the ones taking all steps of the protocol into account because we allow the quantum operations to be carried out in parallel by Alice and Bob. This is because we are interested in the value of the secret key and how an eavesdropper can learn this value, and hence, since the order in which local quantum operations are carried out does not matter,[4] we can make this simplification.

3 Reasoning about knowledge

In this section we explain the notion of knowledge for distributed quantum systems, which was defined earlier in [8]. Our results are phrased in the context of quantum

[3] We could have modelled setting up the values of a and b within our protocol as well; again, we have chosen not to do so for simplicity.

[4] This just follows from the mathematical identity $L_1 \otimes L_2 = (I \otimes L_2)(L_1 \otimes I) = (L_1 \otimes I)(I \otimes L_2)$, where L_1 and L_2 are local operations.

networks, but in fact our notion of knowledge is model-independent. That is to say, any agent-based model for distributed quantum computation would benefit from knowledge as defined below, or slight adaptations thereof.

Facts.

Before we can actually define modal operators for knowledge or time, we need to clarify what the propositions are that these act upon. It is not our intention to define a full-fledged language for primitive propositions; rather, we define these abstractly, and specify the usual rules for combining them with standard logical connectives. We do give some examples of the kinds of properties that we are interested in later on.

Each network $\mathcal{N}$ determines a set of configurations $\mathcal{C}_{\mathcal{N}}$ that can potentially occur during execution of $\mathcal{N}$. An interpretation of $\mathcal{N}$ is a truth-value assignment to all configurations in $\mathcal{C}_{\mathcal{N}}$ for some basic set of primitive propositions θ. Writing $I(C,\theta)$ for the interpretation of fact θ in configuration C, we then have a first logical statement,

$$C,\mathcal{N} \vDash \theta \iff I(C,\theta) = \text{true} \ . \tag{11}$$

The primitive propositions considered usually depend on the network under study, and are currently specified individually for each application. Typical primitive propositions encountered are the following, where we use slightly ad-hoc logical notation,

$$C,\mathcal{N} \vDash (x = v) \iff \exists i.\Gamma_i(x) = v \tag{12}$$
$$C,\mathcal{N} \vDash (x = y) \iff \exists i,j.\Gamma_i(x) = \Gamma_j(y) \tag{13}$$
$$C,\mathcal{N} \vDash (\mathbf{A}_i \text{ has } q) \iff q \in Q_i \tag{14}$$
$$C,\mathcal{N} \vDash (q_1 \dots q_n = \sigma) \iff q_1 \dots q_n = \sigma \tag{15}$$
$$C,\mathcal{N} \vDash (q_i = q_j) \iff \exists \sigma.q_i = q_j = \sigma \ . \tag{16}$$

The first two lines concern classical variable value and variable equality. The third line is about qubit ownership. The last two lines are genuine quantum statements, and need to be used with care, in that they only make sense when pertaining to known quantum states. That is, σ should be seen as a preparation or a state that has just been measured, not as an unknown quantum input such as that appearing in teleportation. The last primitive proposition is especially tailored to unknown quantum states: it states that two qubits named q_i and q_j are equal, though the actual state they are in may be unspecified; this is something we need to be able to express for the teleportation protocol, for example. We also allow functions *init* and *fin* for taking the initial and final values of a variable or quantum state. These formulas are currently defined in an ad-hoc manner.

Composite formulas can be constructed from primitive propositions and the

usual logical connectives $\wedge$, $\vee$ and $\neg$. Concretely, we have the following rules.

$$C, \mathcal{N} \vDash \theta_1 \wedge \theta_2 \iff C \vDash \theta_1 \wedge C \vDash \theta_2 \tag{17}$$
$$C, \mathcal{N} \vDash \theta_1 \vee \theta_2 \iff C \vDash \theta_1 \vee C \vDash \theta_2 \tag{18}$$
$$C, \mathcal{N} \vDash \neg\theta \iff C \nvDash \theta \ . \tag{19}$$

Hence the syntax for facts F is as follows,

$$F ::= \theta \mid F \wedge F' \mid F \vee F' \mid \neg F \ . \tag{20}$$

Knowledge.

The standard approach to knowledge representation in multi-agent systems is based on the *possible worlds* model [16,15,12]. The idea is that there exists a set of worlds such that an agent may consider several of these to be possible. An agent *knows* a fact if it is true in all the worlds it considers possible; this is expressed by epistemic modal operators acting on some basic set of propositions. The flexibility of this approach lies in the fact that there are many ways in which one can specify possibility relations. In a distributed system, worlds correspond to global configurations occurring in a particular protocol, and *possible* worlds are determined by an equivalence relation over these configurations. Typically, global network configurations are considered equivalent by an agent if its local view on these configurations is identical.

