Formale Sicherheit in der Quantenkryptologie

Formal Security in Quantum Cryptography

Studienarbeit von Dominique Unruh

Institut für Algorithmen und Kognitive Systeme (IAKS)
Universität Karlsruhe

Betreuer: Prof. Dr. Thomas Beth
Dr. Jörn Müller-Quade

Contents

1 Introduction

1.1 Zusammenfassung

Das Ziel dieser Arbeit ist es, das Konzept der formalen Sicherheit in die Quantenkryptologie einzuführen. Zu diesem Zweck stellen wir ein neues Sicherheitsmodell vor, welches u. a. ein Modell für die Kommunikation in Quantennetzen enthält. Wir werden dabei verstärkt Wert auf formale Details legen, um aufzuzeigen, welche Probleme bei einer formalen Definition von Sicher in der Quantenkryptologie auftreten können.

Auf diesem Modell aufbauend werden wir einige grundlegende Eigenschaften von Quantenprotokollen analysieren:

- Lassen sich Quantenprotokolle modular zusammensetzen?
- Bleiben klassisch sichere klassische Protokolle sicher, wenn man sie in einem Quanten-Umfeld betrachtet?

Beide Fragen werden wir im großen und ganzen positiv beantworten. Die Diskussion dieser Themen (und insbesondere die dabei vorkommenden Beweise) werden weniger formal sein, da die zugrundeliegenden Kommunikationsprozesse so komplex sind, daß dies den Rahmen dieser Arbeit bei weitem sprengen würde.

1.2 Abstract

The objective of this work is to introduce the notion of formal security into quantum computation. In order to achieve this, we introduce a new security model, containing a quantum communication model. We will lay some stress on formal details to show which are the problems arising in a formal definition of quantum security.

Based on this model we will analyse some basic aspects of quantum protocols:

- Are quantum protocols composable?
- Do classically secure classical protocols stay secure in a quantum environment?

Both questions we will answer by and large positively. This discussion (and especially the proofs) will be less formal, because of the complex nature of the underlying communication processes.

1.3 toki lili

wan li ken lukin lon pali ni e ni: sona pi lukin ala pi wan lili li pona ala.

taso tenpo la toki pali li pona, en toki pali ante li pona, kama la toki pali tu pi kama wan li kin pona.

tenpo la wan lili ala la toki pali pi wan lili ala li pona, kama la wan lili la toki pali ni li kin pona.

1.4 Resumo

La celo de tiu ĉi laboro estas la enkonduko de formala koncepto de sekureco en kvantuma kriptografio. Tiucele ni prezentos novan sekurecmodelon, kiu enhavas i. a. modelon de kvantuma komunikado. En nia diskuto ni forte akcentos formalajn detalojn por montri, kiaj estas la problemoj aperantaj je formala defino de kvantuma sekureco.

Uzante tiun modelon, ni esploros kelkajn elementajn ecojn de kvantumaj protokoloj:

- Ĉu kvantumaj protokoloj estas kunmeteblaj?
- Ĉu klasike sekuraj protokoloj restas sekuraj en kvantuma ĉirkaŭajo?

Al ambaŭ demandoj ni respondos pli malpli jese. Tiuj lastaj diskutoj estos malpli formalaj pro la kompleksa naturo de la bazaj komunikadprocesoj.

1.5 Overview

In the first chapter (this one) we give an overview of this work and introduce our modelling of security together with an intuitive approach for its justification.

In the second chapter we discuss some mathematical notation and elementary definitions. This chapter is not directly related to cryptological issues, but gives us some basic tools to operate with in the following chapters.

The third chapter introduces the quantum machine model used throughout this work. We introduce some most abstract machine model, and thereafter present interactive quantum Turing machines. We will discuss run time and termination issues with a view to the interactive nature of these machines. Finally we will introduce classical machines and machines with classical interfaces (i.e. machines which are internally quantum, but restricted to do classical communication).

In the fourth chapter the quantum networks are introduced. These are rather general network models, we do not even give a specific scheduling mechanism there. We do however already implement primitives for message transfer and corruption there.

This simple model is then extended in the fifth chapter, where we introduce the adversarial quantum network. Here issues like scheduling, adversaries, environments, functionalities and primitives are implemented. We will give some simple definitions like that of protocols and communication frameworks to simplify formulations later on. Further several types of adversaries are discussed, like well-formed and non-blocking ones. At the end, we show how the environment in a quantum network generates an overall output bit.

In the sixth chapter the security model is then finally presented. It comes in different flavours, like absolute or approximative security (depending on whether we allow no or only negligible distinguishability probabilities), polynomial security, security with bounded risk (where we can give a precise bound on the probability of distinguishability).

The seventh chapter then introduces extensions to the security model in a less formal way. These are multi-phase protocols, temporary assumptions on available primitives, corruptibility, computational power and knowledge of algorithms, constraints on the security, cheat detection, protocols with partial abort, pliable security, adjustable security and corruption notification.

In the eighth chapter we prove, that a classical protocol, which is proven to be secure when assuming classical communication, is also secure assuming quantum communication.

The ninth chapter handles composability issues. Simple and concurrent composability is discussed and proven, and put together to a notion of universal composability.

In appendix A some rather formal definitions and proofs are given, which would have enlarged the main body of this work unnecessarily.

1.6 An intuitive approach to security

When talking about security, we mostly do not exactly (i.e. formally) know, what this means. In particular cases, e.g. encryption, we have specialised definitions, but in the most general case there where for a long time only an ever growing list of properties a secure system had to provide, e.g. privacy, fairness, and many others. There do however exist more general approaches, which are based on the comparison of some real protocol and an ideal one, the latter being per definition assumed to be secure. Such a system we will introduce, based on the ideas in [Can00b].

The basic idea is a follows: Assume, that we have some given cryptographic application, and we can specify some reference protocol or functionality $\mathcal{F}$, which does implement the wanted behaviour in a secure way. Of course, we do now have to design that functionality, but this is usually easier than designing the protocol, since we do not have to bother with details like insecure communication, or who should carry out computations etc., since $\mathcal{F}$ can be regarded as some trusted party.

Then if we accept, that the functionality is secure (though not necessarily feasible), we can define, that some protocol π is as secure as $\mathcal{F}$, if replacing $\mathcal{F}$ in *any* situation does not result in *any* disadvantage for *any* person (except the adversary, of course). But how do we formalise situations and how do we formalise disadvantages? We simply introduce the concept of the environment and the adversary. The environment is something, which does interact with the protocol and the adversary as a black box and at the end may decide, whether some harmful thing has happened. When we quantify over all possible environments (possible restricting the computational power), and for no environment (i.e. in no situation) some harm has happened using π which would not have happened using $\mathcal{F}$, too, then replacing $\mathcal{F}$ by π is clearly a sensible course of action, at least it will surely do no harm, thus π can be considered as secure.

The preceding explanation still has some great drawback: An adversary (for which the protocols internals are no black box, of course) could simply vary its output depending on whether we use π or $\mathcal{F}$. Then the environment would simply consider the output "π is in use" as harmful, and suddenly π would be insecure. But, when considering our requirement, that *everything harmful, which can happen using π, can also happen using $\mathcal{F}$,* we can reasonably interpret it as *everything harmful, which can happen using π with some adversary $\mathcal{A}$, can also happen using $\mathcal{F}$ with some (other) adversary $\mathcal{S}$.* If we accept this formulation, which implies, that in any situation the adversary may freely choose its strategy to be as "harmful" as possible, we get the security definition which we will develop and examine in this work.

One simplification we will still introduce: So far we have required, that using π, we get *at most* that much harm as when using $\mathcal{F}$. In fact, we can change this definition such, that we require that we get *the same amount* of harm. This simplifies the definition and is in fact equivalent, since if one environment defines something as harm, then some other environment could define the inverse as harm, such bounding the amount of harm from both sides and resulting in a claim of equality. After introducing this simplification, the word "harm" is of course inadequate, so we will use the more neutral notion of the *output bit of the environment.*

So if we put these considerations together, we get a first definition sketch, we runs in the following lines: For any adversary $\mathcal{A}$, there is some adversary $\mathcal{S}$, such that for any environment $\mathcal{E}$, the one-bit output of $\mathcal{E}$ when running with $\mathcal{A}$ and π is indistinguishable from that of $\mathcal{E}$ running with $\mathcal{S}$ and $\mathcal{F}$.

Here of course the notion of indistinguishability leaves us some scope of interpretation, we can e.g. require the distributions of the outputs to be identical, or we can require them to be just very close to each other. In the latter case, if we do not want to introduce some fixed and arbitrary bounds, we need to introduce some security parameter k, which is given to all participants in the above experiment, and which may than be a parameter to some negligible function bounding the statistical distance of the two distributions.

We can now take the above definition and try to generalise it: In the real protocol we might have access to some primitives, and instead of the ideal functionality, we might want to use a whole reference protocol ϱ (the ideal protocol), with which we want to compare π. This leads to the following framework: When running our experiment, we do not only have environment, adversary and protocol *or* functionality, but we have environment, adversary, protocol, *and* functionalities. Here the environment can communicate only with the adversary and the protocol, the protocol's parties may communicate with everyone except directly with each other (for this purpose, functionalities must be used, which possibly may leak information to the adversary or have other explicitly designed drawbacks), the functionalities may communicate with the adversary and the protocol, and the adversary may finally communicate with everyone. The case previously discussed, where we compare π with a functionality $\mathcal{F}$ is realised by using a dummy protocol ϱ_0, which simply passes any communication with the environment to and from $\mathcal{F}$. As we can see, we do not need a special definition of primitives, the notion of functionalities is reused instead. Therefore we will use the words *functionality* and *primitive* synonymously throughout this work.

Some point which we have not yet justified, is why the adversary may not depend on the choice of the environment. For this, we do not have any intuitive justification, but our choice is justified by two facts: First, it is the stricter one, secondly it seems necessary for our proof of composability, which represents a very useful feature, and without which a security model might even be considered worthless (see the introduction of chapter 9).

1.7 Quantum security

Since we are going to consider quantum security in this work, we have to specify, what kind of quantum computation and communication we allow. The following points are to be clarified:

- What kind of messages can be transmitted?
- Is the scheduling of quantum nature?
- Is the fact, whether a message has been send or not, quantum?
- What about erasure?

What kind of messages can be transmitted? In this work we will allow any superposition of tape contents to be transmitted. This means especially, that for an incoming message we do not know without measuring, how long it is, taking any finite part of the message might introduce some information loss. We use this modelling due to the fact, that e.g. some harmonic oscillator has an infinite number of base states, though higher ones usually have rapidly decreasing amplitude. Therefore encoding the oscillator's state in a finite number of qubits necessarily involves a measurement.

Is the scheduling of quantum nature? We have two options here. Either the transmission of the token to some machine may be classical, or several machines may have the token at the same time in superposition. The latter case is of course more general, but it would make the model much more complicated, especially the proofs in chapter 8 and chapter 9 make use of the classical nature of the scheduling.[1] Therefore we choose a classical scheduling.

[1] Though we do not know, whether modelling the scheduling quantumly would invalidate the results in those chapters.

Is the fact, whether a message has been send or not, quantum? Here we model the simpler fact, where sending a message is a classical event. The case, where a message may be sent and unsent in superposition, may however be simulated by introducing a special message $\perp$, meaning "no message".

What about erasure? Often it is required, that a protocol is secure, even if the parties are not able to erase data, i.e. the adversary learns all past states upon corruption. In the quantum case, however, this is not so easy to model, since making a copy of the machine's state after each time step would violate the impossibility to clone quantum states. So we may try to get some weaker condition, where we want to disallow the parties to unnecessarily loose information, i.e. the states should only underly reversible transformations or measurements, whose outcomes are stored and accessible to the adversary after corrupting the party. We have taken care to implement this condition in our model, but it nevertheless seems to be of no use, since a machine can always do the following to erase information: Let's say it has a classical bit. Then it applies a Hadamard transform to it and afterwards decides whether to pass the token to the next party depending on the value of that bit in the standard basis. Since the transmission of the token is a measured event, the state of the qubit is now completely random and its prior state lost. Possibly some more useful notion of impossibility of erasure could be formulated, if we would model the transmission of the token to a quantum event, but this is—according to the opinion of the author—improbable.

2 Notation and elementary definitions

2.1 General conventions

This section is intended to avoid some possible confusion resulting from different conventions in the use of elementary mathematical notation. It is probably unnecessary to read it unless there is some confusion about formal details. When an unclear symbol appears in the text, please consult the symbol index in section A.6.

The set $\mathbb{N}$ consists of the natural numbers without zero (i.e. $\mathbb{N} = \{1, 2, 3, \ldots\}$), while $\mathbb{N}_0$ consists of the natural numbers including zero (i.e. $\mathbb{N}_0 = \mathbb{N} \cup \{0\} = \{0, 1, 2, \ldots\}$). The real numbers are denoted $\mathbb{R}$, the non-negative real numbers are $\mathbb{R}_{\geq 0}$, the positive real numbers $\mathbb{R}_+$, and the complex numbers $\mathbb{C}$.

When writing a product in non-commuting semigroups using forms like e.g. $\prod_{i=0}^{n}$ or $\prod_{i=0}^{\infty}$ or some other form implying some ordering, we will assume that the factors coming first in that ordering appear leftmost in the product, e.g. $\prod_{i=1}^{5} U_i = U_5U_4U_3U_2U_1$. We use this convention, since when writing the successive application of some operators, we do not want to write $\prod_{i=5}^{1} U_i|\Psi\rangle$, but $\prod_{i=1}^{5} U_i|\Psi\rangle$.

2.2 Linear operators and vector spaces

When talking about operators, we always assume linear ones, when talking about projections, we assume orthogonal ones. Bases are always orthonormal throughout this work.

Let $\mathbb{1}$ denote the identity.

When an operator M operates on some vector space $A \otimes B$, we say M operates *read-only on A,* if for any $|\Phi\rangle \in A$, $|\Psi\rangle \in B$ it is $A|\Psi\rangle|\Phi\rangle = |\Psi\rangle|\Xi\rangle$ for some $|\Xi\rangle \in B$.

We say, an operator M on $A \otimes B$ *operates on A as the identity,* if M has the form $M = \mathbb{1} \otimes \tilde{M}$, where $\mathbb{1}$ operates on A and $\tilde{M}$ on B.

2.3 Density matrices and probabilities

In this section we will recall the concept of density matrices and also introduce the generalised density matrices, the latter just being a formal tool to facilitate notation.

Definition 2.1: Density matrix
A *density matrix ϱ in a Hilbert space $\mathcal{H}$* is an operator which can be decomposed as follows:[2]

$$\varrho = \sum_{i \in I} |\Psi_i\rangle\langle\Psi_i|$$

with some countable set I and vectors $|\Psi_i\rangle \in \mathcal{H}$ satisfying

$$\sum_{i \in I} \big\| |\Psi_i\rangle \big\|^2 = 1 \tag{1}$$

A density matrix is often also called a *mixed state,* while a *pure state* designates the special case of $|\Psi\rangle\langle\Psi|$ ($|\Psi\rangle \in \mathcal{H}$).

For convenience, we further define $\varrho(|\Psi\rangle) := |\Psi\rangle\langle\Psi|$. □

A density matrix can be thought as a discrete probability distribution over states in $\mathcal{H}$, since all information which could be extracted about that probability distribution can also be extracted from a density matrix by performing measurements (even if we have an infinite number of samples). To convert a probability distribution given by some probabilities $p(|\Psi_i\rangle)$ to a density matrix, simply define $\varrho := \sum_i p(|\Psi_i\rangle)|\Psi_i\rangle\langle\Psi_i|$.

You can then extract the probability of the outcome of a measurement on $\varrho = \sum_{i \in I}|\Psi_i\rangle\langle\Psi_i|$ given by a projection P by calculating $\sum_{i \in I} \|P|\Psi_i\rangle\langle\Psi_i|\|^2$. Note that though the decomposition is not unique, the results of that calculation are independent of the particular decomposition.

If you have an observable M, which is given by $M = \sum_j m_j P_j$, you can get the expectation value E of M-measuring ϱ without finding a decomposition, simply as $E = \operatorname{tr} M\varrho$.

For convenience, we introduce the generalised density matrices, which are density matrices without condition (1). Thus we have:

Definition 2.2: Generalised density matrix
We call ϱ a *generalised density matrix* if there is a density matrix $\tilde{\varrho}$ and non-negative real number α, such that $\varrho = \alpha\tilde{\varrho}$.

[2]Note that this decomposition is not necessarily unique.

Then we call α the *intensity of* ϱ, and if $\alpha \neq 0$ we say $\tilde{\varrho}$ is the *normalised density matrix of* ϱ. □

Note that $\alpha = \operatorname{tr} \varrho$ and $\tilde{\varrho} = \frac{\varrho}{\alpha}$, so the intensity and the normalised density matrix are well-defined. A generalised density matrix is a density matrix if and only if its intensity is 1.

The idea behind this generalised density matrices is to model states, which appear only with a given probability. E.g. if you have a polarised photon, which is emitted with a probability α, you could model it using a density matrix over a space containing basis vectors $|\updownarrow\rangle$, $|\leftrightarrow\rangle$ and $|\text{no photon}\rangle$. This does also model the possibility, that there may be transformations on that space which e.g. swap the amplitude of $|\updownarrow\rangle$ and $|\text{no photon}\rangle$. If we want to model the simpler case, that the fact whether the photon is present or absent is classical, we write down the probability α of that photon being present, together with its density matrix $\tilde{\varrho}$. Or we simply write $\varrho = \alpha\tilde{\varrho}$. This form has the advantage, that we can now simply calculate on the matrices, e.g. when there is probability α_1 for a photon with density matrix ϱ_1, and a probability α_2 for another indistinguishable photon with density matrix ϱ_2, then the overall generalised density matrix is simply given by $\alpha_1\varrho_2 + \alpha_2\varrho_2$.

By using the following definition we can even reuse the above interconnections between probabilities and density matrices.

Definition 2.3: Generalised probability distribution
A *generalised (discrete) probability distribution* p *on* I is a mapping

$$p : I \to \mathbb{R}_{\geq 0},$$

where I must be a countable set, and $\sum_{i \in I} p(i) \neq 0$. □

Note that for $\sum_{i \in I} p(i) = 1$, a generalised probability distribution is a (discrete) probability distribution.

Like with the generalised density matrices before, we have a natural addition on generalised probability distributions.

To facilitate writing, we further introduce the following function:

Definition 2.4: Density matrix operator
If X is a projector P or an observable M or a measurement[3] $\{P_i, i \in I\}$, or a unitary transformation U, operating on the Hilbert space $\mathcal{H}$, then we define the *density operator* $\ddot{X}$ operating on the *generalised* density matrixes in $\mathcal{H}$ to be respectively given by

- in case of a projector P: $\ddot{X}\varrho := P\varrho P^\dagger$,
- in case of an observable $M = \sum_i \alpha_i P_i$ with projections P_i onto the different eigenspaces of M: $\ddot{X}\varrho := \sum_i P_i \varrho P_i^\dagger$,
- in case of a measurement $\{P_i, i \in I\}$: $\ddot{X}\varrho := \sum_i P_i \varrho P_i^\dagger$,
- and in case of a unitary transformation U: $\ddot{X}\varrho := U\varrho U^\dagger$.

□

Note that one has to know here, of what type a given matrix X is before being able to calculate $\ddot{X}$, e.g. $X := \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)$ could be a projection or an observable, in both cases yielding different results for $\ddot{X}$.

If ϱ is a density matrix and X is *not* a projection, then $\ddot{X}\varrho$ is a density matrix again.

The density matrix operator $\ddot{X}$ can be interpreted as giving the generalised density matrix, that is obtained when applying X to the state and ignoring an eventual measurement outcome. For projections, the intensity is multiplied with the probability that the state is measured to be in the image of that projection, in the other cases the intensity is not changed. E.g. if we have a photon polarised in a random direction (i.e. having density matrix $\mathbb{1}$), then measuring in the $\{|\updownarrow\rangle, |\leftrightarrow\rangle\}$-basis yields the mixed state $\frac{1}{2}|\updownarrow\rangle\langle\updownarrow| + \frac{1}{2}|\leftrightarrow\rangle\langle\leftrightarrow|$, which is exactly what gives $\ddot{M}\mathbb{1}$, where M shall be our measurement.

[3]That is a countable set of projectors P_i with $P_i P_j = 0$ $(i \neq j)$, each associated with some other measurement outcome i.

2.4 Indistinguishability

In this section we will introduce the notion of the indistinguishability of two families of stochastic variables.

We begin by defining what functions (of the security parameter) are so small, that they are—for practical considerations—virtually zero:

Definition 2.5: Negligibility
We say some function $\varepsilon : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is *negligible,* if for any $c \in \mathbb{R}_+$ there is an $k_0 \in \mathbb{N}$, s.t. $\varepsilon(k) \leq k^{-c}$ for all $k \geq k_0$. □

And an alternative and stricter definition is

Definition 2.6: Exponential negligibility
We say some function $\varepsilon : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is *exponentially negligible,* if there is a $c \in \mathbb{R}_+$ and a $k_0 \in \mathbb{N}$, s.t. $\varepsilon(k) \leq c^{-k}$ for all $k \geq k_0$. □

Of course, exponential negligibility implies negligibility.

We first define indistinguishability with respect to a given class of bounds:

Definition 2.7: Stochastic indistinguishability with respect to $\mathcal{B}$
If $\mathcal{B}$ is a set of functions $\mathbb{N} \to \mathbb{R}_{\geq 0}$, then we say two sequences of stochastic variables $(X_k)_{k\in\mathbb{N}}$, $(Y_k)_{k\in\mathbb{N}}$ having values in some countable set A are *stochastically indistinguishable with respect to $\mathcal{B}$,* if it is there is an $\varepsilon \in \mathcal{B}$, s.t.

$$\frac{1}{2}\sum_{a\in A}|P(X_k = a) - P(Y_k = a)| < \varepsilon(k)$$

for all $k \in \mathbb{N}$. □

Note that we require the statistical distance (the left hand side of the equation) to be bounded by ε for all k, not only for sufficiently large ones. So if it is needed to relax this constraint only to hold for sufficiently large k, just define $\tilde{\mathcal{B}}$ consisting of all functions which are almost everywhere equal to some function in $\mathcal{B}$ and then talk about $\tilde{\mathcal{B}}$-indistinguishability instead of $\mathcal{B}$-indistinguishability.

Definition 2.8: Stochastic indistinguishability
Stochastic indistinguishability means stochastic indistinguishability with respect to $\mathcal{B}$, where $\mathcal{B}$ is the set of all negligible functions. □

Definition 2.9: Exponential stochastic indistinguishability
Exponential stochastic indistinguishability means stochastic indistinguishability with respect to $\mathcal{B}$, where $\mathcal{B}$ is the set of all exponentially negligible functions. □

Again exponential stochastic indistinguishability is stronger than stochastic indistinguishability, but we will use stochastic indistinguishability throughout this work, due to the following reasoning: The indistinguishability is used e.g. to model the fact, that we cannot distinguish some probability distributions with some given probability using a reasonable number of samples. Since a reasonable number usually means a polynomial one, we get polynomial bounds for the statistical distance, i.e. indistinguishability, not exponential indistinguishability.