A first attempt to define knowledge for quantum distributed systems is found in [24]. Therein, two different notions of knowledge are defined. First, an agent i can *classically* know a formula θ to hold, denoted $K_i^c\theta$; in this case the possibility relation is based on equality of local classical states. Second, an agent can *quantumly* know a formula to hold, denoted $K_i^q\theta$. For the latter, the possibility relation is based on equality of reduced density matrices for that agent. The authors argue that K_i^q is an information-theoretic idealisation of knowledge, in that the reduced density matrix embodies what an agent, in principle, could determine from its local quantum state. However, there are two main problems with this approach. The first is that one cannot assume that the reduced density matrix is always known, because in quantum mechanics, observing a state alters it irreversibly. So, knowledge does not consist of possession of a quantum state: it is not because an agent has a qubit in its lab that the agent knows anything about it. Indeed, consider the situation where a qubit has just been sent from **A** to **B**. Then **B** knows nothing about its newly acquired qubit – it is possible, even, that **A** knows more about it than **B** does. The second problem with the above approach is that one loses information on correlations between agents by considering only the reduced density matrix, a crucial ingredient in distributed quantum primitives.

What we need is a proper notion of knowledge, which captures the information an agent can obtain about its quantum state. This includes the following ingredients: first, an agent knows states that it has prepared; second, an agent knows a state when it has just measured it; and third, an agent may obtain knowledge

by classical communication of one of the above. While knowledge of preparation states is automatically contained in the description of the protocol, our notion of equivalence precisely captures the latter two items. As we shall see below, in doing this we find a similar notion as $K_i^c\theta$. Our main argument, then, is that there is no such thing as quantum knowledge in the sense of $K_i^q\theta$; rather knowledge is about classically knowing facts about quantum systems. To stress this we refer to our notion in full as *classical knowledge for quantum systems.*[5]

Hence in order to define knowledge, we need to define an equivalence relation on configurations for each of the agents, embodying what an agent knows about the global configuration from its own information only. We deliberately do not say *local* information here, as, via the network preparation, an agent may also have non-local information under the form of correlations at its disposal. Each agent's equivalence relation has to reflect what an agent knows about the network state, the execution of the protocol and the results of measurements. All classical information an agent has is stored in its local state Γ; this includes measurement outcomes and classical values passed on by other agents. Just like in classical distributed systems, an agent can certainly differentiate configurations for which the local state is different. As for quantum information, an agent knows which qubits it owns, what local operations it applies on these qubits, and, moreover, what (non-local) preparation state it starts out with, i.e. what entanglement it shares with other agents initially. All of the above information is in fact captured by an agent's event sequence in a particular configuration, together with its local state. Therefore, we obtain the following definition.

Definition 3.1 Given a network $\mathcal{N}$ and configurations $C = \sigma \parallel |_i\Gamma_i, \mathbf{A}_i : Q_i.\mathcal{E}_i$ and $C' = \sigma' \parallel |_i\Gamma'_i, \mathbf{A}_i : Q'_i.\mathcal{E}'_i$ in $\mathcal{C}_\mathcal{N}$, we say that agent $\mathbf{A}_i$ considers C and C' to be *equivalent*, denoted $C \sim_i C'$, if $\Gamma_i = \Gamma'_i$ and $\mathcal{E}_i = \mathcal{E}'_i$. For each agent $\mathbf{A}_i$ the relation $\sim_i$ is an equivalence relation on $\mathcal{C}_\mathcal{N}$, called the *possibility relation* of $\mathbf{A}_i$.

Via possibility relations we can now define what it means for an agent $\mathbf{A}_i$ to know a fact θ in a configuration C in the usual way,

$$C, \mathcal{N} \vDash K_i\theta \iff \forall C' \sim_i C : C' \vDash \theta \ . \tag{21}$$

Our choice of equivalence embodies that agents cannot distinguish configurations if they only differ in that other agents have applied local operations to their qubits; neither can they if other agents have exchanged messages with each other. This situation is represented schematically in Fig. 1. While the global network state does change as a result of local operations, an agent not executing these has no knowledge of this, and no way of obtaining it. This is precisely what we capture with the relation $\sim_i$.

[5] Note that is different from the terminology used in [8], where we talk about *quantum knowledge.*

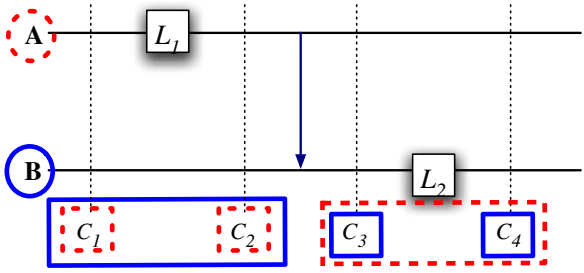

Fig. 1. A schematic event diagram with equivalence classes.

Time.

One typically also wants to investigate how knowledge *evolves* during a computation, for example due to communication between agents. We use the approach of *computational tree logic* (CTL) [11] to formalise time-related logical statements, providing state as well as path modal operators. The reason for this is that, due to the fact that quantum networks typically have a branching structure, we need to be able to express statements concerning *all* paths as well as those pertaining to *some* paths. Concretely, we introduce the traditional temporal state operators $\Box$ ("always") and $\Diamond$ ("eventually") into our model, and combine these with the path operators A ("for all paths") and E ("there exists a path"), as follows [6]

$$C, \mathcal{N} \vDash A\Box\theta \iff \forall\gamma, \forall C' \text{ with } C \overset{\gamma}{\Longrightarrow} C' : C' \vDash \theta \tag{22}$$

$$C, \mathcal{N} \vDash E\Box\theta \iff \exists\gamma, \forall C' \text{ with } C \overset{\gamma}{\Longrightarrow} C' : C' \vDash \theta \tag{23}$$

$$C, \mathcal{N} \vDash A\Diamond\theta \iff \forall\gamma, \exists C' \text{ with } C \overset{\gamma}{\Longrightarrow} C' : C' \vDash \theta \tag{24}$$

$$C, \mathcal{N} \vDash E\Diamond\theta \iff \exists\gamma, \exists C' \text{ with } C \overset{\gamma}{\Longrightarrow} C' : C' \vDash \theta \ . \tag{25}$$

Obviously, we have that any formula with A implies the corresponding one with E, and likewise any formula with $\Box$ implies the corresponding one with $\Diamond$.

Properties.

The properties we wish to express about protocols are constructed from the ingredients above: facts, logical connectives, and modal operators of time and knowledge. Concretely, a property P is generated from the following syntax

$$P ::= F \mid K_i P \mid A\Box P \mid E\Box P \mid A\Diamond P \mid E\Diamond P \ , \tag{26}$$

where the syntax for facts is given in (20).

When investigating knowledge issues in a distributed system, one naturally arrives at situations where one needs to describe formally how knowledge evolves as the computation proceeds. This can be done adequately by combining knowledge operators K_i with the temporal operators defined above. As usual, one needs to

[6] $\overset{\gamma}{\Longrightarrow}$ is the closure of the small-step transition relation $\Longrightarrow$ defined in [8]. That is, we have $C \overset{\gamma}{\Longrightarrow} C'$ if C' can be reached form C by a series of consecutive small-step transitions, specified by the path γ.

proceed with caution when doing this, since it is not always intuitively clear what the meaning of each of these different combinations is. For example, it is generally *not* the case that the formula $A\Box K_i\theta$ is equivalent to $K_i A\Box\theta$. Typically, we want to prove things that are always known by an agent, no matter what branch the protocol follows; this is embodied by the former, and represented schematically in Fig. 2. In the figure the arrows represent the equivalence relation $\sim_i$ and the circled points are those for which θ should hold.

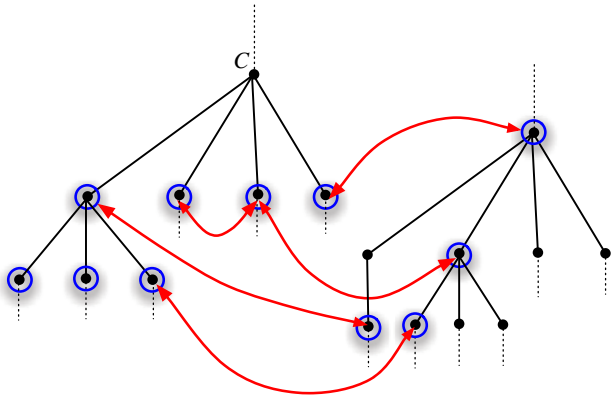

Fig. 2. A schematic representation of $C, \mathcal{N} \vDash A\Box K_i\theta$.

4 Security issues for QKD

With epistemic and temporal notions for quantum networks in place, we are ready to evaluate quantum key distribution and its security from a knowledge-based perspective.