2.5 Symbols and tapes

In modelling the configurations or our machines and networks, we will work very much with symbols and tapes. In this section we will define them and give a short list of some elementary operations on them.

The most elementary data type is the symbol, it is an element out of some set Σ, of which we only specify, that it contains at least the symbols #, `0`, `1`. We call # the blank symbol or simply the blank. We assume further, that there is some numbering of the symbols, i.e. that each symbol is associated with some number in $\{0, \ldots, \#\Sigma - 1\}$. We further assume that # is associated with the number 0.

A tape does, of course, not only hold a single symbol, but sequences of those. There are two important sets of such symbol sequences:

Definition 2.10: Tape content
A *tape content* t is a function $t : \mathbb{Z} \to \Sigma$, where $t = \#$ almost everywhere.

The *empty tape content* # is that tape content t, which is # everywhere.

The set of all tape contents we write $\Sigma^{\mathbb{Z}}_{\text{fin}}$. □

Definition 2.11: String
A *string* s consists of a *length* $l \in \mathbb{N}_0$ together with a function $s : \{1, \ldots, l\} \to \Sigma \setminus \{\#\}$.

If $l = 0$, we talk about the *empty string* $\#$.
The set of all strings we write Σ^*. □

The strings have a natural embedding into the tape contents. A string s of length l can be considered as a tape content t with $t = s$ on $\{1, \ldots, l\}$ and $t = \#$ on $\mathbb{Z} \setminus \{1, \ldots, l\}$.

At some points in this paper we will explicitly specify some strings containing the symbols `a`–`z` (they are recognisable as string literals, because they are typeset with a typewriter font). These literals stand for some arbitrary but fixed strings, which do *not* contain the symbol `0`.

When talking about tapes in this paper, we mean variables which can contain tape contents or strings, it is therefore always necessary to specify, what kind of data a given tape may hold.

On tape contents we have an operation Ѫ to interleave an unbounded number of tape contents into a single one. (We interleave an infinite number, but almost all of them must be empty, otherwise the resulting tape content will have an infinite number of non-empty cells.)

This is done by dove-tailing the cells of the tape contents $a, b, c, d, \ldots$ into one tape content $\text{Ѫ}(a, b, c, d, \ldots)$ which looks as follows (spaces are for legibility only):

$$\ldots,\quad d_{-1}, c_{-2}, b_{-3}, a_{-4},\quad c_{-1}, b_{-2}, a_{-3},\quad b_{-1}, a_{-2},\quad a_{-1},\quad a_0,\quad a_1, b_0,\quad a_2, b_1, c_0,\quad a_3, b_2, c_1, d_0,\quad \ldots$$

Here a_0 is at index 0.

Formally this is:

Definition 2.12: Interleaving tape contents (mapping Ѫ)
Let $(t^{(n)})_{n \in \mathbb{N}}$ be a family of tape contents with $t^{(n)} = \#$ for almost all $n \in \mathbb{N}$. Then the *interleaved tape contents* $(c_i)_{i \in \mathbb{Z}} := \text{Ѫ}(t^{(1)}, t^{(2)}, t^{(3)}, \ldots)$ is a tape content defined by $c_{f(n,i)} := t_i^{(n)}$, where

$$f(n,i) := \begin{cases} \frac{(n+i)(n+i-1)}{2} + n - 1, & \text{if } i \geq 0, \\ -\frac{(n-i-1)(n-i-2)}{2} - n, & \text{if } i < 0. \end{cases}$$ □

Note that Ѫ is bijective, and that the index f is efficiently computable and its value polynomial in its inputs.

For strings we have a similar operation, the structured concatenation. It concatenates a finite number of strings, together with information allowing to extract those strings from the result again.

Definition 2.13: Structured concatenation (mapping Ж)
Let $s_1, \ldots, s_n$ ($n \geq 0$) be some strings. Then the *structured concatenation* $\text{Ж}(s_1, \ldots, s_n)$ of these strings is defined as follows:

For any string s with length l it is $\text{Ж}(s) := s \cdot \mathtt{0}\mathtt{1}^l$ and for strings $s_1, \ldots, s_n$ it is

$$\text{Ж}(s_1, \ldots, s_n) := \text{Ж}(s_1) \cdot \text{Ж}(s_2) \cdots \text{Ж}(s_n),$$

where $\cdot$ denotes the normal concatenation of strings, i.e. writing the strings directly one after another. □

Note that the concatenation is injective, from any image the number of inputs and the i-th input can be efficiently calculated. The output is guaranteed to have at least one `0` in it, i.e. it can be distinguished from strings over $\Sigma \setminus \{\#, \mathtt{0}\}$, in particular from string literals.

The structured concatenation is not associative, we cannot get results like

$$\text{Ж}(a, b, c) = \text{Ж}(\text{Ж}(a, b), c),$$

so we need some operation Ⓠ that has the following property:

$$\text{Ⓠ}(\text{Ж}(s_1, \ldots, s_n), s_{n+1}) = \text{Ж}(s_1, \ldots, s_n, s_{n+1}) \tag{2}$$

This condition completely specifies Ⓠ on $\operatorname{im} \text{Ж} \times \Sigma^*$, but we will need a definition on the whole domain $\Sigma^* \times \Sigma^*$, which now follows:

Definition 2.14: Structured append (mapping Ⓠ)
Let s, t be strings. Then let the *structured append* be defined by

$$\text{Ⓠ}(s, t) := s \cdot \text{Ж}(t),$$

where $\cdot$ is, as before, the normal concatenation. □

This definition satisfies (2) and Ⓠ is injective. Note that it is important that we have defined $\text{Ж}(s) := s\cdot\mathtt{0}\mathtt{1}^l$, not $\text{Ж}(s) := \mathtt{1}^l\mathtt{0}\cdot s$, since in the second case we would only have injectivity on $\operatorname{im}\text{Ж}\times\Sigma^*$, e.g. $\text{Ⓠ}(\mathtt{111101110},\mathtt{0}) = \text{Ⓠ}(\mathtt{11110},\mathtt{100}) = \mathtt{111101110100}$.

It can be efficiently checked, whether $s \in \text{Ⓠ}(\operatorname{im}\text{Ж}\times\Sigma^*)$.

Though there is a natural embedding of strings into the set of tape contents, there is no such in the other direction. From time to time however we need to regard some tape content as a string, i.e. we need a function $\text{Љ} : \Sigma^{\mathbb{Z}}_{\text{fin}} \to \Sigma^*$, which satisfies $\text{Љ}(s) = s$ for $s \in \Sigma^*$. We could define Љ always to yield some fixed value for non-strings, but then Љ could not be implemented by Turing machines any more, since this would imply to check, whether a tape content is a string, which is impossible for Turing machines. So we choose the following approach, which is easily implementable using Turing machines:

Definition 2.15: String extraction (mapping Љ)
For any tape content (t_i), let $l \geq 0$ be minimal with $t_{l+1} = \#$. Then $\text{Љ}((t_i))$ is the string $t_1,\dots,t_l$. □

Due to the reversible nature of quantum computation, the XOR operation is used very often, because of the following properties

- Associativity: $(a\oplus b)\oplus c = a\oplus(b\oplus c)$
- Commutativity: $a\oplus b = b\oplus a$
- Neutrality of 0: $a\oplus 0 = 0\oplus a = a$
- Annulment: $a\oplus a = 0$
- Reversibility: The equations $x\oplus a = b$ and $a\oplus x = b$ have a unique solution x.[4]

We would like to have such an operation for symbols, strings and tape contents, too (with # as the neutral element). Unfortunately, it is not possible on every domain to have an operation fulfilling all five properties.[5] So we will define two operations $\oplus$ and $\ominus$ instead, where $\oplus$ satisfies the associativity, commutativity, neutrality of 0 and the reversibility, while $\ominus$ has the annulment and the reversibility. You can calculate with $\oplus$ and $\ominus$ as with $+$ and $-$.

Definition 2.16: Operations $\oplus$, $\ominus$
Let $n : \Sigma \to \{0,\dots,\#\Sigma-1\}$ be the numbering of Σ. Then we define $\oplus$ and $\ominus$ on $\Sigma\times\Sigma$ via

$$\begin{aligned} n(a\oplus b) &\equiv n(a)+n(b) \mod \#\Sigma,\\ n(a\ominus b) &\equiv n(a)-n(b) \mod \#\Sigma. \end{aligned}$$

On $\Sigma^{\mathbb{Z}}_{\text{fin}}\times\Sigma^{\mathbb{Z}}_{\text{fin}}$ we define $(a_i)_i\oplus(b_i)_i := (a_i\oplus b_i)_i$ and $(a_i)_i\ominus(b_i)_i := (a_i\ominus b_i)_i$
Let $\tilde{n}' : \Sigma^* \to \mathbb{N}_0$ be defined as

$$\tilde{n}'(s) = \sum_{i=1}^{l(s)} n(s_i)(\#\Sigma-1)^{i-1}$$

where $l(s)$ is the length of s, and let $n' : \Sigma^* \to \mathbb{Z}$ be

$$n'(s) = \begin{cases} \frac{\tilde{n}'(s)+1}{2}, & \text{if } \tilde{n}'(s) \text{ odd},\\ -\frac{\tilde{n}'(s)}{2}, & \text{if } \tilde{n}'(s) \text{ even}. \end{cases}$$

Then let $\oplus$, $\ominus$ be on $\Sigma^*\times\Sigma^*$ as follows:

$$\begin{aligned} n'(a\oplus b) &= n'(a)+n'(b),\\ n'(a\ominus b) &= n'(a)-n'(b). \end{aligned}$$ □

The details of these operations are only given for completeness. In this paper we only need their aforementioned properties, and the fact, that they can be efficiently computed.

Finally we need some way to represent integers. For our needs the following very simple definition is sufficient:

Definition 2.17: Representing integers as strings (mapping u)
Let $u : \mathbb{Z} \to \Sigma^*$ be defined by $u(n) := \mathtt{1}^n$ for $n \geq 0$, $u(n) := \mathtt{0}\mathtt{1}^{-n}$ for $n < 0$. □

[4] The reversibility follows from the annulment and the associativity, of course.

[5] Assume such an operation on $\{0,1,2\}$. It is $1\oplus 2\oplus 2 = 1$, therefore $1\oplus 2 \neq 0$ and $1\oplus 2\neq 1$. Further $2\oplus 1\oplus 1 = 2$, thus $1\oplus 2 = 2\oplus 1 \neq 2$. So such an operation is impossible. ■

3 Quantum Machine Models

3.1 Interactive Quantum Systems

The most general type of quantum machines we will discuss in this work is the *interactive quantum system.* We will give the formal definition first and then discuss its properties in detail.

Definition 3.1: Interactive quantum system (IAQS)
An *interactive quantum system* $\mathcal{M}$ consists of the following objects:

- a unitary *time evolution operator* $U_{\mathcal{M}}$ operating on $\mathcal{H}_{Q,\mathcal{M}}$,
- an *end configuration projection* $P_{\mathrm{F},\mathcal{M}}$ operating on $\mathcal{H}_{Q,\mathcal{M}}$,
- a unitary *frame send operator* $U_{\mathrm{FS},\mathcal{M}}$ operating on $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F$,
- a unitary *send operator* $U_{\mathrm{S},\mathcal{M}}$ operating on $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_C$,
- a unitary *receive operator* $U_{\mathrm{R},\mathcal{M}}$ operating on $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C$ with $\operatorname{im} U_{\mathrm{R},\mathcal{M}} \subseteq \mathcal{H}_{Q,\mathcal{M}} \otimes |\#\rangle \otimes |\#\rangle$,
- a unitary *corruption operator* $U_{\mathrm{corr},\mathcal{M}}$ operating on $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_C$,
- and a *start configuration* $|\Psi_{0,\mathcal{M}}\rangle \in \mathcal{H}_{Q,\mathcal{M}}$,

where $\mathcal{H}_{Q,\mathcal{M}}$ is the *configuration space!of an IAQS* $\mathcal{H}_C$ the *communication tape space* and $\mathcal{H}_F$ the *frame tape space.* It is

$$\mathcal{H}_{Q,\mathcal{M}} \cong \mathbb{C}^{\Sigma^{\mathbb{Z}}_{\mathrm{fin}}}, \qquad \mathcal{H}_C \cong \mathbb{C}^{\Sigma^{\mathbb{Z}}_{\mathrm{fin}}}, \qquad \mathcal{H}_F \cong \mathbb{C}^{\Sigma^*}.$$

The set of all IAQS is called IAQS. □

The time evolution operator defines one single calculation step of the IAQS $\mathcal{M}$. It operates on the whole configuration of the machine (unlike e.g. the state transition functions of Turing machines which operates on the state and a finite part of the tape) which is modelled by the countable dimensional Hilbert space $\mathcal{H}_{Q,\mathcal{M}}$.

We have arbitrarily chosen $\mathcal{H}_{Q,\mathcal{M}} \cong \mathbb{C}^{\Sigma^{\mathbb{Z}}_{\mathrm{fin}}}$, since the similar structure of the configuration space and the communication tape space will simplify the definition of particular corruption operations later on. Since all input of the machine will be of countable dimension, only a countable-dimensional part of the configuration space will be populated in a finite number of steps, so an uncountable-dimensional configuration space would not give more generality. Since all countable-infinite-dimensional Hilbert spaces are isomorphic, our choice is of maximal generality.

Because we have imposed no restrictions on $U_{\mathcal{M}}$ (except to be unitary), and our configuration space is large enough, our machines can evaluate any classical function (i.e. a permutation of base configurations) depending on all past inputs in a single step. Moreover any quantum mechanical operation on countable-dimensional Hilbert spaces can be modelled.

In order to model calculations we also need the notion of an end configuration. Since we want most generality, we just give a measurement projection $P_{\mathrm{F},\mathcal{M}}$. The 1-eigenspace E_1 should be the subspace of all end configurations. Normally E_1 is the span of a set of base vectors, though this must not necessarily apply. Note that in the definition of the quantum network (see section 4.1) this measurement will be applied after each time step, so the configuration of the IAQS will always be in $E_0 \cup E_1$ and no interferences between paths which pass through E_1 at *different* times will occur. Note that the end configuration does not denote the final end of the calculation, the IAQS may be reactivated later. Mostly $\mathcal{M}$ goes into the end configuration to send a message (see below).

We want to model communication processes, therefore we need some way to interact. In our model we have split this operation into three different operations. The first one is writing on the frame tape. This is done via the frame send operator $U_{\mathrm{FS},\mathcal{M}}$. This operator is activated whenever $\mathcal{M}$ reaches an end configuration. Its function is to transfer some output generated by $\mathcal{M}$, which is still stored internally, onto the frame tape. We may assume that when activating the operator the frame tape space $\mathcal{H}_F$ is cooled. The frame tape will be measured (see section 4.1) after application of $U_{\mathrm{FS},\mathcal{M}}$. The information output this way is intended for control information as e.g. the target of a message transmission, the request to corrupt a party, etc., i.e. actions which are measured anyway in our model (see section 1.7).

Let us clarify the usage of $U_{\mathrm{FS},\mathcal{M}}$ by example: If we implement $\mathcal{M}$ e.g. as a multi-tape Turing machine[6], we could have a distinct frame tape inside, and $U_{\mathrm{FS},\mathcal{M}}$ would copy (or swap) the data from the internal frame tape to the external one.

[6]I.e. we think of $\mathcal{H}_{Q,\mathcal{M}}$ as a set of tapes, of head positions and a state of the finite automate of the Turing machine.

The second operation is sending data. This is done via the send operator $U_{\mathrm{S},\mathcal{M}}$. It will be applied whenever $\mathcal{M}$ has requested sending a message. Control data like the message recipient must be given in the frame. As does $U_{\mathrm{FS},\mathcal{M}}$, the send operator transfers data to be sent from inside the machine to the outside. It can rely on $\mathcal{H}_C$ to be cooled before activation, but it must not rely one the fact that $\mathcal{H}_C$ will be measured, since this channel is intended for sending quantum data.

To continue our example: $U_{\mathrm{S},\mathcal{M}}$ would swap an internal communication tape with the external one. Note that copying is not possible in this case, since the data should not be measured.

The third operation is receiving data. Here frame and data transmission are not separated, since the IAQS does not have to request the reception of a message; whenever a message arrives, the operator $U_{\mathrm{R},\mathcal{M}}$ is executed (but it is guaranteed that this will only happen while $\mathcal{M}$ is in the end configuration). To be able to ensure that $\mathcal{H}_F$ and $\mathcal{H}_C$ are cooled before the frame sending resp. the data sending operation, $U_{\mathrm{R},\mathcal{M}}$ must leave $\mathcal{H}_F$ and $\mathcal{H}_C$ in the cooled state after operation.

In our example: $U_{\mathrm{R},\mathcal{M}}$ appends the data from the frame tape onto some internal tape (appending is no problem, since the frame tape contains classical data, so we may measure its length), erases the frame tape (possible, since we have made a copy), and then moves the frame tape into some reservoir of internal tapes of $\mathcal{M}$, replacing it by a cooled tape, approximately like this:

$$|\mathrm{comm}\rangle \otimes |\mathit{tape}_1\rangle \otimes |\mathit{tape}_2\rangle \otimes |\mathit{tape}_3\rangle \otimes \cdots \longmapsto |\#\rangle \otimes |\mathrm{comm}\rangle \otimes |\mathit{tape}_1\rangle \otimes |\mathit{tape}_2\rangle \otimes \cdots$$

Note that this operation is unitary, though not surjective.

The last operator appearing in the definition of the IAQS is the corruption operator $U_{\mathrm{corr},\mathcal{M}}$. When a corruption occurs, this operator extracts the configuration from $\mathcal{M}$ and writes it onto the communication tape to be transfered to the corrupting machine. This may destroy $\mathcal{M}$ in the sense that further activations of any operators have an undefined meaning. $U_{\mathrm{corr},\mathcal{M}}$ should not copy the information, since this might create additional and unwanted entanglement between the adversary and the "corpse" of $\mathcal{M}$. In the simplest (and most important) case $U_{\mathrm{corr},\mathcal{M}}$ just swaps $\mathcal{H}_{Q,\mathcal{M}}$ and $\mathcal{H}_C$. We will however need more sophisticated corruption operators when modelling classical machines (see definition 3.14).

The meaning of the start configuration is—as its name suggests—the configuration in which the IAQS is at the beginning of its execution.

3.2 Execution transcripts

In order to define some properties like polynomial running time etc., we need to be able to formulate statements like "for any external interaction, the IAQS $\mathcal{M}$ behaves in such and such a way with a probability of..." To achieve this, we will in this section define execution transcripts.

Definition 3.2: Quantum interaction
A *quantum interaction for the IAQS $\mathcal{M}$* is a sequence of linear operators (O_i) operating on $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^2$, together with some projections $(P_k^{(i)})_{i\in\mathbb{N},k\in\{run,send,rec,world\}}$, where each operator O_i is of the form:

$$\begin{aligned}
O_i|\Psi\rangle \otimes |\mathrm{frame}\rangle \otimes |\mathrm{comm}\rangle \otimes |n\rangle \otimes |1\rangle \longmapsto\ & P^{\complement}_{\mathrm{F},\mathcal{M}} U_{\mathcal{M}}|\Psi\rangle \otimes |\mathrm{frame}\rangle \otimes |\mathrm{comm}\rangle \otimes |n\rangle \otimes |1\rangle && (3)\\
&+ U_{\mathrm{FS},\mathcal{M}}(P_{\mathrm{F},\mathcal{M}} U_{\mathcal{M}}|\Psi\rangle \otimes |\mathrm{frame}\rangle) \otimes |\mathrm{comm}\rangle \otimes |n\rangle \otimes |0\rangle && (4)\\
O_i|\Psi\rangle \otimes |\mathrm{frame}\rangle \otimes |\mathrm{comm}\rangle \otimes |n\rangle \otimes |0\rangle \longmapsto\ & P^{\complement}_{\mathrm{F},\mathcal{M}} U_{\mathcal{M}}|\Psi\rangle \otimes |\mathrm{frame}\rangle \otimes |\mathrm{comm}\rangle \otimes P^{(i)}_{run}|n\rangle \otimes |1\rangle && (5)\\
&+ U_{\mathrm{FS},\mathcal{M}}(P_{\mathrm{F},\mathcal{M}} U_{\mathcal{M}}|\Psi\rangle \otimes |\mathrm{frame}\rangle) \otimes |\mathrm{comm}\rangle \otimes P^{(i)}_{run}|n\rangle \otimes |0\rangle && (6)\\
&+ U_{\mathrm{S},\mathcal{M}}(|\Psi\rangle \otimes |\mathrm{frame}\rangle \otimes |\mathrm{comm}\rangle) \otimes P^{(i)}_{send}|n\rangle \otimes |0\rangle && (7)\\
&+ U_{\mathrm{R},\mathcal{M}}(|\Psi\rangle \otimes |\mathrm{frame}\rangle \otimes |\mathrm{comm}\rangle) \otimes P^{(i)}_{rec}|n\rangle \otimes |0\rangle && (8)\\
&+ |\Psi\rangle \otimes U^{(i)}(|\mathrm{frame}\rangle \otimes |\mathrm{comm}\rangle \otimes P^{(i)}_{world}|n\rangle) \otimes |0\rangle, && (9)
\end{aligned}$$

and where $P_k^{(i)}$ are projections operating on $\mathbb{C}^{\mathbb{Z}}$, satisfying $\sum_k P_k^{(i)} = \mathbb{1}$, and $U^{(i)} \in U(\mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}})$. □

The idea behind this sequence of operators is to model any interaction of $\mathcal{M}$ with the exterior world. The state of the world including $\mathcal{M}$ is modelled by $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^2$, where $\mathcal{H}_{Q,\mathcal{M}}$, $\mathcal{H}_F$, and $\mathcal{H}_C$ represent the state, frame tape and communication tape of $\mathcal{M}$, $\mathbb{C}^{\mathbb{Z}}$ is the state of the external world, and $\mathbb{C}^2$ is a qubit denoting whether $\mathcal{M}$ is running.

So the first equation (3, 4) says, that if $\mathcal{M}$ is running, O_i shall apply the time evolution $U_{\mathcal{M}}$, then measure, whether the machine has reached an end configuration, store this result in $\mathbb{C}^2$, and, if yes, let the frame send operator $U_{\mathrm{FS},\mathcal{M}}$ operate on $\mathcal{M}$.

If however the machine is not running (as denoted by having $|0\rangle$ in $\mathbb{C}^2$), we distinguish four cases, depending on the result which is yielded by measuring whether the content of $\mathbb{C}^{\mathbb{Z}}$ (the state of the world) falls into the subspace of P_{run}, P_{send}, P_{rec} or P_{world}. In case of P_{send} (5,6), we do the same things as in the first equation, i.e. we let the machine run one step.

In case of P_{send} (7) and P_{rec} (8), we apply the send operator $U_{\mathrm{S},\mathcal{M}}$ resp. the receive operator $U_{\mathrm{R},\mathcal{M}}$.

In case of P_{world} (9), some arbitrary unitary transformation is done on the state of the world and the external tapes of $\mathcal{M}$.

Note that the O_i, as defined above, are unitary.