Fig. 3 pictures the equivalence classes of configurations for Alice (dotted boxes) and Bob (boxes) respectively. Equivalence across both trees holds for Alice for a fixed, and for Bob for b fixed. Vertically, each level is discerned by each agent because a local event has occurred, while horizontally, different measurement outcomes ensure the non-equivalence of configurations. Note that at each time step *all* configurations in which say, Alice, has a particular measurement outcome are

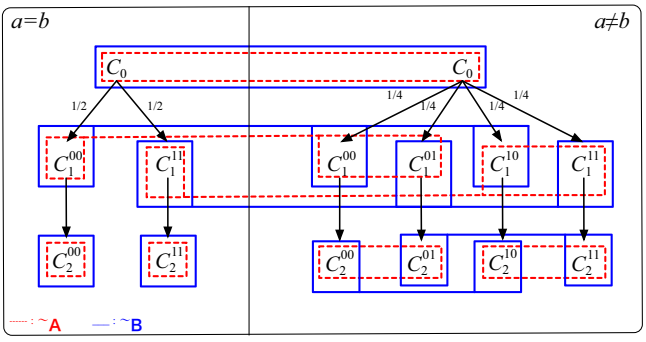

Fig. 3. Configuration trees and equivalence classes.

equivalent, i.e. this works across all configuration trees with the same value for a. More formally, we have the following two equivalence classes of configurations for Alice at level 1, one for each value of s_1,

$$[C_1^{s_1}(a)]_{\mathbf{A}} = \{\forall b, s_2 : C_1^{s_1 s_2}(a,b)\} \ . \tag{27}$$

At the final level Alice and Bob have interchanged the values of a and b and so they can tell the difference between a number of configurations that were equivalent at the previous level. Concretely we move to a set of equivalence classes for Alice as follows,

$$[C_2^{s_1}(a,b)]_{\mathbf{A}} = \{\forall s_2 : C_2^{s_1 s_2}(a,b)\} \ . \tag{28}$$

Bob has analogous equivalence classes $[C_1^{s_2}(b)]_{\mathbf{B}}$ and $[C_2^{s_2}(a,b)]_{\mathbf{B}}$, which are defined as the above but with the roles of a and b and of s_1 and s_2 interchanged.

When $a = b$ Alice and Bob have identical single-configuration equivalence classes at the final stage of the computation. This means that each agent's knowledge is identical at that point, and that we can prove formally that in this case one bit of a secret key has been established,

$$a = b \Rightarrow C_0(a,b), QKD \vDash \ A\Diamond(K_{\mathbf{A}}(s_1 = s_2) \wedge K_{\mathbf{B}}(s_1 = s_2)) \tag{29}$$

In this section we prove that QKD is secure against several types of attacks within our logical framework of knowledge and time. While these security issues have been studied before, it is the logical-based approach that is original here. We head off by showing that eavesdroppers listening in on classical communication channels cannot derive the key. This is of course the core property in making QKD so valuable as an alternative to classical private symmetric secret key establishment protocols.

Listening in on classic channels.

Suppose we have a third agent Eve, who eavesdrops on classical communication between Alice and Bob. Formally, we can model this by adding a third agent **E**, acting in parallel with **A** and **B**, to the QKD network given in (3), as follows,

. (30)

$$\begin{aligned} QKD = &\mathbf{A}(a) : \{1\}.[(\mathsf{c}!a)(\mathsf{c}?b).M_1 H_1^a] \\ &|\mathbf{B}(b) : \{2\}.[(\mathsf{c}?a)(\mathsf{c}!b).M_2 H_2^b] \\ &|\mathbf{E}.[(\mathsf{c}!a, b)(\mathsf{c}?a, b)] \\ &\| E_{12} \ . \end{aligned} \tag{31}$$

Note that Eve has no type since she has no qubits. The idea is that Eve intercepts the messages sent by Alice and Bob, reads them, and sends them along so that the other agents cannot detect her presence. However, neither can she derive any

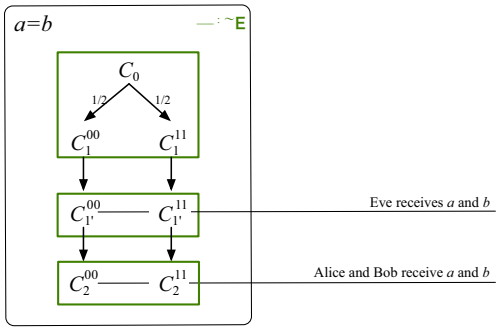

Fig. 4. Eve's equivalence classes when $a = b$.

information about the key. This we know already from straightforward quantum mechanics, but let us prove this now in our framework. By including Eve an extra level is inserted in our configuration trees, between level 1 and 2. Calling this level 1', we thus obtain a number of new configurations $C_{1'}$ that our agents can reason upon. A representation of the $a = b$ case is given in Fig. 4.