The above definition of quantum interactions still has one problem: It may happen, that the operators $U_{\mathrm{FS},\mathcal{M}}$ and $U_{\mathrm{S},\mathcal{M}}$ are invoked while $|\mathrm{frame}\rangle$ or $|\mathrm{comm}\rangle$ are not cooled. The behaviour of the IAQS may be unpredictable in this case, since our specification promises that those two tapes are cooled on application of $U_{\mathrm{FS},\mathcal{M}}$ and $U_{\mathrm{S},\mathcal{M}}$.

Definition 3.3: Well-formedness of an quantum interaction
We say that a quantum interaction ι is *well-formed,* if

$$(\mathbb{1} \otimes (P_{\#} \otimes P_{\#})^{\complement} \otimes (P^{(n+1)}_{run} + P^{(n+1)}_{send}) \otimes \mathbb{1}) \prod_{i=1}^{n} (O_i P_{r_i} P_{f_i} P^{(i)}_{k_i})(|\Psi_{0,\mathcal{M}}\rangle \otimes |\#\rangle \otimes |\#\rangle \otimes |0\rangle \otimes |0\rangle) = 0$$

for all $n \in \mathbb{N}_0$ and all $r_i \in \{0,1\}$, $f_i \in \Sigma^*$, $k_i \in \{run, send, rec, world\}$, where $P_{\#}$ is the projection onto $|\#\rangle$, $P^{(i)}_k$ with $k \in \{run, send, rec, world\}$ are the projections from the quantum interaction, P_{r_i} is the projection of $\mathbb{C}^2$ onto $|r_i\rangle$ and P_{f_i} the projection of $\mathcal{H}_F$ onto $|f_i\rangle$. This equation lives in $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^2$.

Throughout this work, we will always implicitly assume well-formedness when talking about quantum interactions. □

In this definition, we start with the initial state $|\Psi_{0,\mathcal{M}}\rangle \otimes |\#\rangle \otimes |\#\rangle \otimes |0\rangle \otimes |0\rangle$ of the system, which consists of the start configuration of $\mathcal{M}$ and of cooled tapes. Then we apply the n-steps of the quantum interaction, and measure afterwards, whether we are in the *run* or *send* state (which may involve the frame send operator or the send operator), and $|\mathrm{frame}\rangle$ or $|\mathrm{comm}\rangle$ are not cooled. The probability for this to happen after n steps should be 0 for any n.

We will now define the *execution transcript,* which is formalises the classical behaviour of a machine in response to some quantum interaction. Since the system is of quantum and therefore after measurement probabilistic nature, the execution transcript is a probability distribution of sequences of output events. Each output event can be one of

- $(run, \#)$: The machine is running in this time step.
- $(stop, f)$: The machine stops in this time step and—after application of the frame send operator—outputs some frame f.
- (rec, f): The machine receives some data and a frame f. The message's data is not given, since this data should not be measured because of its quantum nature.
- $(send, \#)$: The machine sends some data. Again the data itself is not measured.
- $(world, \#)$: In this time step, some transformation outside the IAQS happens.

This stochastic variable is formally defined as follows:

Definition 3.4: Execution transcript
Let $\mathcal{M}$ be an IAQS. Let further an quantum interaction ι be given. Then the *execution transcript of $\mathcal{M}$ on ι* is a stochastic variable $(X_i)_{i\in\mathbb{N}_0}$, where each X_i has values from $\{run, stop, rec, send, world\} \times \Sigma^*$. The distribution of this stochastic variable is derived of that of the stochastic variable (Y_i) via the equations

$$\begin{aligned}
X_i = (k, \#) &\iff Y_i = (k, 0, \cdot) && (k \in \{send, world\}) && (10)\\
X_i = (rec, f) &\iff Y_i = (rec, 0, f) && (f \in \Sigma^*) && (11)\\
X_i = (run, \#) &\iff Y_i = (run, 0, \cdot) \text{ and } Y_{i+1} = (\cdot, 1, \cdot) \text{ or} && && \\
& \qquad Y_i = (\cdot, 1, \cdot) \text{ and } Y_{i+1} = (\cdot, 1, \cdot) && && (12)\\
X_i = (stop, f) &\iff Y_i = (run, 0, \cdot) \text{ and } Y_{i+1} = (\cdot, 0, f) \textit{ or} && && \\
& \qquad Y_i = (\cdot, 1, \cdot) \text{ and } Y_{i+1} = (\cdot, 0, f) && (f \in \Sigma^*) && (13)
\end{aligned}$$

where (Y_i) be defined via

$$P(Y_i = (k_i, f_i, r_i), i = 1, \dots, n) = \left\| \prod_{i=1}^{n} (O_i P_{r_i} P_{f_i} P_{k_i}^{(i)})(|\Psi_{0,\mathcal{M}}\rangle \otimes |\#\rangle \otimes |\#\rangle \otimes |0\rangle \otimes |0\rangle) \right\|^2$$

for $k_i \in \{\mathit{run}, \mathit{send}, \mathit{rec}, \mathit{world}\}$, $r_i \in \{0, 1\}$ and $f_i \in \Sigma^*$, and where P_{r_i} denotes the projection of $\mathbb{C}^2$ onto $|r_i\rangle$, P_{f_i} denotes the projection of $\mathcal{H}_F$ onto $|f_i\rangle$, and $P_{k_i}^{(i)}$ are the projections from the quantum interaction ι. The equation lives in $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^2$.

We also call a sequence (x_i) with components in $\{\mathit{run}, \mathit{stop}, \mathit{rec}, \mathit{send}, \mathit{world}\} \times \Sigma^*$ an execution transcript, therefore the concept both stands for outputs of a single run and for the distribution of such outputs.

We say an execution transcript (x_i) is *impossible on* ι, if there is some $n \in \mathbb{N}$, such that $P(X_i = x_i, i = 1, \dots, n) = 0$. A set of execution transcripts is said to be impossible if each transcript is impossible. □

In order to define the execution transcript (X_i), we have first defined some stochastic variable (Y_i), which denotes the output of the following algorithm:

1. Measure which will be the next action of the quantum interaction (run, rec, send, world). To achieve this, the quantum interaction itself provides the projections $(P_k^{(i)})_{i \in \mathbb{N}, k = run, send, rec, world}$.
2. Measure what is written on the frame tape.
3. Measure, whether the machine is running (i.e. will be running during the following time step). We have constructed the quantum interaction thus, that this information can be found in $\mathbb{C}^2$.
4. Output the results of these measurements.
5. Go to step 1.

From the data gathered by the preceding algorithm, we can easily deduce everything we wanted to put into the execution transcript:

- If we measured *send* or *world* as the quantum interaction's action, while the machine was not running, we write into the execution transcript, a $(\mathit{send}, \#)$ resp. a $(\mathit{world}, \#)$ event.
- If we measure a *rec* action and a frame f, we know that the machine is going to receive f, so we append (rec, f) to the transcript.
- When we have a *run* action (or the machine was already running), and the machine will still be running after this time step, we append $(\mathit{run}, \#)$.
- When we have a *run* action (or the machine was already running), but after this step the machine is not running any more, the machine obviously stops in this step, so we take the frame f of the *next* time step and append (stop, f) to the transcript.

This is formalised by equations (10–13). Note that X_i is in fact well-defined, since exactly one of the right-hand sides of those equations will hold.

The notion of impossibility is best clarified by an example. Consider the result of tossing a coin an infinite number of times and writing down the result of each coin toss twice. Then the sequence $(0, 0, 0, \dots)$ and the sequence $(1, 0, 0, \dots)$ both have probability 0, but the second is impossible, while the first is not. We have formalised this idea for execution transcripts, but in fact this definition can be used for any stochastic variable which yields a sequence of objects, each coming from a countable set.[7]

Note that sets of impossible transcripts have probability 0, since for an impossible set E it is

$$\begin{aligned} P(\mathrm{X} \in E) &= P(\mathrm{X} \in E \cap \bigcup_{n}^{\infty} \{(x_i) : P(\mathrm{X}_i = x_i, i = 1, \dots, n) = 0\}) \\ &\le \lim_{n \to \infty} P(\mathrm{X} \in \{(x_i) : P(\mathrm{X}_i = x_i, i = 1, \dots, n) = 0\}) \\ &\le \lim_{n \to \infty} \sum_{\substack{x_1, \dots, x_n \text{ with} \\ P(x_i = \mathrm{X}_i, i = 1, \dots, n) = 0}} \underbrace{P(\mathrm{X}_i = x_i, i = 1, \dots, n)}_{=0} = 0, \end{aligned}$$

because there is only a countable number of possible *finite* sequences $x_1, \dots, x_n$. ■

[7] If we would allow non-countable sets, too, we get a problem: Consider (X_i) to be a series of i.i.d. stochastic variables uniformly distributed in $[0, 1]$, Than any value for (X_i) is impossible, because any value for X_i has probability 0. So the probability, that (X_i) has an impossible output, is 1.

3.3 Running time of an IAQS

3.3.1 Running time

We will now first introduce a stochastic variable which represents the running time of an IAQS. This will allow us to formulate most of the ideas of the following sections more easily.

Definition 3.5: Running time
Consider some execution transcript $\chi = (x_i)$. It is $\chi \in K_k$ $(k \geq 0)$, if one of the following two conditions is met:

- For the first element of χ having the form (rec, f) and appearing before the first element having the form $(\mathit{run}, \#)$ or $(\mathit{stop}, \cdot)$, the word f has length k.
- There is no such element and $k = 0$.

It is further $\chi \in E_{k,n,t}$ $(k \geq 0,\, n \geq 1,\, t \in \mathbb{N} \cup \{\infty\})$, if $\chi \in K_k$ and one of the following conditions is satisfied:

- Let $(r_i)_{i=1}^{m}$ (where m may be ∞) be the n-th sequence of consecutive elements of (x_i) which has the form $r_i = (\mathit{run}, \#)$ for $(i \neq m)$ and $r_m = (\mathit{stop}, f)$ (where (r_i) may also consist only of one element (stop, f)), or be an infinite sequence of $(\mathit{run}, \#)$, and is not preceded by an element $(\mathit{run}, \#)$. Then $m \leq t$ (and in particular $m \neq \infty$ if $t \neq \infty$).
- There is no such sequence.

For convenience, we set $E_{k,n,0} := \emptyset$.
Let E be the set of all execution transcripts.
The *running time $T_{k,n}^{\mathrm{X}}$ of an execution transcript* X *in invocation n with security parameter k* is defined by

$$P(T_{k,n}^{\mathrm{X}} = t) = \frac{P(\mathrm{X} \in E_{k,n,t} \setminus E_{k,n,t-1})}{P(\mathrm{X} \in \bigcup_{t \in \mathbb{N} \cup \{\infty\}} E_{k,n,t})} \qquad (k \geq 0, n \geq 1, t > 0).$$

If the denominator is 0, we set $P(T_{k,n}^{\mathrm{X}} = t) := 0$ for $t > 0$ and $P(T_{k,n}^{\mathrm{X}} = \perp) := 1$, otherwise $P(T_{k,n}^{\mathrm{X}} = \perp) := 0$. Let $P(T_{k,n}^{\mathrm{X}} = \infty) := P(T_{k,n}^{\mathrm{X}} \notin \mathbb{N} \cup \{\perp\})$. □

In this definition, we have two auxiliary concepts: K_k is the set of all transcripts, in which the length of the first input frame is k (with $k := 0$ if there is no such frame, or if it only appears after the first invocation).

$E_{k,n,t}$ is the set of all transcripts, in which the length of the first input frame is k (with $k = 0$ if there is no such frame), and in which the n-th invocation has running time at most t (or does not happen at all).

In this definition, the probability of running time t in invocation n with security parameter k is defined as the conditional probability of having an execution transcript with security parameter k and which satisfies, that there is no n-th invocation or that the n-th invocation has running time t $(E_{k,n,t} \setminus E_{k,n,t-1})$, under the condition that in this transcript the security parameter is k $(\bigcup_{t \in \mathbb{N} \cup \{\infty\}} E_{k,n,t} = K_k)$. Note that t may also be ∞.

If the probability of the denominator is 0, this means that the security parameter is not k, and we simply set $T_{k,n}^{\mathrm{X}} := \perp$.

3.3.2 Polynomial IAQS

We will now examine the concept of polynomially bounded running time. In the context of networks, the notion of running time is a little more complex than regarding stand-alone machines: There is the running time per invocation and the number of invocations, which both should be polynomial to ensure, that no computationally infeasible problems are solved by the machines. In our definition of polynomially bounded IAQS we will however only consider the running time of a single invocation, since for reasons discussed in section 5.2 a limitation of the number of invocations would cause some problems. Bounding the number of invocations to a polynomial shall then be realised by the scheduling, see section 5.2 and the definition of polynomial security (definition 6.3).

A further difference to traditional notions of security is the fact, that we cannot let the running time of each invocation depend on the length of the input of that specific invocation, since this would cause two problems:

- Some invocations might get no or little input, but nevertheless the machine might want to work on some data from previous invocations which would request a running time bounded depending of the size of that previous input.

- Two machines might send each other messages, each twice the size of the preceding one. If each invocation has a running time linear in its input length, the running time is nevertheless growing exponentially.

Therefore it is necessary to let the running time depend only on the length of the first input.

We come to the following notion of polynomial running time: A machine has polynomial running time if there is a polynomial p, such that the following holds: Let k' be the length of the first frame ever received, 0 if no such exists. Then each invocation terminates after at most $p(k')$ steps.

This definition also makes it fairly simple to distribute some security parameter k to all machines. We just send a frame with the unary encoded security parameter before sending anything else, then the running time in each invocation will be bounded by $p(c_1 k + c_2)$, where $c_1 \geq 1$, $c_2 \geq 0$ denote some overhead used in encoding the frame containing the security parameter. Since this $p(c_1 X + c_2)$ is a polynomial again, the running time of each invocation will be bounded polynomially in k.

There is a further finesse in the notion of polynomial running time. We have the following choices:

- *Strict polynomial running time*: The running time cannot exceed $p(k)$. Note that here it is the same to say "the probability is 0" and "it is impossible" (see section A.3 for a proof).

- *Reliable polynomial running time*: The probability that the running time exceeds $p(k)$ is for each invocation bounded by a function negligible in k.

- *Expected polynomial running time*: The expectation value of the running time for each invocation is bounded by $p(k)$.

The first of these choices is obviously the strictest one. And in cases where we consider absolute security (i.e. where we want the probability distributions of the environment's output to be identical, see definition 6.1) it is the only acceptable one.

In the case that we want only indistinguishable distributions, the definition of reliable polynomial running time seems an interesting choice, too, but it can be seen that by replacing a machine which exceeds its running time with a negligible probability by a machine which forcibly terminates after $q(p(k))$ steps (the polynomial q shall denote some overhead for counting steps) but operates identically to the other machine, the overall behaviour of the system is—with overwhelming probability—only changed in its overall step count (which we will not consider in our definition of security). So in this case we may also use the first definition of polynomially bounded running time.

The definition of expected polynomial time is of course weaker than the strict one. But it is not comparable with the reliable one, as the following examples demonstrate:

- A machine having running time T with $P(T = n) = 2^{-n}$ has expected polynomial running time (since $ET = \sum n2^{-n}$ is finite and independent of k), but for each polynomial p the bound $p(k)$ is exceeded with a non-vanishing probability independent of k, so it is not of reliable polynomial running time.

- A machine running a constant number of steps with overwhelming probability but non-terminating with a non-vanishing probability is of course of reliable polynomial running time, but not of expected polynomial time, since the expectation value of the running time does not even exist.

There is a disadvantage to the definition of expected polynomial running time. If we have e.g. some machine $\mathcal{M}_1$ with a running time $P(T^{(1)}_{n,k} = t) \propto t^{-3}$, and another machine $\mathcal{M}_2$ simulating $\mathcal{M}_1$ with quadratic overhead, then $\mathcal{M}_1$ has expected polynomial running time, since $ET^{(1)}_{n,k} \propto \sum_t^\infty t^{-2}$ is finite and independent of k and n, whereas $ET^{(2)}_{n,k} \propto \sum_t^\infty t^{-1} = \infty$, so $\mathcal{M}_2$ is not of expected polynomial running time. Therefore this class of machines is not closed under simulation with polynomial overhead.

We do not know, whether using this definition of polynomial running time would make a difference to our notion of security.

There are many further possible variants of the definition of polynomial running time, e.g. in the definition of reliable polynomial time we could consider the probability, that the bound is exceeded for any invocation (instead of calculating that probability for each invocation separately), or combinations of the above definitions.

We will choose the notion of strict polynomial running time as the default, since the exact implications of the expected one are unclear in this context, we will nevertheless also formalise the other two for completeness.

Definition 3.6: (Strict) polynomial running time

We say an IAQS $\mathcal{M}$ has *(strict) polynomial running time* or *(strictly) polynomially bounded running time* if there is a polynomial p such that for any quantum interaction ι it is

$$P(T^{\mathrm{X}}_{k,n} < p(k+n) \text{ or } T^{\mathrm{X}}_{k,n} = \perp) = 1$$

for all $n \in \mathbb{N}$, $k \in \mathbb{N}_0$ with X being the execution transcript on ι.

An IAQS with polynomially bounded running time we call a *polynomial IAQS (PIAQS)*, the set of all polynomial IAQS is written PIAQS. □

Definition 3.7: Reliable polynomial running time
An IAQS $\mathcal{M}$ has *reliable polynomial running time* or *reliably polynomially bounded running time*, if there is a polynomial p and a negligible bound ε, such that for any quantum interaction ι it is

$$P(T^{\mathrm{X}}_{k,n} < p(k+n) \text{ or } T^{\mathrm{X}}_{k,n} = \perp) > 1 - \varepsilon(k)$$

for all $n \in \mathbb{N}$, $k \in \mathbb{N}_0$ with X being the execution transcript on ι. □

Definition 3.8: Expected polynomial running time
Then a IAQS $\mathcal{M}$ has *expected polynomial running time* or *expected polynomially bounded running time*, if there is a polynomial p, such that for any quantum interaction ι, it is $ET^{\mathrm{X}}_{n,k} \leq p(t)$ or $P(T^{\mathrm{X}}_{n,k} = \perp) = 1$ for all $n \in \mathbb{N}$, $k \in \mathbb{N}_0$, where X is the execution transcript on ι. □

3.3.3 Terminating IAQS

As with polynomial running time, we have several possibilities to define termination:

- *Strict termination*: It is impossible, that the machine does not terminate.
- *Weak termination*: It has probability 0, that the machine does not terminate.
- *Reliable termination*: It has negligible probability that the machine does not terminate.

In this case—other than with polynomial running time—the concepts of impossibility and vanishing probability differ. Therefore both the strict and the weak termination are analogues to the strict polynomial running time. An example for an weakly terminating machine, which is not strictly terminating would be one, which in every step stops with a probability of $\frac{1}{2}$. It is of course possible, that this machine will run forever, but the probability of this is 0.

Here the three definition form a strict hierarchy: Strict termination is stronger than weak one, which again is stronger than reliable one.

Again we will take the strict termination as the default.

We will now formalise those three notions:

Definition 3.9: (Strict) termination
Let $H^{\complement}$ be the set of all execution transcripts (x_i), such that there exists an $n_0 \in \mathbb{N}$, such that for every $i \geq n_0$ it is $x_i = (run, \lambda)$.

We then call an IAQS *(strictly) terminating*, if $H^{\complement}$ is impossible on any quantum interaction. □

Here $H^{\complement}$ can be thought of as the set of all non-terminating execution transcripts.

Definition 3.10: Weak termination
Let $H^{\complement}$ be as in the previous definition.

We then say that an IAQS is *weakly terminating*, if for any quantum interaction ι it is $P(\mathrm{X} \in H^{\complement}) = 0$, where X is the execution transcript on ι. □

Definition 3.11: Reliable termination
Let $H^{\complement}$ be as in the foregoing two definitions and K_k as in the definition of running time (definition 3.5)

We then say that an IAQS is *reliably terminating*, if for any quantum interaction ι and any $k \in \mathbb{N}_0$, the probabilities $(P(\mathrm{X} \in H^{\complement} \cap K_k))_k$ are negligible in k, where X is the execution transcript on ι. □

It is $H^{\complement}_k$ the set of all execution transcripts where the first received frame has length k (or does not exist if $k = 0$), and which additionally do not terminate.

3.3.4 Relative polynomial running time

Besides classification of the running time of a single machine, we may also compare running times. An especially important notion is that of relative polynomial running times, i.e. we want to formalise statements like "$\mathcal{M}_2$ has a running time polynomial in that of $\mathcal{M}_1$".

We will now proceed by giving two possible definitions of relative polynomial running time. Informally they go as follows:

- *(Hard) relative polynomial running time:* For any invocation and any security parameter, let T_1 be the worst-case running time of $\mathcal{M}_1$. The worst case running time T_2 of $\mathcal{M}_2$ shall then be polynomial in T_1.

- *Stochastic relative polynomial running time:* For any invocation and any security parameter, let T_1 be the distribution of the running time of $\mathcal{M}_1$, and T_2 that of $\mathcal{M}_2$. Then the probability, that T_2 exceeds some value t_2, should be bounded by the probability, that $p(T_1)$ exceeds t_2.

The formalisations of these notions are:

Definition 3.12: (Hard) relative polynomial running time
Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be IAQS. Let further ι_1 and ι_2 be quantum interactions for $\mathcal{M}_1$ resp. $\mathcal{M}_2$ and X_1 and X_2 execution transcripts for $\mathcal{M}_1$ resp. $\mathcal{M}_2$ on ι_1 resp. ι_2.

We then say, $\mathcal{M}_2$ *has (hard) polynomial running time relative to* $\mathcal{M}_1$, if the following holds for any choice of ι_1, ι_2:

For $i = 1, 2$ let $T_{i,k,n} \in \mathbb{N}$ be minimal with $P(T_{k,n}^{\mathrm{X}_i} > T_{i,k,n}) = 0$, resp. $T_{i,k,n} := \perp$ if $P(T_{k,n}^{\mathrm{X}_i} = \perp) = 1$. Then there is a polynomial p independent of the choice of ι_1, ι_2, such that $T_{2,k,n} \leq p(T_{1,k,n} + k + n)$ for every $k \geq 0$, $n \geq 1$, where we consider the inequality as true, if $T_{1,k,n} = \perp$ or $T_{2,k,n} = \perp$. □

Note that we have given $T_{1,k,n} + k + n$ as argument to the polynomial. This is due to the fact, that if we have e.g. $\mathcal{M}_1$ with constant running time, then being polynomial just in $T_{1,k,n}$ would imply that $\mathcal{M}_2$ has constant running time, too. With our definition however, $\mathcal{M}_2$ may then be of polynomial running time, since its running time can also grow polynomially in k and n.

Definition 3.13: Stochastic relative polynomial running time
Let $\mathcal{M}_1$, $\mathcal{M}_2$, ι_1, ι_2, X_1, X_2 be as in the previous definition.

We then say, $\mathcal{M}_2$ *has stochastic polynomial running time relative to* $\mathcal{M}_1$, if the following holds for any choice of ι_1, ι_2:

There is a polynomial p independent of ι_1 and ι_2, such that

$$P(T_{k,n}^{\mathrm{X}_2} \geq t) \leq P(p(T_{k,n}^{\mathrm{X}_1} + k + n) \geq t) \tag{14}$$

for all $n \in \mathbb{N}$, $k \in \mathbb{N}_0$, $t \in \mathbb{N}_0$. We consider the inequality to be true, if $T_{k,n}^{\mathrm{X}_1} = \perp$ or $T_{k,n}^{\mathrm{X}_2} = \perp$. □

The main advantage of the hard relative polynomial time is that is probably much simpler to handle and more intuitive, further it is probably approximately this definition people have in mind, when they say that some machine has a running time polynomially bounded in that of some other.

The advantage of the stochastic relative polynomially time is that it takes finer details of the two run time distributions into account. E.g. if $\mathcal{M}_1$ has a running time distribution $P(T = t) \propto 2^{-t}$, this would not be distinguished from a non-terminating machine by the first definition, the second however would require $\mathcal{M}_2$ to have—somewhat unprecisely spoken—a running time distribution in which the probabilities for higher running times diminish in some way exponentially. And if $\mathcal{M}_1$ is of strict polynomial time, the set of machines with stochastic polynomial running time relative to $\mathcal{M}_1$ would be exactly the set of machines of strict polynomial running time. If $\mathcal{M}_1$ is of reliable polynomial time, then any machine with stochastic polynomial running time relative to $\mathcal{M}_1$ is of reliable polynomial time, too.

However, if $\mathcal{M}_1$ is of expected polynomial time, then $\mathcal{M}_2$ may have stochastic polynomial running time relative to $\mathcal{M}_1$ without being of expected polynomial time. The counterexample given after the definition of expected polynomial time is valid here, too (see definition 3.8).

Another advantage of these definitions is, that if some IAQS $\mathcal{M}_2$ simulates another IAQS $\mathcal{M}_1$ with polynomial overhead, $\mathcal{M}_2$ is of hard and of stochastic polynomial running time relative to $\mathcal{M}_1$.

If $\mathcal{M}_2$ is of stochastic polynomial running time relative to $\mathcal{M}_1$, then it is also of hard polynomial running time relative to $\mathcal{M}_1$, so stochastic relative polynomial time is the stricter definition.

3.4 Classical Systems

If we do not only want to speak about quantum, but also about classical security, we need some notions what the word "classical" means in our context. This is achieved by the following definition:

Definition 3.14: classical
Let $U_{\text{copy},\mathcal{M}}$ be the operator on $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C$, which is defined by

$$U_{\text{copy},\mathcal{M}} : |\text{Ѫ}(q, r_1, r_2, \ldots)\rangle \otimes |c\rangle \otimes |f\rangle \longrightarrow |\text{Ѫ}(q, q, c, f, r_1, r_2, \ldots)\rangle \otimes |c\rangle \otimes |f\rangle$$

for $q, r_i, c \in \Sigma_{\text{fin}}^{\mathbb{Z}}$, $f \in \Sigma^*$.

We call an IAQS $\mathcal{M}$ classical, if

- there are $\tilde{U}_{\mathcal{M}} \in U(\mathcal{H}_{Q,\mathcal{M}})$, $\tilde{U}_{\text{FS},\mathcal{M}} \in U(\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F)$, $\tilde{U}_{\text{S},\mathcal{M}} \in U(\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_C)$, $\tilde{U}_{\text{R},\mathcal{M}} \in U(\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C)$, $\tilde{U}_{\text{corr},\mathcal{M}} \in (\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_C)$, such that

$$\begin{aligned}
U_{\mathcal{M}} &= \tilde{U}_{\mathcal{M}} \cdot U_{\text{copy},\mathcal{M}}, \\
U_{\text{FS},\mathcal{M}} &= U_{\text{copy},\mathcal{M}} \cdot \tilde{U}_{\text{FS},\mathcal{M}} \cdot U_{\text{copy},\mathcal{M}}, \\
U_{\text{S},\mathcal{M}} &= U_{\text{copy},\mathcal{M}} \cdot \tilde{U}_{\text{S},\mathcal{M}} \cdot U_{\text{copy},\mathcal{M}}, \\
U_{\text{R},\mathcal{M}} &= \tilde{U}_{\text{R},\mathcal{M}} \cdot U_{\text{copy},\mathcal{M}}, \\
U_{\text{corr},\mathcal{M}} &= U_{\text{copy},\mathcal{M}} \cdot \tilde{U}_{\text{corr},\mathcal{M}} \cdot U_{\text{copy},\mathcal{M}}.
\end{aligned}$$

- for $q, r_1, \tilde{r}_1, r_2, \tilde{r}_2, \ldots \in \Sigma_{\text{fin}}^{\mathbb{Z}}$ with $r_i \neq \tilde{r}_i$ for some $i \in \mathbb{N}$ it is

$$\big(\langle \text{Ѫ}(q, r_1, r_2, \ldots)| \otimes \mathbb{1} \otimes \mathbb{1}\big) M \big(|\text{Ѫ}(\tilde{q}, \tilde{r}_1, \tilde{r}_2, \ldots)\rangle \otimes \mathbb{1} \otimes \mathbb{1}\big) = 0 \tag{15}$$

  for each $M \in \{\tilde{U}_{\mathcal{M}} \otimes \mathbb{1} \otimes \mathbb{1}, \tilde{U}_{\text{FS},\mathcal{M}} \otimes \mathbb{1}, \tilde{U}_{\text{S},\mathcal{M}} \otimes \mathbb{1}, \tilde{U}_{\text{R},\mathcal{M}}, \tilde{U}_{\text{corr},\mathcal{M}} \otimes \mathbb{1}, P_{\text{F},\mathcal{M}} \otimes \mathbb{1} \otimes \mathbb{1}\}$.

□

The ideas behind this definition are the following: The configuration space of $\mathcal{M}$ is split into an infinite sequence of spaces using the Ѫ-function, each again containing a complete tape, i.e.

$$\mathcal{H}_{Q,\mathcal{M}} \cong \mathcal{H}_{Q,\mathcal{M},1} \otimes \mathcal{H}_{Q,\mathcal{M},2} \otimes \mathcal{H}_{Q,\mathcal{M},3} \otimes \ldots.$$

Now we defined all operators operating on $\mathcal{H}_{Q,\mathcal{M}}$ to be preceded in every application by a copy operation, which pushes a copy of $\mathcal{H}_{Q,\mathcal{M},1}$, $\mathcal{H}_F$ and $\mathcal{H}_C$ (in the standard basis) onto the stack of tapes $\mathcal{H}_{Q,\mathcal{M},2}$, $\mathcal{H}_{Q,\mathcal{M},3}$,... (we will call these the measurement tapes). Doing this is equivalent to measuring $\mathcal{H}_{Q,\mathcal{M},1}$ in the standard basis. The operators which write onto the external tapes are also followed by this copy operation.

To ensure, that the measurement tapes are not modified any more, we force all operators (excepting the just described copying operation) to be read-only on the measurement tapes. This is done by (15).

Note that this in fact defines probabilistic machines, though there is no random tape, since the IAQS is allowed to do unitary transformations, which are not necessarily permutations of basis vectors, and since this is followed by a "measurement" afterwards, we can get random results.[8]

That these constraints are in fact sufficient to make a machine classical, is demonstrated by the following lemma:

Lemma 3.15
Let $\mathcal{M}$ be a classical IAQS and ι some quantum interaction for $\mathcal{M}$ consisting of operators (O_i).

Let $M_i : \mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^2 \to \mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^2$ $(i = 1, \ldots, n+1)$ be observables with eigenspaces spanned by vectors of the standard basis of $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^2$, where $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes |0\rangle$ should be completely contained in one of the eigenspaces.

We define the mixed states

$$\varrho_1 := \ddot{M}_r \ddot{M}_f \ddot{M}^{(i)} \prod_{i=1}^{n} (\ddot{O}_i \ddot{M}_r \ddot{M}_f \ddot{M}^{(i)})\, \varrho(|\Psi_{0,\mathcal{M}}\rangle \otimes |\#\rangle \otimes |\#\rangle \otimes |0\rangle \otimes |0\rangle),$$

$$\varrho_2 := \ddot{M}_r \ddot{M}_f \ddot{M}^{(i)} \ddot{M}_{n+1} \prod_{i=1}^{n} (\ddot{O}_i \ddot{M}_r \ddot{M}_f \ddot{M}^{(i)} \ddot{M}_i)\, \varrho(|\Psi_{0,\mathcal{M}}\rangle \otimes |\#\rangle \otimes |\#\rangle \otimes |0\rangle \otimes |0\rangle),$$

[8] E.g. a random bit could be obtained by applying the Hadamard transform on a single qubit, after measuring it will have the value 0 or 1 with equal probability.

in $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^2$, where M_f is the standard basis measurement of $\mathcal{H}_F$, M_r that of $\mathbb{C}^2$, and $M^{(i)}$ is the measurement $\{P_k^{(i)}, k \in \{\mathit{run}, \mathit{send}, \mathit{rec}, \mathit{world}\}\}$, with $P_k^{(i)}$ as in the definition of quantum interactions (definition 3.2). $\ddot{X}$ denotes the density matrix operator for X.

Let M be any observable satisfying for each of its eigenspaces E, that

$$(\langle\tilde{q}| \otimes \langle\tilde{f}| \otimes \langle\tilde{c}| \otimes \mathbb{1} \otimes \mathbb{1}) P_E (|q\rangle \otimes |f\rangle \otimes |c\rangle \otimes \mathbb{1} \otimes \mathbb{1}) = 0$$

for any basis states $|\tilde{q}\rangle, |q\rangle \in \mathcal{H}_{Q,\mathcal{M}}$, $|\tilde{f}\rangle, |f\rangle \in \mathcal{H}_F$, $|\tilde{c}\rangle, |c\rangle \in \mathcal{H}_C$ with $|\tilde{q}\rangle \otimes |\tilde{f}\rangle \otimes |\tilde{c}\rangle \neq |q\rangle \otimes |f\rangle \otimes |c\rangle$, where P_E is the projection onto E.

Then the outcomes of M-measuring ϱ_1 and ϱ_2 have the same probability distribution. □

This lemma can be interpreted as follows: Each M_i is a classical observation of the machines configuration and its external tapes; however if the machine is not running (i.e. we have a state in $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes |0\rangle\rangle$), the measurement always yields the same value. The latter is necessary for formal reasons, because in our modelling of the quantum interaction the external tapes may temporarily be used by the external world in a quantum way while the machine is not running.

Then we defined two mixed states. The first, ϱ_1, is the result of applying the quantum interaction similar to the way already described in the definition of the execution transcript. But instead of projecting onto measurement results chosen in advance, we perform unobserved measurements. Furthermore we end in a measurement, not with an application of O_i.

The second one, ϱ_2, we get by applying the same operations as when creating ϱ_1, but by measuring the machine's classical data with the measurements M_i before and after every time step.

Note that the order of the measurements does not matter, since they do all commute.

The result of this lemma is that we cannot distinguish ϱ_1 and ϱ_2 by applying a measurement that behaves classically on the machine's configuration and the external tapes, i.e. which satisfies, that any basis vector in those spaces stays unmodified after the measurement.

The whole formalises the intuitive notion that the machine is classical and thus justifies the above definition.

A proof for this lemma can be found in section A.1.

It is not only important to speak about completely classical machines, we also have to formalise the notion of machines, which can do quantum calculations, but which cannot communicate with each other in a quantum manner. These machines can be thought as a classical computer having access to some internal quantum computer. Such a machine can obviously be simulated by a classical probabilistic computer with an exponential overhead with arbitrary precision. We have named this kind of IAQS an IAQS with *classical interfaces.*

Definition 3.16: Classical interfaces
Let $U_{\mathrm{copyif},\mathcal{M}}$ be the unitary operator on $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C$, which is defined by

$$U_{\mathrm{copyif},\mathcal{M}} : |\text{Ѫ}(q, r_1, r_2, \ldots)\rangle \otimes |c\rangle \otimes |f\rangle \longrightarrow |\text{Ѫ}(q, c, f, r_1, r_2, \ldots)\rangle \otimes |c\rangle \otimes |f\rangle$$

for $q, r_i, c \in \Sigma_{\mathrm{fin}}^{\mathbb{Z}}$, $f \in \Sigma^*$.

We say an IAQS $\mathcal{M}$ has *classical interfaces,* if

- There are some $\tilde{U}_{\mathrm{S},\mathcal{M}} \in U(\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_C)$, $\tilde{U}_{\mathrm{FS},\mathcal{M}} \in \mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F$, and $\tilde{U}_{\mathrm{R},\mathcal{M}} \in \mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C$, such that

$$\begin{aligned} U_{\mathrm{S},\mathcal{M}} &= U_{\mathrm{copyif},\mathcal{M}} \cdot \tilde{U}_{\mathrm{S},\mathcal{M}}, \\ U_{\mathrm{FS},\mathcal{M}} &= U_{\mathrm{copyif},\mathcal{M}} \cdot \tilde{U}_{\mathrm{FS},\mathcal{M}}, \\ U_{\mathrm{R},\mathcal{M}} &= \tilde{U}_{\mathrm{R},\mathcal{M}} \cdot U_{\mathrm{copyif},\mathcal{M}}, \end{aligned}$$

- and for $q, r_1, \tilde{r}_1, r_2, \tilde{r}_2, \ldots \in \Sigma_{\mathrm{fin}}^{\mathbb{Z}}$ with $r_i \neq \tilde{r}_i$ for some $i \in \mathbb{N}$ it is

$$(\langle\text{Ѫ}(q, r_1, r_2, \ldots)| \otimes \mathbb{1} \otimes \mathbb{1}) M (|\text{Ѫ}(\tilde{q}, \tilde{r}_1, \tilde{r}_2, \ldots)\rangle \otimes \mathbb{1} \otimes \mathbb{1}) = 0$$

  for each $M \in \{U_{\mathcal{M}} \otimes \mathbb{1} \otimes \mathbb{1}, \tilde{U}_{\mathrm{FS},\mathcal{M}} \otimes \mathbb{1}, \tilde{U}_{\mathrm{S},\mathcal{M}} \otimes \mathbb{1}, \tilde{U}_{\mathrm{R},\mathcal{M}}, U_{\mathrm{corr},\mathcal{M}} \otimes \mathbb{1}, P_{\mathrm{F},\mathcal{M}} \otimes \mathbb{1} \otimes \mathbb{1}\}$.

□

This definition can be understood quite analogous to that of classical IAQS. The only difference is, that we do only copy the content of the external tapes onto our measurement tapes, and that this copy operation only takes place immediately after send and frame send operations, and immediately before receive operations, because only then the external tapes contain something to be measured.

We will now introduce two mappings from the set of IAQS to the set of classical IAQS resp. the set of IAQS with classical interfaces. These shall formalise the result of taking an IAQS $\mathcal{M}$ and measuring the the external tapes and in the first case also the configuration after each time step. The second one we will need in chapter 8 when discussing the equivalence of classical and quantum security in some cases. The first one is given for completeness.

Definition 3.17: Making an IAQS classical (mapping $\mathfrak{C}$)
The mapping

$$\mathfrak{C} : \mathrm{IAQS} \longrightarrow \mathrm{IAQS}$$

maps an IAQS $\mathcal{M}$ to a classical IAQS $\mathcal{C} := \mathfrak{C}(\mathcal{M})$ constructed as follows:

$$M_{\mathcal{C}} : |\text{Ѫ}(q, r_1, r_2, \ldots)\rangle|\Psi\rangle|\Phi\rangle \longmapsto \sum_{\tilde{q}\in\Sigma_{\mathrm{fin}}^{\mathbb{Z}}} (\langle\tilde{q}|\langle\Psi|\langle\Phi|M_{\mathcal{M}}|q\rangle|\Psi\rangle|\Phi\rangle)|\text{Ѫ}(\tilde{q}, r_1, r_2, \ldots)\rangle|\Psi\rangle|\Phi\rangle \tag{16}$$

for $q, r_1, r_2, \ldots \in \Sigma_{\mathrm{fin}}^{\mathbb{Z}}$, $|\Psi\rangle \in \mathcal{H}_F$, $|\Phi\rangle \in \mathcal{H}_C$, and for every $(M_{\mathcal{C}}, M_{\mathcal{M}})$ in

$$\begin{aligned}\{(\tilde{U}_{\mathcal{C}} \otimes \mathbb{1} \otimes \mathbb{1}, U_{\mathcal{M}} \otimes \mathbb{1} \otimes \mathbb{1}), \qquad & (\tilde{U}_{\mathrm{FS},\mathcal{C}} \otimes \mathbb{1}, U_{\mathrm{FS},\mathcal{M}} \otimes \mathbb{1}),\\ (\tilde{U}_{\mathrm{S},\mathcal{C}} \otimes \mathbb{1}, U_{\mathrm{S},\mathcal{M}} \otimes \mathbb{1}), \qquad & (\tilde{U}_{\mathrm{R},\mathcal{C}}, U_{\mathrm{R},\mathcal{M}})\}.\end{aligned}$$

Let then

$$\begin{aligned}\tilde{U}_{\mathrm{corr},\mathcal{C}} : {} & |\text{Ѫ}(q, r_1, r_2, \ldots)\rangle \otimes |\text{Ѫ}(q', r'_1, r'_2, \ldots)\rangle \\ & \longmapsto \sum_{\tilde{q},\tilde{q}'\in\Sigma_{\mathrm{fin}}^{\mathbb{Z}}} (\langle\tilde{q}| \otimes \langle\tilde{q}'|U_{\mathrm{corr},\mathcal{M}}|q\rangle \otimes |q'\rangle)|\text{Ѫ}(\tilde{q}, r_1, r_2, \ldots)\rangle \otimes |\text{Ѫ}(\tilde{q}', r'_1 \oplus r_1, r'_2 \oplus r_2, \ldots)\rangle\end{aligned} \tag{17}$$

and $P_{\mathrm{F},\mathcal{C}}$ be the projector onto

$$\mathrm{span}\{\sum_{i\in I} \alpha_i|\text{Ѫ}(q_i, r_1, r_2, \ldots)\rangle : q_i, r_i \in \Sigma_{\mathrm{fin}}^{\mathbb{Z}}, P_{\mathrm{F},\mathcal{M}} \sum_{i\in I} \alpha_i|q_i\rangle = \sum_{i\in I} \alpha_i|q_i\rangle, \#I < \infty\}.$$

The $U_{\mathcal{C}}$, $U_{\mathrm{FS},\mathcal{C}}$, $U_{\mathrm{S},\mathcal{C}}$, $U_{\mathrm{R},\mathcal{C}}$ and $U_{\mathrm{corr},\mathcal{C}}$ operators are then constructed as in the definition of classical IAQS. □

This mapping constructs the send, frame send, and receive operator of $\mathcal{C}$ thus, that they operate on $\mathcal{H}_{Q,\mathcal{C},1}$, as the operators of $\mathcal{M}$ would have done on $\mathcal{H}_{Q,\mathcal{M}}$ (equation (16)). These operators are then completed by the invocations of $U_{\mathrm{copy},\mathcal{C}}$ required by the definition of classical IAQS.

The corruption operator is defined such, that $\mathcal{H}_C$ is decomposed into $\mathcal{H}_{C,1} \otimes \mathcal{H}_{C,2} \otimes \cdots$, and then the $U_{\mathrm{corr},\mathcal{C}}$ operates on $\mathcal{H}_{Q,\mathcal{C},1} \otimes \mathcal{H}_{C,1}$, as $U_{\mathrm{corr},\mathcal{M}}$ would have done one $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_C$, and copies $\mathcal{H}_{Q,\mathcal{C},i}$ to $\mathcal{H}_{C,i}$ ($i \geq 2$). So this corruption operator delivers all information about past measurements, but does not allow to undo any measurement, therefore the information is copied onto the communication tape, but not removed from the measurement tapes. This behaviour is specified by (17). Finally the calls to $U_{\mathrm{copy},\mathcal{C}}$ necessitated by the definition of classical IAQS are added to $U_{\mathrm{corr},\mathcal{C}}$.

The end configuration projection $P_{\mathrm{F},\mathcal{C}}$ projects $\mathcal{H}_{Q,\mathcal{C},1}$, as $P_{\mathrm{F},\mathcal{M}}$ would have done with $\mathcal{H}_{Q,\mathcal{M}}$. The measurement tapes are ignored for the decision, whether we have reached an end configuration.

To get the mapping onto the set of IAQS with classical interfaces, only few modifications of the previous definition are necessary:

Definition 3.18: Making the interfaces of an IAQS classical (mapping $\mathfrak{C}_{\mathfrak{I}}$)
The mapping

$$\mathfrak{C}_{\mathfrak{I}} : \mathrm{IAQS} \longrightarrow \mathrm{IAQS}$$

maps an IAQS $\mathcal{M}$ to an IAQS $\mathcal{C} := \mathfrak{C}_{\mathfrak{I}}(\mathcal{M})$ constructed as follows:

The operators $\tilde{U}_{\mathcal{C}}$, $\tilde{U}_{\mathrm{FS},\mathcal{C}}$, $\tilde{U}_{\mathrm{S},\mathcal{C}}$, $\tilde{U}_{\mathrm{R},\mathcal{C}}$, $\tilde{U}_{\mathrm{corr},\mathcal{C}}$ and $P_{\mathrm{F},\mathcal{C}}$ are constructed as in the preceding definition.

Then $U_{\mathrm{FS},\mathcal{C}}$, $U_{\mathrm{S},\mathcal{C}}$, and $U_{\mathrm{R},\mathcal{C}}$ are defined as in the definition of classical interfaces, and

$$U_{\mathcal{C}} := \tilde{U}_{\mathcal{C}}, \qquad U_{\mathrm{corr},\mathcal{C}} := \tilde{U}_{\mathrm{corr},\mathcal{C}}.$$

□

This definition is almost the same as that of $\mathfrak{C}$, with the exception, that we have added invocations of $U_{\text{copyif},\mathcal{C}}$ instead of $U_{\text{copy},\mathcal{C}}$, and only there, where it is required by the definition of classical interfaces, i.e. in $U_{\text{FS},\mathcal{C}}$, $U_{\text{S},\mathcal{C}}$, and $U_{\text{R},\mathcal{C}}$.

There are of course other conceivable models, which we will not formalise, but which should be mentioned here:

- We could model machines which have no quantum storage, this could be done by measuring the machines content after reaching the end configuration, but not after each step.
- We could also model machines, which do not have full quantum interfaces, but which can choose in what basis they want to send their *classical* data, and in which basis to measure incoming data (here we could further distinguish, whether we have to specify a basis for each cell of the input or a basis for the whole incoming tape).

  This models e.g. a quantum channel realised by photons which we may polarise and measure using filters which can be placed in any orientation, but without the possibility to let some photons interact with each other, and without any quantum storage.
- The corruption operator does not copy the measurement tapes, but swaps them with the communication tape. This models the case, that real classical computation is impossible, and that existing classical machines just hide their measurement tapes in the laboratories thermal activity or the like, and that a sufficiently skillful adversary may isolate those "lost states" again and operate on them.

  This probably is not a very present threat, but should nevertheless be thought of.