The crucial point is that, since the values of s_1 and s_2 remain local throughout the protocol,[7] configurations with different values for these parameters are equivalent for Eve. In particular for $i = 1', 2$ we find the following equivalence classes

$$[C_i(a,b)]_{\mathbf{E}} = \{\forall s_1, s_2 : C_i^{s_1 s_2}(a,b)\}. \tag{32}$$

In other words, Eve can never derive the values of the secret key, since within one equivalence class there will always be configurations corresponding to different measurement outcomes. More formally,

$$a = b \Rightarrow C_0(a,b), QKD \vDash \ A\Box\neg K_{\mathbf{E}}(s_1 = s_2 = v) \tag{33}$$

where v is a value. Note, however, that Eve can derive that in this case measurement outcomes for Alice and Bob are identical.

Measuring the Bell state.

A second possible attack is that Eve intercepts the Bell state before it is distributed among Alice and Bob, measuring it and resending it to Alice and Bob in collapsed form. For simplicity, we assume that Eve always measures in the ordinary basis. This situation would involve changing the specification of the QKD network given in (3), as follows,

$$\begin{aligned} QKD = &\mathbf{A}(a).[(\mathtt{c}!a)(\mathtt{c}?b).M_1 H_1^a(\mathtt{qc}?1)] \\ &|\mathbf{B}(b).[(\mathtt{c}?a)(\mathtt{c}!b).M_2 H_2^b(\mathtt{qc}?2)] \\ &|\mathbf{E}.[(\mathtt{qc}!1)(\mathtt{qc}!2).M_{12}] \\ &\|E_{12}\ . \end{aligned} \tag{34}$$

[7] Local in the sense that they are not sent around directly.

Note that Alice and Bob now have an empty type, as they receive their qubits at the start of the computation from Eve (whom they assume to be a safe quantum channel). The interesting case again is where $a = b$, and in particular $a = b = 1$ since Eve always uses the ordinary basis. In this case Alice and Bob may obtain different measurement outcomes even though they measure in the same basis.[8] Hence they can figure out that an attack has been attempted, and thus that their network is not secure.

In our framework, again we have an extra level of configurations, this time right after $C_0(a, b)$, where Eve measures and sends qubits 1 and 2. However, let us move straight to the final stage of the computation. Now we have the following final equivalence classes for Alice

$$[C_2^{s_1}(1,1)]_{\mathbf{A}} = \{\forall s_0, s_2 : C_2^{s_0 s_1 s_2}\}, \tag{35}$$

where s_0 is Eve's measurement outcome. Indeed, since Eve's and Bob's measurement outcomes are now uncorrelated with Alice's, and they have not communicated these values to Alice, she has no way of discerning between configurations where her outcome s_1 is fixed but theirs is arbitrary. Thus the following statement holds:

$$C_0(1,1), QKD \vDash A\Diamond\neg K_{\mathbf{A}}(s_1 = s_2) \ . \tag{36}$$

While in this case Alice and Bob cannot derive whether or not their protocol was safely executed, they can do so when they exchange their measurement outcomes s_1 and s_2. In this case we have

$$C_0(1,1), QKD \vDash E\Diamond K_{\mathbf{A}}(s_1 \neq s_2) \ , \tag{37}$$

and hence they know without doubt that the security of their network was compromised.

Entangling with the Bell state.

A third possibility is that Eve entangles herself with the Bell state shared by Alice and Bob and operates on her qubit only. This is described by the following network specification, where **A** and **B** are defined as before, O is an arbitrary quantum operation (i.e. a completely positive map), and $|\psi_{123}\rangle$ is an arbitrary 3-qubit state which behaves like the Bell state on the first two qubits[9]

$$QKD = \mathbf{A} \ldots |\mathbf{B} \ldots |\mathbf{E} : \{3\}.O_3 \| |\psi_{123}\rangle \ . \tag{38}$$

Suppose O implements some unitary U acting on qubit 3 only. Since measurement statistics for Alice and Bob are unchanged, in the interesting case where $a = b$ we again have a situation as in Fig. 4, and hence (33) holds: Eve cannot derive the secret key. Next, suppose O implements a measurement of some sort. Only if qubit

[8] This is because we have $|00\rangle = \frac{1}{2}(|++\rangle + |+-\rangle + |-+\rangle + |--\rangle)$, and a similar expression for $|11\rangle$.

[9] That is, the reduced density matrix of $|\psi_{123}\rangle$ on the Hilbert space corresponding to the first two qubits equals the density matrix of the Bell state.

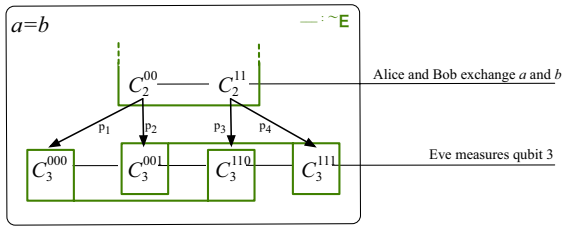

Fig. 5. Eve's equivalence classes when measuring her disentangled qubit.