3.5 Turing machines

3.5.1 Interactive quantum Turing machines

In this section we will introduce interactive quantum Turing machines and how to regard them as IAQS. We will strongly draw from [BV93], but introduce important changes due to the interactive nature of those Turing machines.

To show which are the crucial points when making a quantum Turing machine interactive, we will first recollect how a classical interactive Turing machine is usually constructed. They are basically normal multi-tape Turing machines, where some tapes have special meaning, the communication tapes. If there is some incoming message, it is written onto or appended to the incoming communication tape; if a message shall be send, it is copied from the outgoing communication tape.

In our case, we have to restrictions, which obviate such a simple solution: We want to model the impossibility of data erasure, and we need unitary operations due to the quantum nature of our model.

The handling of outgoing messages can easily be fixed with respect to these restrictions, instead of copying the outgoing message, we move it off the outgoing tape, leaving the latter in a cooled state.

In the case of incoming messages, the world gets complicated. We cannot overwrite[9] the incoming communication tape, since then we would erase data, but appending is not feasible, either. Though appending is a well-defined operation even in quantum mechanics, we run into the following problem: When we have two messages on the incoming communication tape, we cannot read the second one without knowing the length of the first one. It may however be conceivable, that some protocol inhibits measuring the length of some messages, and in the general case there might not even be known bounds on the message length.

So apparently there is only one way to solve this problem: We have to add the incoming message as a new tape into the Turing machine.[10] So the interactive quantum Turing machines presented here have a finite but dynamically growing number of tapes.

Since the state transition function cannot operate on a variable number of tapes,[11] we introduce the concept of a tape revolver, which is part of our Turing machine in addition to the normal tapes. This is a cyclic sequence of tapes, of which exactly one is in use at a time. We can turn this revolver in two directions, thus navigating to any one of the tapes. The state transition function then operates on the normal tapes and the selected revolver tape. Thus the transition function is finite and can nevertheless access a growing number of tapes.

[9] Here the word *overwriting* also covers the case, where we move the content of the incoming communication tape to some inaccessible location and then put the new date in its place.

[10] In fact, we could also interleave the message with one of the existing tapes, but this is essentially the same, since we add new cells to the Turing machine.

[11] Otherwise it would be possible to encode an infinite amount of information into the transition function, allowing to solve even non-computable problems.

In this Turing machine, receiving a message is implemented by inserting frame and message as two new tapes into the revolver, while when sending a message, we take the active revolver tape as frame, and the following one as the message data.[12] Note that we move the content off the tape when sending as described above, but we do not remove that tape, since this would imply mapping the number of the active tape onto some number of a smaller set, an intrinsically non-reversible operation.

We will now proceed and give the formal details of quantum Turing machines.

Definition 3.19: Computable real and complex numbers $\tilde{\mathbb{R}}$, $\tilde{\mathbb{C}}$
We call $\tilde{\mathbb{R}}$ the set of all numbers $\alpha \in \mathbb{R}$, such that there is a deterministic algorithm A_α, that computes α to within a precision of 2^{-n} in time polynomial in n.

Let then $\tilde{\mathbb{C}} := \tilde{\mathbb{R}} + i\tilde{\mathbb{R}}$ and $\tilde{\mathbb{R}}_{\geq 0} := \tilde{\mathbb{R}} \cap \mathbb{R}_{\geq 0}$. □

We will restrict the amplitudes in the definition of an IQTM to such computable numbers to ensure, that we cannot hide some not efficiently computable information in the amplitudes. See [BV93] for more information on this.

Definition 3.20: Interactive quantum Turing machine (IQTM)
An *interactive quantum Turing machine (IQTM)* $\mathcal{M} = (Q, q_0, q_f, n, \delta)$ consists of a finite set of *states* $Q \subseteq \Sigma^*$, an *initial state* $q_0 \in Q$, a *final state* $q_f \in Q$, a *tape number* $n \geq 2$ and a transition function

$$\delta : Q \times \Sigma^n \longrightarrow \tilde{\mathbb{C}}^{\Sigma^n \times Q \times \{L,R\}^{n-1} \times \{L,R,U,D\}}.$$

□

The interpretation of the transition function δ is as follows: It takes as input the current state $q \in Q$ and the symbols at the current head positions in the n different tapes. Here the n-th tape is the active revolver tape. Each revolver tape has its own head position.

We consider a state $q \in Q$ as a string ($Q \subset \Sigma^*$) in order to be able to write the configuration of a machine as a string (see definition 3.28).

The output of δ is a superposition of transitions, each consisting of n new symbols to be written at the current head positions, a new state $q' \in Q$ and n directions for head movements. Here the n-th direction, which applies to the revolver can either move the head of the active revolver tape (L, R), or turn the revolver up or down (U, D).

Definition 3.21: Time evolution of an IQTM
The *time evolution $U_\mathcal{M}$ of an IQTM* $\mathcal{M} = (Q, q_0, q_f, n, \delta)$ is a linear operator on $\mathcal{H}_{Q,\mathcal{M}}$, which is defined as follows:

Let $\delta(q, \sigma_1, \ldots, \sigma_n, q', \tau_1, \ldots, \tau_n, d_1, \ldots, d_n) := \langle q', \tau_1, \ldots, \tau_n, d_1, \ldots, d_n | \delta(q, \sigma_1, \ldots, \sigma_n)$ and let $u(n)$ be as in definition 2.17.

For any basis state of $\mathcal{H}_{Q,\mathcal{M}}$ having the form $|\text{Ӿ}(q, u(r_m), u(r), \mu, u(h_1), t_1, u(h_2), t_2, \ldots)\rangle$ with $q \in Q$, $r_m \in \mathbb{N}$, $r_m \geq 3$, $r \in \{0, \ldots, r_m - 1\}$, $\mu \in \Sigma^*$, $h_i \in \mathbb{Z}$, $t_i \in \Sigma_{\text{fin}}^{\mathbb{Z}}$, and $h_i = 0$, $t_i = \#$ for $i > n + r_m - 1$, it is

$$\begin{aligned} &U_\mathcal{M} |\text{Ӿ}(q, u(r_m), u(r), \mu, (u(h_i), t_i)_i)\rangle \\ &= \sum \delta(q, (t_{i,h_i})_{i=1}^{n-1}, t_{n+r,h_{n+r}}, q', (\tau_i)_{i=1}^n, (d_i)_{i=1}^n) |\text{Ӿ}(q', u(r_m), u(r + \bar{d}_n \bmod r_m), \mu', (u(h_i + \tilde{d}_i), \tilde{t}_i)_i)\rangle \end{aligned} \tag{18}$$

where $\tilde{t}_i$ is the tape resulting from the tape t_i by replacing the symbol at index h_i by τ_i ($i < n$) resp. τ_{r+n} ($i = n$). For $i \geq n$, $i \neq r + n$, it is $\tilde{t}_i = t_i$. Let $\mu' := \mu$.

Further let be

$$(\tilde{d}_i, \bar{d}_i) := \begin{cases} (0,0), & \text{if } i \geq n \text{ and } i \neq r+n, \\ (-1,0), & \text{if } d_i = L \ (i < n) \text{ or } d_n = L \ (i = r+n), \\ (1,0), & \text{if } d_i = R \ (i < n) \text{ or } d_n = R \ (i = r+n), \\ (0,-1), & \text{if } d_n = U \ (i = r+n), \\ (0,1), & \text{if } d_n = D \ (i = r+n). \end{cases}$$

The sum in (18) runs over all $q' \in Q$, $\tau_1, \ldots, \tau_n \in \Sigma$, $d_1, \ldots, d_{n-1} \in \{L, R\}$, $d_n \in \{L, R, U, D\}$.

For any basis state $|\Psi\rangle \in \mathcal{H}_{Q,\mathcal{M}}$ which has not the form above, it is $U_\mathcal{M}|\Psi\rangle = |\Psi\rangle$. □

[12] It would be possible to specify, that for sending normal tapes are used, but when choosing revolver tapes, we can resend a received message without having to copy it onto another tape, the latter being impossible in the case of a message of unknown length.

In this definition, we embed the space $\mathcal{H}_{\mathrm{IQTM}} := \mathbb{C}^Q \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^{\Sigma^*} \otimes (\mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^{\Sigma^{\mathbb{Z}}_{\mathrm{fin}}})^{\otimes \mathbb{N}}$ in $\mathcal{H}_{Q,\mathcal{M}}$. Our embedding is not surjective, outside of this subspace we define $U_{\mathcal{M}}$ arbitrary to be the identity. States outside $\mathcal{H}_{\mathrm{IQTM}}$ will never be reached.

A basis state in $\mathcal{H}_{\mathrm{IQTM}}$ consists of the automaton's state q, the size of the revolver r_m, the number r of the current revolver tape, some string μ which will be ignored,[13] and tapes t_i ($i \in \mathbb{N}$) together with head positions $h_i \in \mathbb{Z}$.

The tapes $t_1, \dots, t_{n-1}$ are the $n-1$ normal tapes, the tapes $t_n, \dots, t_{n+r_m-1}$ are the revolver tapes, of which t_{r+n} is the currently selected one.

We assume that $r_m \geq 3$ (otherwise moving the revolver in different directions will give the same position, which might lead to problems with the unitarity) and, of course, $r \in \{0, \dots, r_m - 1\}$. Otherwise we operate as the identity.

For any given state, we can calculate the value of δ, which is $\delta(q, (t_{i,h_i})_{i=1}^{n-1}, t_{n+r,h_{n+r}})$. This value is a superposition of actions $(q', (\tau_i)_{i=1}^n, (d_i)_{i=1}^n)$. Here q' is the automaton's new state, τ_i are the symbols to write to the current head positions, and d_i are the head and revolver movements to apply afterwards.

The new tape contents $\tilde{t}_i$ are then constructed by replacing the cell at position h_i in t_i by the new symbol τ_i, if it is a normal tape. For the current revolver tape $\tilde{t}_{r+n}$, we replace by τ_n, of course, since there is no τ_{r+n}. Inactive revolver tapes are not modified.

The head positions are then shifted by $\tilde{d}_i$. Here $\tilde{d}_i$ is directly derived from d_i for normal tapes. In case of the current revolver tape d_n may have four values R, L, U, D. In case of R or L, we proceed as with the normal tapes to calculate $\tilde{d}_{r+n}$. But for U or D, the is no head movement, but $\bar{d}_i$, the movement of the revolver, is set to -1 resp. 1. For inactive tapes, no movement occurs ($\tilde{d}_i = 0$).

If $\bar{d}_i$ was set to a nonzero value, the revolver position r is moved by $\bar{d}_i$ modulo the revolver size.

This time evolution is in general not unitary. Therefore we will concentrate on those IQTM, where it is unitary, the well-formed ones:

Definition 3.22: Well-formedness of an IQTM
An IQTM is *well-formed,* if its time evolution is unitary.

The set of all well-formed IQTM is denoted IQTM.

Throughout this work, we will always implicitly assume well-formedness when talking about IQTM. □

Definition 3.23: Start configuration of an IQTM
The *start configuration* $|\Psi_{0,\mathcal{M}}\rangle$ of an IQTM $\mathcal{M} = (Q, q_0, q_f, n, \delta)$ is defined by

$$|\Psi_{0,\mathcal{M}}\rangle := |\text{Ѫ}(q_0, u(3), \#, \#, \#, \dots)\rangle. \qquad \square$$

In the start configuration, the state is the initial state q_0, the revolver size is set to its minimal size, all head positions are zero ($u(0) = \#$), and all tapes are empty.

Definition 3.24: End configuration projection of an IQTM
The *end configuration projection* $P_{\mathrm{F},\mathcal{M}}$ of an IQTM $\mathcal{M} = (Q, q_0, q_f, n, \delta)$ is the projection onto

$$\operatorname{span}\{|\text{Ѫ}(q_f, x_1, x_2, x_3, \dots)\rangle : x_1, x_2, \dots \in \Sigma^{\mathbb{Z}}_{\mathrm{fin}}, x_i = \# \text{ for almost every } i\}. \qquad \square$$

The condition for being an end configuration is that the current state q is the final state q_f.

Definition 3.25: Frame send operator of an IQTM
The *frame send operator* $U_{\mathrm{FS},\mathcal{M}}$ of an IQTM $\mathcal{M} = (Q, q_0, q_f, n, \delta)$ operates on $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F$ as follows:

$$U_{\mathrm{FS},\mathcal{M}}|\text{Ѫ}(q, u(r_m), u(r), \mu, (u(h_i), t_i)_i)\rangle \otimes |f\rangle = |\text{Ѫ}(q, u(r_m), u(r+1 \bmod r_m), \mu, (u(h_i), t_i)_i)\rangle \otimes |f \oplus \text{Љ}(t_{r+n})\rangle.$$

On basis vectors not having that form, or where $r \notin \{0, \dots, r_m - 1\}$, $U_{\mathrm{FS},\mathcal{M}}$ operates as the identity. □

The frame send operator takes the current revolver tape, copies a string from its content (i.e. from head position 0 to the first #-symbol) to the frame tape $|\mathrm{frame}\rangle$. Copying is feasible here, since we assume $|\mathrm{frame}\rangle$ to contain classical data.

The revolver is then moved one position, since the send operator, which is often executed directly after the frame send operator, should not use the same tape, of course.

Definition 3.26: Send operator of an IQTM
The *send operator* $U_{\mathrm{S},\mathcal{M}}$ of an IQTM $\mathcal{M} = (Q, q_0, q_f, n, \delta)$ operates on $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_C$ as follows:

$$U_{\mathrm{S},\mathcal{M}}|\text{Ѫ}(q, u(r_m), u(r), \mu, (u(h_i), t_i)_i)\rangle \otimes |c\rangle = |\text{Ѫ}(q, u(r_m), u(r+1 \bmod r_m), \mu, (u(h_i), \tilde{t}_i)_i)\rangle \otimes |t_{r+n}\rangle,$$

[13] It will be used when defining interactive classical Turing machines in section 3.5.2. It is included in the definition of IQTM to have more uniform definitions.

where $\tilde{t}_i := t_i$ if $i \neq r+n$ and $\tilde{t}_i := c$ if $i = r+n$. On basis vectors not having that form, or where $r \notin \{0, \ldots, r_m - 1\}$, $U_{\mathrm{S},\mathcal{M}}$ operates as the identity. □

The send operator works similar to the frame send operator, except that the content of the current revolver tape is swapped, not copied, with the communication tape.

Definition 3.27: Receive operator of an IQTM
The *receive operator* $U_{\mathrm{FS},\mathcal{M}}$ of an IQTM $\mathcal{M} = (Q, q_0, q_f, n, \delta)$ operates on $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C$ as follows:

$$U_{\mathrm{R},\mathcal{M}}|\mathfrak{K}(q, u(r_m), u(r), \mu, (u(h_i), t_i)_i)\rangle \otimes |f\rangle \otimes |c\rangle = |\mathfrak{K}(q, u(r_m+2), u(r), \mu, (u(\tilde{h}_i), \tilde{t}_i)_i\rangle \otimes |\#\rangle \otimes |\#\rangle,$$

where

$$(\tilde{h}_i, \tilde{t}_i) := \begin{cases} (h_i, t_i), & \text{if } i < r+n, \\ (0, f), & \text{if } i = r+n, \\ (0, c), & \text{if } i = r+n+1, \\ (h_{i-2}, t_{i-2}), & \text{if } i \geq r+n+2. \end{cases}$$

□

In the case of the receive operator, the number of revolver tapes r_m is increased by 2, and the new two tapes are inserted at the current revolver position. The new tapes have the content of the frame tape and the communication tape respectively. The frame and the communication tape are left empty after this operation, as required by the definition of IAQS.

Definition 3.28: Corruption operator of an IQTM
The *corruption operator* $U_{\mathrm{corr},\mathcal{M}}$ of an IQTM $\mathcal{M} = (Q, q_0, q_f, n, \delta)$ operates on $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_C$ as $U_{\mathrm{corr},\mathcal{M}}|q\rangle \otimes |c\rangle = |c\rangle \otimes |q\rangle$. □

The corruption operator gives all information about the IQTM to the corrupting party.

With those operators, an IQTM is an IAQS, so we have an embedding IQTM $\subseteq$ IAQS.

Definition 3.29: Polynomial interactive quantum Turing machine
An well-formed IQTM, which has polynomial running time, we call a *polynomial interactive quantum Turing machine (PIQTM).*

The set of all PIQTM we call PIQTM. □

Lemma 3.30: Local well-formedness condition for IQTM
An IQTM $\mathcal{M} = (Q, q_0, q_f, n, \delta)$ is well-formed, if and only if the following conditions are fulfilled for δ:

- Unit length: $\|\delta(q, \sigma_1, \ldots, \sigma_n)\| = 1$ for all $q \in Q$, $\sigma_1, \ldots, \sigma_n \in \Sigma$.
- Orthogonality: $\delta(q, \sigma_1, \ldots, \sigma_n) \perp \delta(\tilde{q}, \tilde{\sigma}_1, \ldots, \tilde{\sigma}_n)$ for any $q, \tilde{q} \in Q$, $\sigma_i, \tilde{\sigma}_i \in \Sigma$ with $(q, \sigma_1, \ldots, \sigma_n) \neq (\tilde{q}, \tilde{\sigma}_1, \ldots, \tilde{\sigma}_n)$.
- Separability: For any $I \subseteq \{1, \ldots, n\}$, $I \neq \emptyset$ it is: For $\sigma_i, \tilde{\sigma}_i, \sigma'_i, \tilde{\sigma}'_i \in \Sigma$, $q, \tilde{q} \in Q$, $d_i, \tilde{d}_i \in \{L, R\}$ $(i < n)$, $d_n, \tilde{d}_n \in \{L, R, U, D\}$, $d_i \neq \tilde{d}_i$, it is

$$\sum_{q'} \sum \delta(q, (\sigma_i)_i, (\underline{\sigma}'_i)_i, q', (\underline{d}_i)_i)^\dagger \delta(\tilde{q}, (\tilde{\sigma}_i)_i, (\underline{\tilde{\sigma}}'_i)_i, q', (\underline{\tilde{d}}_i)_i) = 0$$

  where the second sum runs over all $\underline{\sigma}'_i, \underline{\tilde{\sigma}}'_i \in \Sigma$, $\underline{d}_i, \underline{\tilde{d}}_i \in \{L, R\}$ $(i < n)$, $\underline{d}_n, \underline{\tilde{d}}_n \in \{L, R, U, D\}$, which satisfy for $i \in I$: $\sigma'_i = \underline{\sigma}'_i$, $\tilde{\sigma}'_i = \underline{\tilde{\sigma}}'_i$, $d_i = \underline{d}_i$, $\tilde{d}_i = \underline{\tilde{d}}_i$; and for $i \notin I$: $\underline{\sigma}'_i = \underline{\tilde{\sigma}}'_i$, $\underline{d}_i = \underline{\tilde{d}}_i$.

Here δ is used as in the definition of the time evolution (definition 3.21). □

This lemma is the natural extension of the local well-formedness condition for quantum Turing machines presented in [BV93] to our model of IQTM. When removing the revolver and restricting the number of tapes to 1, then our lemma specialises to the well-formedness condition given there.

A proof for this lemma is given in section A.6.

3.5.2 Interactive classical Turing machines

Beside IQTM, we also need interactive classical Turing machines (ICTM) to model participants, which are not able to do anything non-classical.

The classical Turing machine has a state transition function very similar to that of a quantum Turing machine, except that the image is not a superposition of actions, but a probability distribution.

Definition 3.31: Interactive classical Turing machine (ICTM)
An *interactive classical Turing machine (ICTM)* (Q, q_0, q_f, n, δ) consists of a finite set of *states* $Q \subseteq \Sigma^*$, an *initial state* $q_0 \in Q$, a *final state* $q_f \in Q$, a *tape number* $n \geq 2$ and a transition function

$$\delta : Q \times \Sigma^n \longrightarrow \tilde{\mathbb{R}}_{\geq 0}^{\Sigma^n \times Q \times \{L,R\}^{n-1} \times \{L,R,U,D\}}.$$

where for any $x \in \operatorname{im} \delta$ the sum of all components of x is 1.

The set of all ICTM is written ICTM. □

We want to consider an ICTM as an IAQS, and we have the additional requirement, that it matches the definition of a classical IAQS (definition 3.14). In order to achieve this, we first define an intermediate IAQS $\mathcal{C}'$, which has the desired behaviour, and then make it classical using the mapping $\mathfrak{C}$ (definition 3.18).

Definition 3.32: Pseudo time evolution of an ICTM
The *pseudo time evolution* $U_{\mathcal{C}'}$ *of an ICTM* $\mathcal{C} = (Q, q_0, q_f, n, \delta)$ is defined almost exactly as the time evolution of an IQTM $\mathcal{C}' := (Q, q_0, q_f, n, \delta')$ with

$$\delta'(q, \sigma_1, \dots, \sigma_n) := \sum \sqrt{\delta(q, \sigma_1, \dots, \sigma_n)_{\tau_1, \dots, \tau_n, q', d_1, \dots, d_n}}\, |\tau_1, \dots, \tau_n, q', d_1, \dots, d_n\rangle$$

Here the sum runs over all $(\tau_1, \dots, \tau_n, q', d_1, \dots, d_n) \in \Sigma^n \times Q \times \{L, R\}^{n-1} \times \{L, R, U, D\}$.

The only difference is, that

$$\mu' := \mathbb{Q}(\mu, \text{Ж}(q, q', (t_{i,h_i})_{i=1}^{n+r_m-1}, d_1, \dots, d_n, \tau_1, \dots, \tau_n)). \qquad \square$$

This time evolution is analogous to that of an IQTM, with the exception, that the string μ gets all information appended, which is needed to

- Undo the last time step.
- Re-execute the last time step if only the contents of μ are known.

These two properties make $U_{\mathcal{C}'}$ unitary, since due to the first condition, two different basis states go onto orthogonal states and due to the second property a basis state goes into a superposition of orthogonal states, each with a squared amplitude equal to the corresponding value of δ, which sum up to 1.

Definition 3.33: An ICTM as an IQTM
An ICTM $\mathcal{C}$ is considered as an IQTM $\mathcal{M}$, which is constructed as follows:

Let $\mathcal{C}'$ be the IAQS consisting of the pseudo time evolution of $\mathcal{C}$, and the end configuration projection, the frame send operator, the send operator, the receive operator and the corruption operator as defined for the (not necessarily well-formed) IQTM $\mathcal{C}$.

Then $\mathcal{M} := \mathfrak{C}(\mathcal{C}')$. □

Because $U_{\mathcal{C}'}$ is unitary, $\mathcal{C}'$ is well-formed, and so is $\mathcal{M}$, its image under $\mathfrak{C}$. We thus have ICTM $\subset$ IQTM.

3.5.3 Classical interfaces

The last variant of Turing machines we need is that of quantum Turing machine with classical interfaces (IQCTM).

These are easy to construct using the mapping $\mathfrak{C}_{\mathfrak{I}}$. We simply construct a normal IQTM and then make the interfaces classical by measuring them. See the explanation of $\mathfrak{C}$ in the previous section.

Definition 3.34: Interactive quantum Turing machine with classical interfaces (IQCTM)
An *interactive quantum Turing machine with classical interfaces (IQCTM)* is defined exactly as is an IQTM. □

Definition 3.35: An IQCTM as a IAQS
An IQCTM $\mathcal{C}$ is considered to be an IAQS $\mathcal{M}$, which is constructed as follows:

Let $\mathcal{C}'$ be the IAQS belonging to the IQTM $\mathcal{C}$. Then $\mathcal{M} := \mathfrak{C}_{\mathfrak{J}}(\mathcal{C}')$. □

Definition 3.36: Well-formedness
We call an IQCTM *well-formed,* if it is a well-formed IAQS.
The set of all well-formed IQCTM we call IQCTM. □

3.6 ITM and partial ITM

Mostly, if we want to allow a class of Turing machines in some definition (e.g. when restricting to polynomial Turing machines), we do not care which kind of interactive Turing machines are used, i.e. whether IQTM, ICTM, or IQCTM. Therefore we define the interactive Turing machine as follows:

Definition 3.37: Interactive Turing machine (ITM)
An *interactive Turing machine (ITM)* is an IQTM, an ICTM, or an IQCTM.
The set of all well-formed ITM we denote as ITM $\subset$ IAQS. The set of all polynomial well-formed ITM we write PITM $\subset$ ITM. □

At some point we will also need partial Turing machines, these are defined straightforwardly as:

Definition 3.38: Partial interactive Turing machine
A *partial interactive Turing machine (partial ITM)* is defined as an IQTM, ICTM or IQCTM, except that the state transition function is allowed to be a partial function.
The *restriction of an ITM $\mathcal{M} = (Q, q_0, q_f, n, \delta)$ to a set of states Q'* is a partial ITM $\mathcal{M} = (Q, q_0, q_f, n, \delta')$, where δ' obtained by restricting δ to $Q' \times \Sigma^n$. □

There is no natural interpretation of a partial ITM as an IAQS.

4 Quantum Network Models

4.1 Quantum networks

In this chapter we will introduce a model for quantum networks. It is based on the model of IAQS described in the previous chapter and shall be as general a model as possible, not only restricted to purposes of cryptology. Therefore we do not include notions like the adversary or the environment in this model, nor do we give a specific scheduling mechanism. This will be done in the specialisation described in section 5.1 where the adversarial quantum network is introduced. We will however already include primitive operations like corruption at this point.

Definition 4.1: Quantum Network (QN)
A *quantum network* is a countable set of well-formed IAQS $(\mathcal{N}_i)_{i \in I_{\mathcal{N}}}$ and a set of machine IDs $I_{\mathcal{N}} \subseteq \Sigma^*$ together with a scheduling function

$$\mathrm{Sched}_{\mathcal{N}} : \Sigma^* \longrightarrow E^* \times I_{\mathcal{N}} \times \Sigma^*.$$

Here the elements of E^* are finite, possibly empty sequences of events, each of which can be:

- a *transmission event* containing a source s and a destination d from $I_{\mathcal{N}}$, and a new frame $f \in \Sigma^*$, such an event is written $(\mathit{transmit}, s, d, f)$,
- a *control event* containing a destination $d \in I_{\mathcal{N}}$ and a new frame $f \in \Sigma^*$, such an event is written $(\mathit{control}, d, f)$,
- a *corruption event* containing a corrupting party c and a corruption target t, both from $I_{\mathcal{N}}$, and a new frame $f \in \Sigma^*$, such an event is written $(\mathit{corrupt}, c, t, f)$.

□

The scheduling function gets as input a string containing information on the state and history of the quantum network (the exact content of which will be specified below when defining the time evolution of a QN). The output is a list of actions to be taken, i.e. delivering a message (transmission event) or corrupting a party (corruption event), the token, i.e. the ID of the machine to be activated next, and a string to be given as part of the input to the next invocation of $\mathrm{Sched}_{\mathcal{N}}$.

The transmission event comes together with the source and the destination of the message and further with a new frame which the recipient of the message gets. So the recipient may get wrong or no information concerning e.g. the sender of the message. Note that the message itself is not contained in the event, after issuing the event the actual message will be read from the sender using the send operator $U_{\mathrm{S},\mathcal{M}}$.

The control event only has a destination and a new frame. It is meant for classical notifications and control messages sent from the system to individual parties, e.g. the security parameter is given to the parties via a control event.

The corruption event has a corrupting party which will get the information of the corrupted party, and the target which is to be corrupted. The corruption event will cause the transmission of information from the target to the corrupting party, but it will not influence the transmission of messages to or from the target later on or hinder the further activation of the target. So if it is wanted that the target is activated no more and all communication rerouted to the corrupting party, the scheduling function has to implement this. This is doable, since the scheduling function decides who gets the activation token. Further the scheduling function may reroute messages by just setting the source or destination to appropriate values and changing the new frame thus, that the recipient will not notice this rerouting.

Outstanding a formal definition, we will give here a partial list, which are the informations the scheduling function can extract of its input:

- What machines have been activated in which order so far?
- In particular: Which one was activated last?
- Which were the running times of those machines?
- What frame was uttered by which machine after which activation?
- Which decisions and outputs did the scheduling function make during past invocations?

4.2 Time evolution

The above definition of a quantum network is still quite meaningless. We need to know how it evolves over the time. Therefore we will now specify what happens in a single time step.

First we need to know in what space our network operates and in which state it begins:

Definition 4.2: Configuration space and start configuration of a QN
The start configuration of an QN $\mathcal{N}$ is given as

$$|\Psi_{0,\mathcal{N}}\rangle := |\#\rangle \otimes |\#\rangle \otimes |\#\rangle \otimes \bigotimes_{i \in I_{\mathcal{N}}} (|\Psi_{0,\mathcal{N}_i}\rangle \otimes |\#\rangle)$$

which is an element of

$$\mathcal{H}_{\mathcal{N}} := \mathcal{H}_{G,\mathcal{N}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \bigotimes_{i \in I_{\mathcal{N}}} (\mathcal{H}_{Q,\mathcal{N}_i} \otimes \mathcal{H}_{H,\mathcal{N}_i}).$$

The symbols $|\Psi_{0,\mathcal{N}_i}\rangle$, $\mathcal{H}_{Q,\mathcal{N}_i}$, $\mathcal{H}_F$, $\mathcal{H}_C$ denote the start configuration, the configuration space, the frame tape space resp. the communication space of machine $\mathcal{N}_i$ as defined in section 3.1. Three new Hilbert spaces appear above: The networks *configuration space* $\mathcal{H}_{\mathcal{N}}$, the *machine history tape space* of $\mathcal{N}_i$, which is defined as $\mathcal{H}_{H,\mathcal{N}_i} := \mathbb{C}^{\Sigma^*}$, and the *global history tape space* of $\mathcal{N}$, given by $\mathcal{H}_{G,\mathcal{N}} := \mathbb{C}^{\Sigma^*}$. □

We see that the configuration of a QN consists of one storage for a "global history", the (shared) communication and frame tapes, and for each machine of its state and a storage for a "machine history".

The purpose of the global history is to store all information which gets "measured" during the execution of the quantum network. Since we do no assume an external measurement process (but see section 5.5), we have to simulate the measurement by copying all information which is taken as classical onto the global history tape. There is another purpose of this history: the scheduling function gets its content as input and can thus base on all past measured information to decide which actions to invoke.

The local history serves another purpose. At different times some measurements are done on the IAQS, e.g. it is measured whether the machine has terminated, what frame it has issued, etc. Since we want to model systems without reliable erasure (resp. its weakened quantum variant as discussed in 1.7) a machine could use the following trick to circumvent this restriction: Assume some number, coded unary, at a known position on some tape. The machine goes to the end of that number, starts erasing 1's until a # is found. Then it terminates. Every step is reversible, so the time evolution of this machine is in fact unitary. But the end configuration is independent of the number. What has happened? The information is not lost, but it is now encoded in the running time T of the machine. We have to apply the inverse time evolution T times to get the initial state. Since we do not want to enable the machine to forget information using this method, we write informations like the running time onto the history tape and make this tape accessible to a corrupting party.

We will now first describe the *time evolution* $U_{\mathcal{N}}$ *of a quantum network* and give the formal but quite complex definition in section A.4.

Let

$$|G\rangle \otimes |\text{frame}\rangle \otimes |\text{comm}\rangle \otimes \bigotimes_{i \in I_{\mathcal{N}}} (|\Psi_{\mathcal{N}_i}\rangle \otimes |\text{hist}_i\rangle)$$

be the state of the QN, where we assume everything exept $|\Psi_{\mathcal{N}_i}\rangle$ to be a base state.[14] Then one step of the time evolution is given by application of the following steps:

1. Find the last datum of the form Ж(`sched`, *events*, *token*, *data*) on $|G\rangle$ and set $i :=$ *token*. If there is no such datum, continue with step 9.
2. Let $U_{\mathcal{N}_i}$ operate on $|\Psi_{\mathcal{N}_i}\rangle$.
3. Measure, whether $|\Psi_{\mathcal{N}_i}\rangle$ is an end configuration (using $P_{\mathrm{F},\mathcal{N}_i}$). We will call the result of this measurement F_i.
4. Append `endconf:`F_i to $|G\rangle$ and $|\text{hist}_i\rangle$.
5. If $F_i = 0$ ("not an end configuration"), do not execute the following steps, but append `nosched` to $|G\rangle$.[15]

[14] If the state is actually a superposition of thus formed states, just apply the time evolution to each summand.
[15] We append `nosched`, so that step 10 by itself is unitary. It has no relevance for the intuitive meaning of this algorithm.

6. Apply $U_{\mathrm{FS},\mathcal{N}_i}$ to $|\Psi_{\mathcal{N}_i}\rangle \otimes |\mathrm{frame}\rangle$.
7. Append `frame:`$|\mathrm{frame}\rangle$ to $|G\rangle$ and $|\mathrm{hist}_i\rangle$.
8. Set $|\mathrm{frame}\rangle$ to an empty string.
9. Evaluate $(\mathit{events}, \mathit{token}, \mathit{data}) := \mathrm{Sched}_{\mathcal{N}}(|G\rangle)$.
10. Append Ж(`sched`, *events-txt*, *token*, *data*) to $|G\rangle$.

    Here *events-txt* is the textification of *events*, defined via $(e_1, \ldots, e_n) :=$ *events*, $(t_i, c_{i1}, \ldots, c_{im}) := e_i$, $t'_i :=$ `transmit`, `corrupt` or `control`, depending on t_i, $e'_i := (t'_i, c_{i1}, \ldots, c_{im})$ and *events-txt* $:=$ Ж$(e'_1, \ldots, e'_n)$.
11. For each event e in *events* evaluate it as described below.

A transmission event we evaluate as follows:

1. Let *source* be the source of the event and *dest* its destination. *newframe* shall denote the new frame.
2. Let $U_{\mathrm{S},\mathcal{N}_{source}}$ operate on $|\Psi_{\mathcal{N}_{source}}\rangle \otimes |\mathrm{comm}\rangle$.
3. Write *newframe* onto $|\mathrm{frame}\rangle$.
4. Let $U_{\mathrm{R},\mathcal{N}_{dest}}$ operate on $|\Psi_{\mathcal{N}_{dest}}\rangle \otimes |\mathrm{frame}\rangle \otimes |\mathrm{comm}\rangle$.

A control event is handled by:

1. Let *dest* be the destination of the event and *newframe* the the new frame.
2. Write *newframe* onto $|\mathrm{frame}\rangle$.
3. Let $U_{\mathrm{R},\mathcal{N}_{dest}}$ operate on $|\Psi_{\mathcal{N}_{dest}}\rangle \otimes |\mathrm{frame}\rangle \otimes |\mathrm{comm}\rangle$.

A corruption event is evaluated by the following steps:

1. Let *corrupter* be the corrupting party and *target* the corruption target. As above, *newframe* is the new frame.
2. Write *newframe* onto $|\mathrm{frame}\rangle$ and append $|\mathrm{hist}_{target}\rangle$ by copying.
3. Let $U_{\mathrm{corr},\mathcal{N}_{target}}$ operate on $|\Psi_{target}\rangle \otimes |\mathrm{comm}\rangle$.
4. Let $U_{\mathrm{R},\mathcal{N}_{corrupter}}$ operate on $|\Psi_{corrupter}\rangle \otimes |\mathrm{frame}\rangle \otimes |\mathrm{comm}\rangle$.

We will now explain the individual steps of the time evolution.

Step 1 finds out, which is the active machine. We do this by searching the last output of $\mathrm{Sched}_{\mathcal{N}}$, which we have stored on $|G\rangle$ in step 10. From now on i contains the ID of the active machine.

Step 2 lets the active machine operate one time step.

In step 3 we find out, whether the active machine $\mathcal{N}_i$ has halted. In step 4 the result of the "measurement" if stored on $|G\rangle$ for further use and to make this information classical. Note that the term "measurement" is not formally correct here, since all that is done is to let the execution of the following steps depend on whether $\mathcal{N}_i$ has halted. Since this is done in superposition, no actual measurement takes place.

Then, due to step 5, we only execute the rest of the steps, i.e. steps 6–11, if the machine has halted. The effect of this is that iterated application of the time evolution executes a loop, which is approximately described by the following pseudo-code:

```
while true do
  repeat
    time step of active machine
  until machine has terminated
  do scheduling, messages transmission, etc.
end while
```

In steps 6–8 we extract the frame out of the active machine, store it onto $|G\rangle$ and then erase the contents of $|\text{frame}\rangle$ again to make it free for future use. The frame the machine has sent is now stored on $|G\rangle$.

Steps 9 and 10 call the scheduling function and store its output onto $|G\rangle$. In our model storing the information on $|G\rangle$ does not make any difference, since the information is deterministically depend of the foregoing content of $|G\rangle$. We nevertheless include this step here to make it clearer, that $\text{Sched}_{\mathcal{N}}$ has access to the output of all its previous invocations, to make formulation of step 1 simpler, and to make it possible to extend the notion of the scheduling function to e.g. a probabilistic one.

In step 11 all the actions requested by $\text{Sched}_{\mathcal{N}}$ are invoked. Note that we do not do anything here with the token the scheduling function has yielded, this is done at the beginning of the next iteration by step 1.

When evaluating a transmission event, we extract the data to be sent from $\mathcal{N}_{source}$ via $U_{\text{S},\mathcal{N}_{source}}$ and transfer it to $\mathcal{N}_{dest}$ using $U_{\text{R},\mathcal{N}_{dest}}$, together with the *newframe* supplied by $\text{Sched}_{\mathcal{N}}$.

A control event differs from the transmission event in that we do not fetch data from a source, but instead leave $|\text{comm}\rangle$ unmodified (i.e. cooled), so that the only information gotten by the destination is that contained in the new frame.

When a corruption event is to be evaluated, we extract the state from $\mathcal{N}_{target}$ using $U_{\text{corr},\mathcal{N}_{target}}$ and send it, together with *newframe* and the history of $\mathcal{N}_{target}$ to $\mathcal{N}_{corrupter}$. The history is given to $\mathcal{N}_{target}$ as part of the frame, since it is classical information and we use the frame for such by convention.

A complete formal definition, together with a proof that the time evolution is unitary, can be found in section A.4.

4.3 Classical quantum networks

We have defined classical IAQS and IAQS with classical interfaces in section 3.4. So its is straightforward to define classical quantum networks and quantum networks with classical interfaces, i.e. quantum networks, that contain only machines with that property.

Definition 4.3: Classical quantum networks
We say a quantum network $\mathcal{N}$ is is *classical,* when all its machines $\mathcal{N}_i$ $(i \in I_{\mathcal{N}})$ are classical. □

Definition 4.4: Quantum networks with classical interfaces
We say a quantum network $\mathcal{N}$ has *classical interfaces,* when all its machines $\mathcal{N}_i$ $(i \in I_{\mathcal{N}})$ have classical interfaces. □

Note that it is enough to require classical behaviour (or classical interfaces) of the machines, since all matrices appearing in the time evolution of the quantum network, except those appertaining to the machines, are permutations of basis vectors.

5 Adversarial Communication

5.1 Adversarial quantum network

In the previous section we have developed the notion of a quantum network and of quantum communication. But this model is still too abstract for our needs, we need a model containing environment, adversary, parties, functionalities, some concrete scheduling, and a security parameter. Therefore we will now define the adversarial quantum network.

Definition 5.1: (Adaptive) adversarial quantum network (AQN)
An *(adaptive) adversarial quantum network (AQN)* $\mathcal{N}$ consists of a sequence of quantum networks $(\mathcal{N}^{(k)})_{k\in\mathbb{N}}$, which are identical except for the scheduling functions, and whose set of IDs is

$$I_{\mathcal{N}^{(k)}} = I_{\mathrm{adv}} = \{E, A\} \cup I_P \cup I_F,$$

where $I_P = \{P\}\cdot(\Sigma\backslash\#)^+$ and $I_F = \{F\}\cdot(\Sigma\backslash\#)^+$, further of an *access structure* $f_{\mathcal{N}} : I_P \times I_F \cup I_F \times I_P \to \{0,1\}$, and of a corruption structure $\mathfrak{A}_{\mathcal{N}} \subseteq 2^{I_P}$. The scheduling function $\mathrm{Sched}_{\mathcal{N}^{(k)}}$ is the adaptive adversarial scheduling function to the parameter k as defined below.

We call $f_{\mathcal{N}}$ a *full access structure,* if its image is $\{1\}$.

The set of all adversarial quantum networks is called AQN. □

Definition 5.2: Static adversarial quantum network (SAQN)
A *static adversarial quantum network (SAQN)* is defined exactly as the adaptive adversarial quantum network, except that the scheduling function is the static adversarial scheduling function. □

5.2 Scheduling

The scheduling is the point, in which we deviate most from the security model presented in [Can00b]—beside the fact that we allow quantum communication, of course. This is due to the fact, that we have chosen not to use message driven scheduling[16], but some kind of non-preemptive multitasking, i.e. the identity of the recipient of some message has no influence on the party which is activated next, instead the adversary is activated after any activation of some other machine to decide, which machine should get the token next. We have chosen this approach, since first this seems to model the reality more exactly, since in reality the delivery of messages does not influence the distribution of computational time; second, the message driven model tends to get unnecessarily complicated, since it has to have a rule, which machine is activated next for each possible action of any participant of the communication.

Another important point concerning the scheduling is ensuring that no polynomial party gets activated more than a polynomial number of times. The most apparent way to do this is to say, that a polynomial machine can be activated at most a polynomial number of times. But then we have to decide, what should happen after reaching that limit for some machine. If we say, that the protocol should stop, if some machine is activated too often, then even a polynomial adversary could halt the protocol just by activating some party too often (a polynomial number of times is sufficient, since the adversary has w.l.o.g. a greater polynomial to bound its running time).

The other alternative is to say that the machine behaves as a dummy machine after reaching the limit, in that way that it just gives back the token without sending a message or the like when activated again. Here the problem is, that the adversary could "kill" a machine by activating it too often. Image a protocol with parties A, B, C, which should implement a functionality communicating only with A and B. Assume further, that the protocol needs C to work. Than the environment could distinguish the real and the ideal protocol if the adversary "kills" C. This is not a desired property.

One could propose, that just environment and adversary should be limited in that way, not the parties and functionalities. But then it would be impossible to make statements like "there is no protocol running in polynomial time that implements...", since this would imply quantifying about polynomial parties.

Therefore we introduce another model to ensure a polynomial total step count: There is a variable in our model, call the *time block number*. This time block number can be increased by the adversary at will, provided that the adversary increases the time block after at most a polynomial number of invocations. Then we will

[16]A message driven scheduling is one, in which the question, which machine is to be activated next, is strongly influenced by the identity of the recipient of the message. A purely message driven approach would always activate the recipient of the message, in a security model however this is not feasible, since the adversary should be activated between the parties' invocations, resulting in some mixed model.

define security such, that security should be provided, when the model runs a polynomial number of time blocks, i.e. it should be secure for *any sufficiently large* polynomial. This is motivated by the fact, that in reality there is no hard limit on the running time of a protocol, up to which we want to maintain security, instead we want to ensure security for any reasonable (i.e. polynomial) running time.

But why do we not just restrict the protocol execution to a polynomial number of invocations instead of introducing the concept of time blocks? This is due to the fact, that the environment can find out, after how many of its activations the execution terminates, using the following trick: After the n-th invocation, the environment terminates with output 0 with a probability of $(n+1)^{-2}$. Then when terminating after the N-th invocation, the probability that the environment has not output 0, is $P_N := \prod_{n=1}^{N}(n+1)^{-2}$. So if in the real and the ideal model the environment is activated a different number of times, we get some different probabilities P_N for a non-zero output independent of the security parameter. Therefore real and ideal protocol are distinguishable only because the number of invocations between two invocations of the environment are different.

Further solving this problem using time blocks makes it easier to think of extensions of the model, where some machines are allowed to be invoked more than a polynomial number of times, while some others are not.

The restriction, that in any time block there are at most a polynomial number of activations, is ensured by the well-formedness of the adversary (see definition 5.5).

So the scheduling in our model goes as follows:

- The adversary is activated first.
- Whenever the adversary is activated, it may decide, which machine is activated next.
- If some machine other than the adversary was activated last, the adversary is activated next.
- Any machine which is activated for the first time, gets a frame containing the security parameter in unary notation.
- If the adversary requests this, the time block number is increased.
- The environment may transmit a message to some party or the adversary. If the recipient is corrupted, the message is rerouted to the adversary.
- The adversary may do one of the following:
  - Send a message to the environment or some functionality, indicating the correct sender.
  - Transmit a message, indicating some corrupted party as the sender. This is allowed in those situations, in which the party would have been allowed to send that message (see below).
  - Corrupt a party. This is only possible if afterwards the set of corrupted parties is an element of $\mathfrak{A}$. The environment is informed about this.
- A party may send a message to the environment or some functionality. If the recipient is a functionality, the party must have access to that functionality.
- A functionality may send a message to the adversary or to some party. If the recipient is a party, the functionality must have access to that party. If the recipient is corrupted, the message is rerouted to the adversary.

These points are realised by the following scheduling function:

Definition 5.3: Adversarial scheduling function
The *adversarial scheduling function to the parameter k*

$$\mathrm{Sched}_{\mathcal{N}^{(k)}} : \Sigma^* \longrightarrow E^* \times I_{\mathrm{adv}} \times \Sigma^*$$

is given the output of the following algorithm (the notation of the foregoing definition still applies):

1. Let *events* $:= \varepsilon$ (sequence of length 0), *token* $:= \perp$.

2. Extract from input:

lasttoken	ID of the machine which last had the token (or $\perp$).
corr	Set of corrupted parties.
timeblock	Time block (from data part of last invocation of $\mathrm{Sched}_{\mathcal{N}}$).
running	List of machines which ever have had the token.
frame	Frame sent by the last activated machine (or $\perp$).
token-req	Next token as requested in *frame* (or $\perp$, if *frame* is syntactically incorrect).
timeblock-req	Equals 1 if an increment of the time block is requested in *frame*, $\perp$ otherwise.
event-req	Event requested in *frame*, i.e. one of $\{\mathit{transmit}, \mathit{corrupt}, \perp\}$.
destination-req	Destination of the requested event (in case of *transmit*).
source-req	Source of the requested event (in case of *transmit*).
target-req	Target of the requested event (in case of *corrupt*).