3 is disentangled from the other two are the measurement statistics for Alice and Bob unaltered. Supposing Eve measures in the last step of the computation, then in this case with $a = b$, a final level of configurations is added to the configuration tree at the left of Fig. 3, as pictured in Fig. 5. Since configurations with different values for the key are equivalent to Eve, again she cannot derive its value. Finally, if Eve carries out a measurement but her qubit is entangled with qubits 1 and 2, the measurement statistics are altered much in the way of Sec. 7 (only the probabilities are different). As we do not reason about probabilities, the analysis of that section applies here as well.

5 Conclusion

We have applied the framework of classical knowledge for quantum systems [8] to analyse the security of quantum key distribution with respect to several types of attacks. Our main point was to show that such developments provide a suitably systematic yet transparent way of investigating cryptographic issues. Of course more work needs to be done. In particular we would like to move to a more abstract level in which we can investigate *all* types of attacks at once, parameterising over arbitrary adversary behaviour. This would entail reasoning about infinite configuration spaces. In particular, it would be interesting to redo Mayers' proof of unconditional security for quantum key distribution [18], to identify prime logical concepts and how they correspond to information-theoretic arguments used in the proof as is. Note that this would require reasoning about multiple runs and probabilities, something for which we currently have no machinery. Surely the model needs to be put further to the test, by investigating other cryptographic primitives and protocols.

In ongoing work [9] we are upgrading our model by merging two previous approaches to reasoning about knowledge in quantum security protocols: the model checking approach exhibited here and the algebraic axiomatics of [22]. These two approaches complement each other: while the latter enjoys a semiautomatic proof search procedure, the former is equipped with a small-step operational semantics for quantum computations. We are currently looking into what combination of both models gets the best of both worlds.

References

[1] Baltag, A. and S. Smets, *LQP: the dynamic logic of quantum information*, Mathematical Structures in Computer Science **16** (2006), pp. 491–525.
URL http://www.vub.ac.be/CLWF/SS/LICS-workshop.pdf

[2] Bennett, C. H. and G. Brassard, *Quantum cryptography: public key distribution and coin tossing*, in: *Proceedings of IEEE international Conference on Computers, Systems and Signal Processing, Bangalore, India* (1984), p. 175.

[3] Bennett, C. H., G. Brassard and N. D. Mermin, *Quantum cryptography without Bell's theorem*, Phys. Rev. Lett. **68** (1992), p. 557.

[4] Danos, V., E. D'Hondt, E. Kashefi and P. Panangaden, *Distributed measurement-based quantum computation*, in: P. Selinger, editor, *Proceedings of the 3rd International Workshop on Quantum Programming Languages (QPL 2005)*, ENTCS **170**, 2005, pp. 73–94, quant-ph/0506070.
URL http://arxiv.org/abs/quant-ph/0506070

[5] Danos, V., E. Kashefi and P. Panangaden, *The measurement calculus*, Journal of the ACM **54** (2007), quant-ph/0704.1263v1.
URL http://arxiv.org/abs/quant-ph/0704.1263v1

[6] D'Hondt, E., "Distributed quantum computation – A measurement-based approach," Ph.D. thesis, Vrije Universiteit Brussel (2005).

[7] D'Hondt, E. and P. Panangaden, *The computational power of the W and GHZ states*, Journal on Quantum Information & Computation **6** (2005), pp. 173–183, quant-ph/0412177.
URL http://arxiv.org/abs/quant-ph/0412177

[8] D'Hondt, E. and P. Panangaden, *Reasoning about quantum knowledge*, in: *Proceedings of the 25th Conference on Foundations of Software Technology and Theoretical Computer Science*, LNCS **3821**, 2005, p. 0544c (to appear), quant-ph/0507176.

[9] D'Hondt, E. and M. Sadrzadeh, *Classical knowledge for quantum security* (2008), accepted at QPL08.

[10] Ekert, A. K., *Quantum cryptography based on Bell's theorem*, Phys. Rev. Lett. **67** (1991), pp. 661–663.

[11] Emerson, E. A., *Temporal and modal logic*, in: J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, MIT Press, 1990 pp. 995–1072.

[12] Fagin, R., J. Y. Halpern, Y. Moses and M. Y. Vardi, "Reasoning about knowledge," MIT Press, 1995.