This means formally:

- Let $k \in \mathbb{N}_0$ and $(a_i)_{i=1}^k$ $(a_i \in \Sigma^*)$ be such, that $\text{Ж}(a_1, \dots, a_k)$ is the input to the algorithm, or, if no such k, (a_i) exist, let $k := 0$.
- Let $(\text{Ж}(\mathtt{sched}, b_{i,ev}, b_{i,tok}, b_{i,dat}))_{i=1}^l$ be the subsequence of (a_i), that consists of all elements having the form $\text{Ж}(\mathtt{sched}, \cdot, \cdot, \cdot)$.
- If $l > 0$ and $b_{l,tok} \in I_{\mathrm{adv}}$, then let $\mathit{lasttoken} := b_{l,tok}$. Otherwise set $\mathit{lasttoken} := \perp$.
- Let *corr* be the set of all $p \in I_P$, which satisfy, that there is an i, s.t. $b_{i,ev}$ has the form $\text{Ж}(\cdot, \dots, \cdot, \text{Ж}(\mathit{corrupt}, \cdot, p, \cdot), \cdot, \dots, \cdot)$.
- Let *timeblock* be such, that $b_{l,dat} = \text{Ж}(\mathtt{timeblock}, u(\mathit{timeblock}))$. Let $\mathit{timeblock} := 0$, if there is no value of *timeblock* satisfying that condition. Here u is defined as in definition 3.21.
- Let $\mathit{running} := \{m \in I_{\mathrm{adv}} : \exists i \in \{1, \dots, l\} : b_{i,tok} = m\}$.
- If there is an element of (a_i) having the form $\text{Ж}(\mathtt{frame}, \cdot)$, then let $\text{Ж}(\mathtt{frame}, \mathit{frame})$ be the last such element. Otherwise let $\mathit{frame} := \perp$.
- Let $(f_i)_{i=1}^m$ be such, that $\text{Ж}(f_1, \dots, f_m) = \mathit{frame}$. If this is not possible, set $m := 0$.
- If there is an element of (f_i) having the form $\text{Ж}(\mathtt{token}, \cdot)$, than let $\text{Ж}(\mathtt{token}, \mathit{token\text{-}req})$ be the first such element. Otherwise let $\mathit{token\text{-}req} := \perp$.
- If there is an element of (f_i) having the form $\mathtt{timeblock}$, than let $\mathit{timeblock\text{-}req} := 1$, otherwise $\mathit{timeblock\text{-}req} := 0$.
- If there is an element of (f_i) having the form $\text{Ж}(\mathtt{event}, \cdot)$, than let $\text{Ж}(\mathtt{event}, e)$ be the first such element. If $e = \mathtt{transmit}$ or $e = \mathtt{corrupt}$, set *event-req* to *transmit* resp. *corrupt*. In all other cases let $\mathit{event\text{-}req} := \perp$.
- If there is an element of (f_i) having the form $\text{Ж}(\mathtt{destination}, \cdot)$, than let $\text{Ж}(\mathtt{destination}, \mathit{destination\text{-}req})$ be the first such element. Otherwise let $\mathit{destination\text{-}req} := \perp$.
- If there is an element of (f_i) having the form $\text{Ж}(\mathtt{source}, \cdot)$, than let $\text{Ж}(\mathtt{source}, \mathit{source\text{-}req})$ be the first such element. Otherwise let $\mathit{source\text{-}req} := \perp$.
- If there is an element of (f_i) having the form $\text{Ж}(\mathtt{target}, \cdot)$, than let $\text{Ж}(\mathtt{target}, \mathit{target\text{-}req})$ be the first such element. Otherwise let $\mathit{target\text{-}req} := \perp$.

3. If $\mathit{timeblock\text{-}req} = 1$, increase *timeblock* by 1.

4. Handle the token as follows:

   (a) If $\mathit{last} = A$, $\mathit{token\text{-}req} \neq \perp$, and $\mathit{token\text{-}req} \notin \mathit{corr}$, set $\mathit{token} := \mathit{token\text{-}req}$.

   (b) Otherwise set $\mathit{token} := A$.

   (c) If $\mathit{token} \notin \mathit{running}$, append
   $$(\mathit{control}, \mathit{token}, \text{Ж}(\mathtt{secparam}, \mathtt{1}^k))$$
   to *events*, where $\mathtt{1}^k$ denotes the unary representation of the security parameter k.

5. In the following table, find that row, which matches *last* and whose conditions are satisfied, and append the corresponding events to *events*. If no row matches, leave *events* unchanged.

last	conditions	events	
$= E$	$\mathit{event\text{-}req} = \mathit{transmit}$ $\mathit{destination\text{-}req} \in \{A\} \cup I_P \setminus \mathit{corr}$	$(\mathit{transmit}, E, \mathit{destination\text{-}req},$ $\text{Ж}(\mathtt{message}, E, \mathit{destination\text{-}req}))$	
	$\mathit{event\text{-}req} = \mathit{transmit}$ $\mathit{destination\text{-}req} \in \mathit{corr}$	$(\mathit{transmit}, E, A,$ $\text{Ж}(\mathtt{message}, E, \mathit{destination\text{-}req}))$	
$= A$	$\mathit{event\text{-}req} = \mathit{transmit}$ $\mathit{destination\text{-}req} \in \{E\} \cup I_F$ $\mathit{source\text{-}req} = A$	$(\mathit{transmit}, A, \mathit{destination\text{-}req},$ $\text{Ж}(\mathtt{message}, A, \mathit{destination\text{-}req}))$	
	$\mathit{event\text{-}req} = \mathit{transmit}$ $\mathit{source\text{-}req} \in \mathit{corr}$ $\mathit{destination\text{-}req} = E$	$(\mathit{transmit}, A, E,$ $\text{Ж}(\mathtt{message}, \mathit{source\text{-}req}, E))$	
	$\mathit{event\text{-}req} = \mathit{transmit}$ $\mathit{source\text{-}req} \in \mathit{corr}$ $\mathit{destination\text{-}req} \in I_F$ $f_{\mathcal{N}}(\mathit{last}, \mathit{destination\text{-}req}) = 1$	$(\mathit{transmit}, A, \mathit{destination\text{-}req},$ $\text{Ж}(\mathtt{message}, \mathit{source\text{-}req}, \mathit{destination\text{-}req}))$	
	$\mathit{event\text{-}req} = \mathit{corrupt}$ $\{\mathit{target\text{-}req}\} \cup \mathit{corr} \in \mathfrak{A}$ $\mathit{target\text{-}req} \notin \mathit{corr}$ $\mathit{token} \neq \mathit{target\text{-}req}$	$(\mathit{corrupt}, A, \mathit{target\text{-}req},$ $\text{Ж}(\mathtt{corrupted}, \mathit{target\text{-}req})),$ $(\mathit{control}, E, \text{Ж}(\mathtt{corrupted}, \mathit{target\text{-}req}))$	$(*)$
$\in I_P$	$\mathit{event\text{-}req} = \mathit{transmit}$ $\mathit{destination\text{-}req} = E$	$(\mathit{transmit}, \mathit{last}, E,$ $\text{Ж}(\mathtt{message}, \mathit{last}, E))$	
	$\mathit{event\text{-}req} = \mathit{transmit}$ $\mathit{destination\text{-}req} \in I_F$ $f_{\mathcal{N}}(\mathit{last}, \mathit{destination\text{-}req}) = 1$	$(\mathit{transmit}, \mathit{last}, \mathit{destination\text{-}req},$ $\text{Ж}(\mathtt{message}, \mathit{last}, \mathit{destination\text{-}req}))$	
$\in I_F$	$\mathit{event\text{-}req} = \mathit{transmit}$ $\mathit{destination\text{-}req} = A$	$(\mathit{transmit}, \mathit{last}, A,$ $\text{Ж}(\mathtt{message}, \mathit{last}, A))$	
	$\mathit{event\text{-}req} = \mathit{transmit}$ $\mathit{destination\text{-}req} \in I_P \setminus \mathit{corr}$ $f_{\mathcal{N}}(\mathit{last}, \mathit{destination\text{-}req}) = 1$	$(\mathit{transmit}, \mathit{last}, \mathit{destination\text{-}req},$ $\text{Ж}(\mathtt{message}, \mathit{last}, \mathit{destination\text{-}req}))$	
	$\mathit{event\text{-}req} = \mathit{transmit}$ $\mathit{destination\text{-}req} \in I_P \cap \mathit{corr}$ $f_{\mathcal{N}}(\mathit{last}, \mathit{destination\text{-}req}) = 1$	$(\mathit{transmit}, \mathit{last}, A,$ $\text{Ж}(\mathtt{message}, \mathit{last}, \mathit{destination\text{-}req}))$	

6. Return $(\mathit{events}, \mathit{token}, \text{Ж}(\mathtt{timeblock}, \mathit{timeblock}))$.

□

Definition 5.4: Static adversarial scheduling function
The *static adversarial scheduling function* is defined exactly like the adaptive one, except that the row in the table of step 5 marked with $(*)$ has the additional condition $\mathit{running} \subseteq \{A, E\}$. □

This additional constraint has the effect, that the adversary may not corrupt any party after some party or functionality has been activated, i.e. only at the very beginning or the protocol execution.

Another, stricter definition of static could have the condition $\mathit{running} \subseteq \{A\}$, i.e. the adversary must decide whom to corrupt even before communicating with the environment.

5.3 Adversary types

As we have seen in the preceding section, the adversary is responsible for the order of activation in the adversarial quantum network. Since it should further be impossible, that a polynomially bounded adversary gets de facto more than polynomial time by activating itself very often without increasing the number of the time block, we should restrict the adversaries to such ones, which are "fair" in this respect, i.e. the adversary should be invoked at most a polynomial number of times between two increases of the time block number.

Definition 5.5: Well-formedness of an adversary
For any polynomial p let $F_{k,p}$ be the set of execution transcripts $\chi = (x_i)$ which satisfy both of the following conditions:

- preceding the n-th element of χ having the form $(\mathit{stop}, \text{Ж}(\cdot, \ldots, \cdot, \texttt{timeblock}, \cdot, \ldots, \cdot))$, there are at most $p(k+n)$ elements of the form (stop, f) with $f \in \Sigma^*$,
- there is an infinite number of elements formed $(\mathit{stop}, \text{Ж}(\cdot, \ldots, \cdot, \texttt{timeblock}, \cdot, \ldots, \cdot))$ in χ,

and one of

- for the first element of χ having the form (rec, f) the word f has length k,
- there is no such element and $k = 0$.

Let E be the set of all execution transcripts.

We then say an adversary $\mathcal{A} \in \text{IAQS}$ is *well-formed*, if there is a polynomial p, such that $E \setminus \bigcup_{k \in \mathbb{N}_0} F_{k,p}$ is impossible. □

Here $F_{k,p}$ is the set of execution transcripts, in which the first frame has length k and there the number of invocations of the adversary until the n-th time block is bounded by $p(k+n)$. So $E \setminus \bigcup_{k \in \mathbb{N}_0} F_{k,p}$ denotes transcripts which violate the informal well-formedness condition stated at the beginning of this section.

Note that it does not make a difference, whether we require $E \setminus \bigcup_{k \in \mathbb{N}_0} F_{k,p}$ to be impossible or to have probability 0.[17]

When we try to consider robustness against denial of service attacks, we see that the adversary may hinder termination of the protocol by just not giving time slices to some party, maybe even taking all time slices for itself. This is clearly allowed by the notion of well-definedness, but in many cases we may need some non-blocking adversary which will not stop any party indefinitely. We again have several variants:

- *Non-blocking:* Each machine is invoked an infinite number of times.
- *Polynomially non-blocking:* Each machine is invoked an infinite number of times and between two of its invocations there passes at most a number of time blocks polynomial in the security parameter, in the ID of the party, and in the number of the invocation.

In the first variant we again have to distinguish between the impossibility of violating our condition and the probability of 0.[18]

In the second one this distinction does not make a difference.[17]

Note that our definitions imply termination of the adversary, since otherwise the machines would not get their infinite number of invocations.

Definition 5.6: (Stochastically) non-blocking adversary
Let R be the set of execution transcripts $\chi = (x_i)$, which satisfy that for any machine $P \in I_{\text{adv}}$ there is an infinite number of x_i in χ with $x_i = (\mathit{stop}, \texttt{token:}P)$.

Let E be the set of all execution transcripts.

We call an adversary $\mathcal{A} \in \text{IAQS}$ *non-blocking*, if $E \setminus R$ is impossible on any quantum interaction.

An adversary $\mathcal{A} \in \text{IAQS}$ is *stochastically non-blocking*, if, for any quantum interaction ι, the probability $P(\mathrm{X} \in E \setminus R)$ vanishes, where X is the execution transcript on ι. □

Here R is the set of the transcripts, in which every party is activated an infinite number of times, or equivalently, in which no party is ever activated for the last time. So this definition requires, that it is impossible or has probability 0, that a party is sometime activated for the last time.

A separating example for those two definitional variants would be an adversary, which on every activation i selects some machine M_i with $P(M_i = j) \propto e^{-j}$. It is

$$P(\text{some } j \text{ appears only a finite number of times in } (P_i)) = 0,$$

but clearly any sequence of activations is possible.

Definition 5.7: Polynomially non-blocking adversary
Let R be as in the previous definition, and let $R_{P,n,m,k}$ ($P \in I_{\text{adv}}$, $n \in \mathbb{N}$, $k, m \in \mathbb{N}_0$) be the set of all execution transcripts $\chi = (x_i)$, which satisfy, that there is an n-th element of χ having the form $(\mathit{stop}, \text{Ж}(\cdot, \ldots, \cdot, \text{Ж}(\texttt{token}, P), \cdot, \ldots, \cdot))$, and in front of it there are at most m elements having the form $(\mathit{stop}, \text{Ж}(\cdot, \ldots, \cdot, \texttt{timeblock}, \cdot, \ldots, \cdot))$, and which additionally satisfy one of:

[17] We omit the proof for this statement, but it uses ideas similar to the reasoning presented in section A.3.

[18] Image an adversary randomly choosing the next recipient of the token out of a linearly growing set of machines. Then it is of probability 0, though not impossible, that some party gets the token only a finite number of times.

- for the first element of χ having the form (rec, f) the word f has length k,
- there is no such element and $k = 0$.

Let E be the set of all execution transcripts and $i : I_{\text{adv}} \to \mathbb{N}_0$ any mapping.

We call an adversary $\mathcal{A} \in$ IAQS *polynomially non-blocking with respect to* i if there is a polynomial p, such that on any quantum interaction the following set is impossible:

$$E \setminus \Big(R \cap \bigcup_{k \in \mathbb{N}_0} \bigcap_{P \in I_{\text{adv}}} \bigcap_{n \in \mathbb{N}} R_{P,n,p(i(P)+n+k),k}\Big) \tag{19}$$

where we consider sets $R_{P,n,m,k}$ with negative m as empty. □

Here $R_{P,n,m,k}$ is the set of all transcripts with security parameter k, in which the machine P was activated for the n-th time in the m-th time block or earlier. So (19) denotes the set of transcripts, which are not in R (i.e. do not even satisfy the non-blocking property define previously) or for which have a security parameter k, such that there is a party P and an invocation number n, such that the n-th invocation of P does not happen before time block $p(i(P) + n + k)$, i.e. within polynomial time.

We have still to clarify the meaning of the function i. As seen above, it is used to assign each party a number, so that the expression "polynomial in the ID of the party" gets meaningful. But what a function shall be used here? This in fact depends strongly of the situation. But the following points should be considered:

- A bounded function i should not be taken, since then an infinite number of parties must be invoked within a finite number of time blocks (namely $p(\max_P i(P) + 1 + k)$), which is obviously impossible.
- Also many unbounded functions i are infeasible for similar reasons, at least for a well-formed adversary.
- If there is only a finite number of non-dummy machines, the following construct can be used for i: Let $\tilde{\imath} : I_{\text{adv}} \to \mathbb{N}_0$ be an arbitrary injective function. Then set $i(P) := 0$ for non-dummy machines and $i(P) := \tilde{\imath}(P)$ for dummy machines.

  This is feasible even for a well-formed adversary[19] and fair with regard to the non-dummy machines.
- Pay attention not to use simply any arbitrary enumeration of I_{adv}. If you have e.g. non-dummy machines with IDs $\mathtt{P{:}1}^n$ an would enumerate them by taking i to yield the number of the party interpreted as a binary string, and not as a unary one, then i would grow exponentially in n, though n would intuitively be considered the number of the party.

  Try to find out what intuitively is the enumeration of the parties, and choose i accordingly.
- If the number of machines partaking the protocol in always finite, but grows with k, then try to choose the i such, that there is a polynomial q such that $i(P) < q(k)$ for each k and all machines partaking on security parameter k, and additionally for all n the cardinality of $i^{-1}(\{n\})$ should be bounded polynomially in n. This gives a sensible and feasible definition of polynomially non-blocking adversaries for this case.

5.4 Protocols and adversarial communication frameworks

In order to facilitate the definition of security in chapter 6, we now introduce the notion of the adversarial communication framework. It is an AQN without adversary, environment, parties, or corruption structure. So it basically is the underlying communication architecture in which we want to execute our protocol.

Definition 5.8: Adversarial communication framework (ACF)
An *adversarial communication framework (ACF)* $\mathcal{C}$ is a sequence of functionalities $(\mathcal{C}_i)_{i \in I_F}$ ($\mathcal{C}_i \in$ IAQS), and an access structure $f_{\mathcal{C}} : I_F \times I_P \cup I_P \times I_F \to \{0, 1\}$. □

The definition of a protocol is fairly simple:

Definition 5.9: Protocol
A protocol π is a mapping $\pi : I_P \to \text{IAQS}, i \mapsto \pi_i$ and an *associated ACF* $\mathcal{C}_\pi$. □

We can now combine those two definitions:

Definition 5.10: Instantiation of an ACF
Let $\mathcal{C}$ be an ACF. Let further be $\mathcal{E}, \mathcal{A} \in$ IAQS (the environment and the adversary), $\mathfrak{A} \subseteq 2^{I_P}$ (the corruption

[19] Activate e.g. in time block n the parties P with $i(P) < n$.

structure) and π a protocol. Then the *adaptive resp. static instantiation of* $\mathcal{C}$ *with environment* $\mathcal{E}$*, adversary* $\mathcal{A}$*, corruption structure* $\mathfrak{A}$ *and the protocol* π is denoted by $\mathcal{C}(adaptive, \mathcal{E}, \mathcal{A}, \mathfrak{A}, \pi)$ resp. $\mathcal{C}(static, \mathcal{E}, \mathcal{A}, \mathfrak{A}, \pi)$, and defined to be the adaptive resp. static AQN $\mathcal{N}$ constructed as follows:

The access structure of is taken from $\mathcal{C}$ ($f_{\mathcal{N}} := f_{\mathcal{C}}$), the specified environment, adversary and corruption structure are used as specified ($\mathcal{N}_E^{(k)} := \mathcal{E}$, $\mathcal{N}_A^{(k)} := \mathcal{A}$, $\mathfrak{A}_{\mathcal{N}} := \mathfrak{A}$), the functionalities are taken from the ACF ($\mathcal{N}_i^{(k)} := \mathcal{C}_i$ for $i \in I_F$) and finally the parties from the protocol ($\mathcal{N}_i^{(k)} := \pi_i$ for $i \in I_P$). □

5.5 Output of an AQN

We now have a notion of adversarial quantum networks, which contain an environment. But for our definition of security we need to define, what the output of the environment is, i.e. the overall outcome of our experiment comparing the two protocols. Therefore we will define the output of an AQN as follows:

Let $\mathcal{N}$ be an AQN. The *output* $\mathrm{Out}_k(\mathcal{N})$ *of* $\mathcal{N}$ *with security parameter* k is a probability distribution over $\{0, 1, \perp\}$, given by the following probabilistic algorithm:

1. Set $|\Psi_{\mathcal{N}^{(k)}}\rangle \in \mathcal{H}_{\mathcal{N}}$ to $|\Psi_{0,\mathcal{N}^{(k)}}\rangle$, i.e. to the start configuration of $\mathcal{N}^{(k)}$.
2. Apply $U_{\mathcal{N}^{(k)}}$ to $|\Psi_{\mathcal{N}^{(k)}}\rangle$, i.e. let the network evolve one time step.
3. Measure the content of the global history tape, i.e. measure $\mathcal{H}_{G,\mathcal{N}^{(k)}}$. Let the outcome of that measurement be G.
4. Search on G for the first frame output by $\mathcal{E}$ which is of the form `return0` or `return1`. Call it r.
5. Go to step 2, if no such frame was found.
6. If $r = \texttt{return0}$, terminate and output 0.
7. If $r = \texttt{return1}$, terminate and output 1.

If the algorithm does not terminate, let its output be $\perp$.

Additionally we need the notion of the *output* $\mathrm{Out}_{k,n}(\mathcal{N})$ *of* $\mathcal{N}$ *with security parameter* k *after* n *time blocks.* It is defined by a similar algorithm, in fact we only need to change step 4 into

4. Search on G for the first frame output by $\mathcal{E}$ which is of the form `return0`, `return1` or for the first output of the scheduling function containing $Ж(\texttt{timeblock}, u(n))$, whichever comes first. Call it r.

and append a last step

8. If $r = Ж(\texttt{timeblock}, u(n))$, terminate and output $\perp$.

Formally, these two concepts are defined as follows:

Definition 5.11: Output of an AQN

Let $\mathcal{N}$ be an AQN. The *output* $\mathrm{Out}_k(\mathcal{N})$ *of* $\mathcal{N}$ *with security parameter* k is a probability distribution over $\{0, 1, \perp\}$, defined as follows:

Let for some $G \in \Sigma^*$ be $Ж(g_1, \dots, g_m) := G$. If this is not possible, let $m := 0$. Let $r_0(G)$ be minimal with $g_{r_0(G)} = Ж(\texttt{frame}, E, \texttt{return0})$ or, if no such index exists, $r_0(G) := \infty$. Let also $r_1(G)$ be minimal with $g_{r_1(G)} = Ж(\texttt{frame}, E, \texttt{return1})$ or $r_1(G) := \infty$.

Let P_0 be the projection onto

$$\mathrm{span}\{G \in \Sigma^* : r_0(G) < \infty, r_0(G) < r_1(G)\}$$

and P_1 the projection onto

$$\mathrm{span}\{G \in \Sigma^* : r_1(G) < \infty, r_1(G) < r_0(G)\}$$

Then

$$P(\mathrm{Out}_k(\mathcal{N}) = i) := \lim_{T\to\infty} \|P_i U_{\mathcal{N}^{(k)}}^T |\Psi_{0,\mathcal{N}^{(k)}}\rangle\|^2 \qquad (i = 1, 2)$$

and

$$P(\mathrm{Out}_k(\mathcal{N}) = \perp) := P(\mathrm{Out}_k(\mathcal{N}) \notin \{0, 1\}).$$

□

In this definition, $r_0(G)$ denotes the index on the global history tape, where the environment has output `return0`. Since to that tape data is only appended, $r_0(G)$ indicates the time, when the environment output this frame. The same holds for $r_1(G)$ and `return1`.