[13] Gay, S. J. and R. Nagarajan, *Communicating quantum processes*, in: P. Selinger, editor, *Proceedings of the 2nd Workshop on Quantum Programming Languages (QPL04)*, Turku Centre for Computer Science (2004).
URL http://arxiv.org/abs/quant-ph/0409052

[14] Gay, S. J., R. Nagarajan and N. Papanikolaou, *Probabilistic model-checking of quantum protocols* (2005).
URL http://arxiv.org/abs/quant-ph/0504007

[15] Halpern, J. Y., *Reasoning about knowledge: a survey*, , **4** (1995), pp. 1–34.
URL http://www.cs.cornell.edu/home/halpern/papers

[16] Hintikka, J., "Knowledge and belief - An introduction to the logic of the two notions," Cornell University Press, Ithaca, N.Y., 1962.

[17] Lalire, M. and P. Jorrand, *A process algebraic approach to concurrent and distributed quantum computation: operational semantics*, in: P. Selinger, editor, *Proceedings of the 2nd Workshop on Quantum Programming Languages (QPL04)*, Turku Centre for Computer Science (2004).
URL http://arxiv.org/abs/quant-ph/0407005

[18] Mayers, D., *Unconditional security in quantum cryptography*, Journal of the ACM **48** (2001), pp. 351–406.

[19] Nielsen, M. A. and I. Chuang, "Quantum computation and quantum information," Cambridge university press, 2000.

[20] Raussendorf, R., D. E. Browne and H. J. Briegel, *Measurement-based quantum computation on cluster states*, Phys. Rev. A **68** (2003), p. 022312.
URL http://arxiv.org/abs/quant-ph/0301052

[21] Sadrzadeh, M., "Actions and Resources in Epistemic Logic," Ph.D. thesis, University of Quebec at Montreal (2005).

[22] Sadrzadeh, M., *High-level quantum structures in linguistics and multi-agent systems*, in: *Proceedings of AAAI Spring Symposium on Quantum Interaction* (2007).

[23] Shor, P. W., *Algorithms for quantum computation: Discrete logarithms and factoring*, in: *IEEE Symposium on Foundations of Computer Science*, 1994, pp. 124–134.

[24] van der Meyden, R. and M. Patra, *Knowledge in quantum systems*, in: *Proceedings of the 9th conference on Theoretical aspects of rationality and knowledge*, Bloomington, Indiana, 2003, pp. 104–117. URL http://www.cse.unsw.edu.au/~meyden/research/kqs.pdf

Appendix: The small-step semantics for measurement patterns.

For completeness we state the small-step operational semantics of measurement patterns here [5,6]. A pattern is defined as follows.

Definition 5.1 A *pattern* $\mathcal{P}$ is given by a *computation space* V, pattern *inputs* I and *outputs* O, finite sets with associated Hilbert spaces $\mathcal{H}_V$, $\mathcal{H}_I$ and $\mathcal{H}_O$ respectively, and a finite sequence of commands $\mathcal{A} = A_k \dots A_1$, and is denoted

$$\mathcal{P}(V, I, O.\mathcal{A}) \ .$$

Commands are either *entanglements* E_{ij}, *measurements* ${}^t[M_i^\alpha]^s$ or corrections X_i^s or Z_i^s, where $i, j \in V$, $\alpha \in [0, 2\pi]$ and $s, t \in \mathbb{Z}_2$.

The small-step semantics for patterns is in one-to-one correspondence with the different commands of which a pattern command sequence is composed. Before we can give the specific rules for E, M and C commands however, we have to specify the semantics of their component objects, signals and angles. For this we need an extra structure in which we can store and look up measurement outcomes. We call this classical component the *outcome map* Γ. Formally it is a function from the computation space $V \backslash O$ (recall that output qubits are never measured) to the outcome space $\mathbb{Z}_2$, initialised to the empty map, denoted $\varnothing$. The idea is that when qubit i is measured the result s_i is filled into the outcome map such that $\Gamma(i) = s_i$. The resulting outcome map is denoted as $\Gamma[s_i/i]$, which is the outcome map Γ with an additional signal value s_i filled in for qubit i. Signals, which are sums of measurement outcomes, are then evaluated by looking up the signal terms in Γ. The outcome map is the equivalent of the information flow vector from [20]. Together with the quantum state $\rho \in \mathcal{D}(\mathcal{H}_V)$ the outcome map forms the total state of a computation. The *computation state space* $\mathcal{S}$ is therefore defined as

$$\mathcal{S} := \bigcup_{V,O} \mathcal{D}(\mathcal{H}_V) \times \mathbb{Z}_2^W \ , \tag{39}$$

where $W \subseteq (V \backslash O)$ can be any finite set of qubits.

Each of our commands acts on a pair (ρ, Γ) in the state space $\mathcal{S}$. The semantics of a signal s however, can be defined solely with respect to the outcome map. We find the following evaluation rules for signals.

$$\frac{}{\Gamma \vdash 0 \downarrow 0} \quad \text{and} \quad \frac{}{\Gamma \vdash 1 \downarrow 1} \tag{40}$$

$$\frac{}{\Gamma \vdash s_i \downarrow \Gamma(i)} \tag{41}$$

$$\frac{}{\Gamma[v/s_i] \vdash s_i \downarrow v} \quad \text{and} \quad \frac{}{\Gamma[v/s_i] \vdash s_j \downarrow \Gamma(j)} \quad \text{if } i \neq j \tag{42}$$

$$\frac{\Gamma \vdash s \downarrow v \quad \Gamma \vdash t \downarrow u}{\Gamma \vdash s + t \downarrow v \oplus u} \tag{43}$$

Here $\oplus$ denotes addition in $\mathbb{Z}_2$.

Angles, which can have signal dependencies percolated through via dependent measurements, are also purely classical. Values of signals are looked up in Γ via the rule set for signals in order to determine the actual value of a measurement angle. This procedure is summarised in the following rules.

$$\frac{}{\Gamma \vdash \alpha \downarrow \alpha} \tag{44}$$

$$\frac{\Gamma \vdash s \downarrow v \quad \Gamma \vdash t \downarrow u}{\Gamma \vdash {}^{t}[\alpha]^{s} \downarrow (-1)^{v}.\alpha + u.\pi} \tag{45}$$

$$\frac{}{\Gamma \vdash [\alpha]^{s} \downarrow {}^{0}[\alpha]^{s}} \quad \text{and} \quad \frac{}{\Gamma \vdash {}^{t}[\alpha] \downarrow {}^{t}[\alpha]^{0}} \tag{46}$$

Having defined how signals and angles are evaluated, we can now move on to the operational semantics of the basic commands. All of these commands operate on the quantum state.

$$\frac{}{\rho, \Gamma, E_{ij} \longrightarrow \wedge Z_{ij} \rho \wedge Z_{ij}, \Gamma} \tag{47}$$

$$\frac{\Gamma \vdash s \downarrow v}{\rho, \Gamma, X_i^s \longrightarrow X_i^v \rho X_i^v, \Gamma} \tag{48}$$

$$\frac{\Gamma \vdash s \downarrow v}{\rho, \Gamma, Z_i^s \longrightarrow Z_i^v \rho Z_i^v, \Gamma} \tag{49}$$

$$\frac{\Gamma \vdash {}^{t}[\alpha]^{s} \downarrow \beta}{\rho, \Gamma, {}^{t}[M_i^{\alpha}]^{s} \longrightarrow_{\lambda_0} \langle +_\beta|_i \, \rho \, |+_\beta\rangle_i, \Gamma[0/i]} \; , \lambda_0 = \frac{\mathrm{tr}(|+_\beta\rangle\langle+_\beta|_i \, \rho)}{\mathrm{tr}\rho} \tag{50}$$

$$\frac{\Gamma \vdash {}^{t}[\alpha]^{s} \downarrow \beta}{\rho, \Gamma, {}^{t}[M_i^{\alpha}]^{s} \longrightarrow_{\lambda_1} \langle -_\beta|_i \, \rho \, |-_\beta\rangle_i, \Gamma[1/i]} \; , \lambda_1 = \frac{\mathrm{tr}(|-_\beta\rangle\langle-_\beta|_i \, \rho)}{\mathrm{tr}\rho} \tag{51}$$

The first three commands are purely quantum and straightforward. The measurement commands are the only commands that affect the quantum state as well as the output map. First, the measurement angle has to be evaluated, which in turn requires evaluating the X- and Z-signals by the previous sets of rules. Measurement commands are also the only nondeterministic commands, as the measured qubit is projected onto either $|+_\alpha\rangle$ or $|-_\alpha\rangle$ with transition probabilities as stated. Usually, the convention is to *renormalise* the state after measurement, but we do not adhere

to it here, as in this way the probability of reaching a given state can be read off its norm, and moreover the overall treatment is simpler. We have presented these rules here for density matrices; in pure state derivations we often use state transitions for brevity. Specifically, for a pure state we have $\rho = |\psi\rangle\langle\psi|$, which is mapped to $L|\psi\rangle\langle\psi|L^\dagger$, with L any of the entanglement, Pauli or projection operators above. A pure state transition can then be alternatively specified as mapping $|\psi\rangle$ to $L|\psi\rangle$.

Finally, we need an evaluation rule for a composition of commands, as follows.

$$\frac{\rho, \Gamma, C_1 \longrightarrow_\lambda \rho', \Gamma'}{\rho, \Gamma, C_2 C_1 \longrightarrow_\lambda \rho', \Gamma', C_2} \tag{52}$$