Therefore P_0 projects onto all configurations, in which there was an output `return0`, and in which it occurred before a potential output `return1`. The analogue is valid for P_1.

So finally

$$\operatorname{tr} \ddot{P}_i(\ddot{M}_G \ddot{U}_{\mathcal{N}^{(k)}})^T \varrho(|\Psi_{0,\mathcal{N}^{(k)}}\rangle)$$

is the probability, that the output of the algorithm is i after T steps, where M_G is the measurement of the global history tape and $\ddot{X}$ is the density matrix operator for X.

Since $U := U_{\mathcal{N}^{(k)}}$ operates injectively on $|G\rangle$, i.e. for $U|\Psi\rangle \otimes |G\rangle = \sum_i \alpha_i |\Psi_i\rangle |G_i\rangle$ the value of G is known, when one of the G_i is known, it is $\ddot{M}_G \ddot{U} \ddot{M}_G = \ddot{M}_G \ddot{U}$.[20]

So we can simplify

$$\operatorname{tr} \ddot{P}_i(\ddot{M}_G \ddot{U}_{\mathcal{N}^{(k)}})^T \varrho(|\Psi_{0,\mathcal{N}}\rangle) = \operatorname{tr} \ddot{M}_G \ddot{P}_i \ddot{U}^T_{\mathcal{N}^{(k)}} \varrho(|\Psi_{0,\mathcal{N}}\rangle) = \operatorname{tr} \ddot{P}_i \ddot{U}^T_{\mathcal{N}^{(k)}} \varrho(|\Psi_{0,\mathcal{N}}\rangle) = \| P_i U^T_{\mathcal{N}^{(k)}} |\Psi_{0,\mathcal{N}}\rangle \|^2,$$

where we use additionally, that the $\ddot{P}_i$ and $\ddot{M}_G$ commute, and that $\ddot{M}_G$ is trace-preserving.

So $P(\mathrm{Out}_k(\mathcal{N}) = i)$ is the probability, that the algorithm yields i at all. (Note that the argument of the limes is monotone in n, so we can use the limes here.)

Definition 5.12: Output of an AQN after n time blocks
Let $\mathcal{N}$ be an AQN. The *output* $\mathrm{Out}_{k,n}(\mathcal{N})$ *of* $\mathcal{N}$ *with security parameter* k is a probability distribution over $\{0, 1, \perp\}$, defined as follows:

For any $G \in \Sigma^*$, let $g_1, \ldots, g_m$, $r_0(G)$ and $r_1(G)$ be as in the preceding definition. Let further $t_n(G)$ be minimal such that $g_{t_n(G)}$ has the form Ж(`sched`$, \cdot, \cdot,$ Ж(`timeblock`$, n)$. Let $t_n(G) := \infty$, if there is no such index.

Let $P_{0,n}$ be the projection onto

$$\operatorname{span}\{G \in \Sigma^* : r_0(G) < \infty, r_0(G) < r_1(G), r_0(G) < t_n(G)\}$$

and $P_{1,n}$ the projection onto

$$\operatorname{span}\{G \in \Sigma^* : r_1(G) < \infty, r_1(G) < r_0(G), r_1(G) < t_n(G)\}$$

Then

$$P(\mathrm{Out}_{k,n}(\mathcal{N}) = i) := \lim_{T \to \infty} \| P_{i,n} U^T_{\mathcal{N}^{(k)}} |\Psi_{0,\mathcal{N}^{(k)}}\rangle \|^2 \qquad (i = 1, 2)$$

and

$$P(\mathrm{Out}_{k,n}(\mathcal{N}) = \perp) := P(\mathrm{Out}_{k,n}(\mathcal{N}) \notin \{0, 1\}).$$

□

Here $t_n(G)$ is the moment, in which the n-th time block is entered, so that $P_{i,n}$ are projections similar to P_i, with the additional constraint, that the concerning frame must have been output before time block n, otherwise it is ignored.

Therefore we get $P(\mathrm{Out}_{k,n}(\mathcal{N}) = i)$ to be the probability of an output i in the above algorithm.

Note that $\mathrm{Out}_k(\mathcal{N}) = \lim_{n \to \infty} \mathrm{Out}_{k,n}(\mathcal{N})$. We omit the formal proof for this.

[20] This is due to the fact, that the result of the first M_G-measurement is completely determined by the second one, so the first one can be omitted. A formal proof of this statement is given in section A.5.

6 Basic Security Model

In this chapter we will introduce the formal definition of security. Due to the work in previous chapters, the definitions are comparatively simple.

We will distinguish several notions of security, differentiated by several parameters:

First we distinguish between adaptive and static security. In the adaptive case, the adversary is free to corrupt parties at any time, while in the static case, it may only corrupt before any machine except the adversary and the environment have been activated.

Secondly we may specify, when two output distribution ensembles are to be considered indistinguishable. Any relation can be specified, but two special cases are of greater importance and have proper names: In the case of absolute security, we demand complete equality between the two ensembles, while in the case of approximative security, the stochastic indistinguishability as defined in definition 2.8 is assumed.

The third point we may specify is, which parties may be corrupted. This is specified by a set of sets $\mathfrak{A}$. A party may be corrupted, if and only if the resulting set of corrupted parties is an element of $\mathfrak{A}$.

Further we may restrict the sets of environments and adversaries, over which is quantified.

And lastly may specify in which ACF the protocols to be compared are to be executed. If unspecified, their associated ACF are assumed.

Definition 6.1: Security
We call a protocol π *adaptively resp. statically as secure as* a protocol ϱ *with respect to an indistinguishability relation* $\approx$*, an environment set* $\mathcal{C}_\mathcal{E}$*, an adversary set* $\mathcal{C}_\mathcal{A}$*, a corruption set* $\mathfrak{A} \subseteq 2^{I_P}$*, and communication frameworks* $\mathcal{C}_\pi$*,* $\mathcal{C}_\varrho$ (written $\pi \geq_{\approx,x,\mathcal{C}_\mathcal{E},\mathcal{C}_\mathcal{A},\mathfrak{A},\mathcal{C}_\pi,\mathcal{C}_\varrho} \varrho$), where x is *adaptive* resp. *static*, if for each well-formed adversary $\mathcal{A} \in \mathcal{C}_\mathcal{A}$ there is a well-formed adversary $\mathcal{S} \in \mathcal{C}_\mathcal{A}$ with polynomial running time relative to $\mathcal{A}$, so that for each environment $\mathcal{E} \in \mathcal{C}_\mathcal{E}$ the following equation holds:

$$(\mathrm{Out}_k(\mathcal{C}_\pi(x, \mathcal{E}, \mathcal{A}, \mathfrak{A}, \pi)))_k \approx (\mathrm{Out}_k(\mathcal{C}_\varrho(x, \mathcal{E}, \mathcal{S}, \mathfrak{A}, \varrho)))_k$$

If x is not specified, we assume $x =$ *adaptive.*

If $\mathcal{C}_\mathcal{E}$ or $\mathcal{C}_\mathcal{A}$ are not specified or given by the context, we assume the set of all IAQS.

If $\mathcal{C}_\pi$ or $\mathcal{C}_\varrho$ are not given we assume the communication framework associated to π resp. ϱ.

If $\approx$ is not given, we assume the stochastic indistinguishability defined in definition 2.8 and talk about *approximative security* (π *is approximatively as secure as* ϱ).

If $\approx$ is equality, we talk about *absolute security* (π *is absolutely as secure as* ϱ). □

The foregoing definition is just what we have developed in section 1.6. Noticeable is only the order of quantifiers. We require, that the ideal adversary $\mathcal{S}$ depends only of the real adversary $\mathcal{A}$, but is independent of the environment $\mathcal{E}$. This is the stricter variant (compared with an ideal adversary depending on the environment), and this form is needed for our proof of the composition theorem, though we do not know whether it would still hold using the weaker variant.

Definition 6.2: Security under polynomial running time
We say π *is as secure as* ϱ *under polynomial running time*[21] if for each well-formed adversary $\mathcal{A} \in \mathcal{C}_\mathcal{A}$ there is a well-formed adversary $\mathcal{S} \in \mathcal{C}_\mathcal{A}$, so that for sufficiently large polynomials p the following equation holds for each environment $\mathcal{E} \in \mathcal{C}_\mathcal{E}$:

$$(\mathrm{Out}_{k,p(k)}(\mathcal{C}_\pi(x, \mathcal{E}, \mathcal{A}, \mathfrak{A}, \pi)))_k \approx (\mathrm{Out}_{k,p(k)}(\mathcal{C}_\varrho(x, \mathcal{E}, \mathcal{S}, \mathfrak{A}, \varrho)))_k$$

Notational conventions are as in the above definition of security. □

Here we restrict the running time (more exact: the number of time blocks, which is the time as seen by the environment) to be bounded by a polynomial in the security parameter. Obviously we should not restrict the definition to some particular polynomial, so we want the indistinguishability for any sufficiently large polynomial. The quantifier for the polynomial has been inserted thus, that the definition is strictest possible.

Note that this definition is more an auxiliary one, more useful is the following one:

Definition 6.3: Polynomial security
We say π *is polynomially as secure as* ϱ[21] ($\pi \geq_{P,\approx,x,\mathcal{C}_\mathcal{E},\mathcal{C}_\mathcal{A},\mathfrak{A},\mathcal{C}_\pi,\mathcal{C}_\varrho} \varrho$ or simply $\pi \geq_P \varrho$) if π is as secure as ϱ under polynomial running time with respect to environment set $\mathcal{C}_\mathcal{E} \cap \mathrm{PITM}$ and adversary set $\mathcal{C}_\mathcal{A} \cap \mathrm{PITM}$.

Notational conventions are as in the above definitions of security. □

[21] with respect to the same things as in the definition of security above.

In the definition of security under polynomial running time we had no restriction on the nature of environment or adversary. In the definition of polynomial security, however, we restrict these to be PITM, since our intuitive notion of polynomial running time is, that no party can evaluate more than a polynomial number of time steps. By restricting the machines to polynomial ones, we comply with this intuitive point of view.

Sometimes we cannot only say, that for any adversary and environment the output distributions are indistinguishable, but we can even give a bound to the statistical distance independent of the given adversary or environment, giving us the possibility to state explicitly what is the maximum probability of a successful attack to the protocol *for a given security parameter*. This stronger notion of security is captured by the definition of security with bounded risk:

Definition 6.4: Security with bounded risk
We call π *as secure as ϱ with bounded risk* ($\pi \geq_{*,\approx,x,\mathcal{C}_\mathcal{E},\mathcal{C}_\mathcal{A},\mathfrak{A},\mathcal{C}_\pi,\mathcal{C}_\varrho} \varrho$ or simply $\pi \geq_* \varrho$) if there is a negligible bound $\delta : \mathbb{N} \to \mathbb{R}_{\geq 0}$ (as defined in definition 2.5), such that $\pi \geq_\approx \varrho$, where $\approx$ is the stochastic indistinguishability with respect to $\{\varepsilon : \varepsilon = \delta \text{ almost everywhere}\}$ (as defined in definition 2.7).

Notational conventions are as in the above definitions of security. □

7 Extensions

In this chapter we will present some extensions or variants to the security model presented previously. These extensions will mostly be given in a less formal way or sometimes even only be hinted.

7.1 Multi-phase protocols

The first extension we present is that of multi-phase protocols. Here we extend our framework (strictly speaking just the adversarial scheduling function) to have some additional variable, the current phase of the protocol. This phase is then known to all machines, and can be used for the following purposes:

- The protocol specification may depend on the phases, you may e.g. design a protocol, where on the first day there is some communication using some secrets, and on the second day all secrets are disclosed.
- You may design further extensions to the security model, where some properties (like available primitives) depend on the current phase.

The existence of phases every machine knows is justified by the fact, that in real life we can usually assume, that all machine have a reliable consensus, which e.g. is the current date, so that a protocol, which runs every day at noon within some seconds will be clearly separated into phases by the day breaks.

A new phase is initiated by the environment, since we assume those phases to be externally given and not modifiable by the adversary (e.g. the current date).

As the set of all phases we take $\mathcal{PH} := \{pre, post\} \cup \mathbb{Q}$. Here *pre* denotes the pre-execution phase, *post* the post-execution phase, and $q \in \mathbb{Q}$ is some phase during the protocol execution. We chose $\mathbb{Q}$ for the protocol phases, so that before, after and between two phases there may be another phase, giving much flexibility. In cases where only a few phases are needed, we can simply partition $\mathbb{Q}$ into intervals and consider all phases inside one interval as the same phase.

The phases are half-ordered, no phase φ_2 may be activated after some phase φ_1, if $\varphi_2 < \varphi_1$. We take here the natural total-ordering on $\mathbb{Q}$, extended by $pre < \mathbb{Q} < post$.

In the *pre* and the *post* phase, all communication to and from the parties is disabled. Only interaction between environment, adversary and functionalities, and corruption of parties is possible in these phases.

The following changes are to be made to the scheduling function:

- Let φ be the current phase.
- At the beginning, $\varphi = pre$.
- If the environment outputs a frame of the form $\text{Ж}(\texttt{phase}, \psi)$, where ψ is some phase,[22] and where $\varphi \leq \psi$, set $\varphi := \psi$.
- If a machine is to be activated, which was activated the last time with another value of φ, then send to that machine the current phase (using a control event with new frame $\text{Ж}(\texttt{phase}, \varphi)$).
- If a message delivery from or to a party is requested, but $\varphi \in \{pre, post\}$, do not deliver that message.

The foregoing definition of multi-phase protocols contains one problem: The current phase might change before the parties have done all calculation intended for that phase. This is contrary to the intuitive notion of phases, since in real life it would not happen, that e.g. some day is suddenly and unexpectedly rather short. This fact could possibly introduce unwanted means of distinguishing real and ideal protocol. Fortunately, this does only make the definition of security stricter. Of course, this problem is only of importance, if we consider non-blocking adversaries, in the case of blocking adversaries, the parties could be blocked until the end of the phase anyway, so premature phase changing does not make any difference.

7.2 Temporary assumptions

Often the security of protocols is proven with respect to some assumptions, e.g. available primitives, computational power, or even known algorithms. In some cases however, it may be reasonable to assume the non-availability of some adversarial resources for the present, though would be is questionable, whether the same resources might not possibly be available in the future. To model this case, we would like to be able to

[22] There must of course be some encoding of phases, e.g. *pre* and *post* are $\texttt{pre}$ resp. $\texttt{post}$, and $\frac{r}{s}$ with $\mathrm{ggT}(r,s) = 1$ and $s > 0$ is $\text{Ж}(u(r), u(s))$ with $u(x) = \texttt{1}^n$ for $n \geq 0$ and $u(x) = \texttt{01}^{-n}$ for $n < 0$.

formalise statements like "the protocol is secure, assuming that some resource X is not available at moment, but assuming further, that it might be available some day after the protocol's termination."

An example would be a broadcast channel implemented using RSA. We could prove it to be secure under computational assumptions, if attacking RSA is not efficiently feasible. But we can further prove, that if some day *after* the protocol execution there will be e.g. the possibility to factor large numbers, then the broadcast would still be secure, since all inputs of the protocol have been made public anyway.

All extensions described below assume multi-phase protocols as described in section 7.1, of course.

7.2.1 Available primitives

The first kind of temporary assumption is that of available primitives. We do not implement this by letting the functionalities be present or absent according to the phase, but instead by allowing or restricting access to them.

To do so, we use

$$f_{\mathcal{N}} : (I'_P \times I_F \cup I_F \times I'_P) \times \mathcal{PH} \longrightarrow \{0,1\}$$

as the access function of the QN $\mathcal{N}$. Here $I'_P := I_P \cup \{A\}$.

Then in the scheduling function, the current phase is given as the second argument to $f_{\mathcal{N}}$, so that accessibility of functionalities is dependent of the current phase.

Further the messages from or to the adversary are subject to the access function, in order that we can hide the functionality even from the adversary.[23]

7.2.2 Corruption

A resource of the adversary, which might be subject to temporal changes, is the possibility to corrupt parties. Here manifold variants are conceivable. Examples include:

- There could be a set of sets of corruptible parties for each phase, and the corruptions in each phase be independent of those in previous phases.
- Each phase could have a set of sets which is a superset of that in the previous phase, and a party could then be corrupted if after corruption the set of corrupted parties lies in that set of sets.
- Or a party could only be corruptible in some later phase, if some other "twin party" had been corrupted in some foregoing phase.

To be able to cope with that variety of possibilities, we choose a rather general approach: Instead of a set of sets $\mathfrak{A}$, we have a function

$$\mathfrak{A} : (\mathcal{PH} \cup \{\perp\})^{I_P} \times \mathcal{PH} \longrightarrow \{0,1\}.$$

It is then determined as follows, whether a party P can be corrupted in phase φ: Let $t(P') := \varphi'$ if the party P' has been corrupted in phase φ', or $t(P') := \perp$ if P' is still uncorrupted. Then P can be corrupted, if and only if $\mathfrak{A}(t,\varphi) = 1$ holds.[24]

7.2.3 Computational power

Computational power, too, may be subject to temporary assumptions. Most important is the case, that we restrict machines (especially adversary and environment) to polynomial time within some phases, but let them run unrestricted (or at least not polynomially restricted) in others.

A first approach would be to introduce some computationally unbounded functionality, which can be used to outsource computation. This functionality might then be subject to the temporary access restrictions described in section 7.2.1. Two problems arise with this approach:

- To give the environment superior computational power, we would have to give it access to the functionalities, which would necessitate a major change of our model.
- We have restricted the ideal adversary to be of relative polynomial time in the running time of the real adversary. It is hard to formalise this, if the computation may be outsourced.

[23] When the adversary fakes communication of a corrupted party, the access to functionalities is determined as for the simulated party, of course, i.e. the argument to $f_{\mathcal{N}}$ is the ID of that party, not A.

[24] And, of course, the token is not requested to be transferred to P.

Therefore we favour another approach: We extend the different definitions of running time introduced in section 3.3:

The definition of running time $T^{\mathrm{X}}_{k,n}$ we change to a *phase dependent running time* $T^{\mathrm{X}}_{k,n,\varphi}$, which denotes the average running time on security parameter k in invocation n and phase φ. This is done by replacing $E_{k,n,t}$ by $E_{k,n,t,\varphi}$, which contains $E_{k,n,t}$ and all transcripts, in which the n-th invocation does not lie in phase φ. Then $P(T^{\mathrm{X}}_{k,n,\varphi} = t)$ is defined by using $E_{k,n,t,\varphi}$ instead of $E_{k,n,t}$.

The definitions (reliable) polynomial time we replace by *(reliable) polynomial time in phase* φ. This is done by simply replacing $T^{\mathrm{X}}_{k,n}$ by $T^{\mathrm{X}}_{k,n,\varphi}$.

We do not discuss the extensions of the definitions of termination, since it does not seem necessary to restrict termination to certain phases. Those extensions would, however, run in the same lines as those of the polynomial running time.

The hard and stochastic relative polynomial running time is extended to a *hard resp. stochastic phase dependent relative polynomial running time.* Here we do now require, that the running time of the $\mathcal{M}_2$ is polynomial in $\mathcal{M}_1$ for each phase separately. This is done by replacing $T^{\mathrm{X}_{1/2}}_{k,n}$ by $T^{\mathrm{X}_{1/2}}_{k,n,\varphi}$, and by quantifying not only over all n, k, and t, but also over all phases $\varphi \in \mathcal{PH}$.

Having those new definitions, we change the definition of security to require phase dependent relative polynomial running time between the adversaries, and we do not restrict the sets of adversaries and environments simply to polynomial or terminating machines, but to machines, which are e.g. terminating, but only in some phases polynomial.

7.2.4 Known algorithms

Often we assume, that for some problem there is no known (efficient) algorithm. For example we might say, that no one knows how to efficiently factor large numbers. While this is a realistic assumption, it may well be that in the future some algorithm is discovered.

Since we cannot easily model knowledge, an obvious way to model ignorance of an algorithm would be to assume non-existence of that algorithm. So we say: "If there is no algorithm for factoring, then the protocol is secure." This may be sufficient for some cases, but there are two major problems:

- If there is an unknown algorithm for factoring, the proof of security is wrong, and it may theoretically be, that there is an attack against the protocol, which does not need knowledge of the factoring algorithm. In praxis it is reasonable to suppose, that if there is an attack, the algorithm for factoring can be found in the proof of security or in the description of the attack. For formally correct reasoning, however, this argument is not vindicable.

- When we want to use temporary assumptions, we would have to say something like "if there is no algorithm for factoring now and if there will be an algorithms for factoring after the protocols execution, then the protocol is secure". Since the premise of this implication is formally wrong, we would trivially be able to prove security and, indeed, anything else, too.

The second problem could be fixed by assuming the non-existence of the algorithm, and then adding a primitive for factoring, which is only available in certain phases, but this solution would be somewhat dirty, and it would not help against the first problem.

Therefore we will pursue an approach, which tries to get hold of the idea of knowledge and ignorance of algorithms. The underlying definition runs approximately in the following terms:

Definition 7.1: Non-usage of an algorithm (informal version)
We say, an algorithm (i.e. an IQTM) $\mathcal{A}$ does *not use an algorithm for a problem* $\mathcal{P}$ *with respect to* Ω, where Ω is an efficiently computable function mapping algorithms to algorithms, if and only if $\Omega(\mathcal{A})$ is not an algorithm for $\mathcal{P}$. □

This definition is used as follows: When we prove, that a protocol π is secure when restricting the class of adversaries and environments to machines, which do not use an algorithm for $\mathcal{P}$ with respect to Ω, where Ω *must* be explicitly given, then we can reason as follows:

We assume, that there is no known algorithm for $\mathcal{P}$. However Ω is known and—if π is successfully attacked—an adversary or an environment $\mathcal{A}$ exists, which does use an algorithm for $\mathcal{P}$, so $\Omega(\mathcal{A})$ is an algorithm for $\mathcal{P}$, but $\Omega(\mathcal{A})$ is also known,[25] so we have a contradiction. Therefore π cannot be successfully attacked.

[25]Since $\mathcal{A}$ and Ω are known. We have, of course, to assume that the person designing the adversary knows Ω, so this reasoning is only valid for publicly published Ω.

We now come to the formal version of the above definition:

Definition 7.2: Problem
A *problem* P is a set of ITM, and we say an ITM $\mathcal{M}$ *solves* $\mathcal{P}$, if $\mathcal{M} \in \mathcal{P}$. □

Here we simply define a problem as the set of all ITM which solve it. So a reasonable definition of the problem of factoring would be the set of all ITM, which on input of a number emit its factorisation with bounded error probability in a time polynomial in the length of the number. Other definitions of a problem, which use a description of the problem rather than enumerating its solutions, might be used, too, since problems defined by specification can be considered as problems in our sense in a natural way. (It is not necessary to explicitly name the elements of a problem.)

Definition 7.3: Non-usage of an algorithm
We say, a partial ITM $\mathcal{A}$ does *not use an algorithm for a problem* $\mathcal{P}$ *with respect to* Ω, where $\Omega : \Sigma^* \to \Sigma^*$ is an efficiently computable function, if and only if $s^{-1}(\Omega(s(\mathcal{A})))$ is not defined, or is not an ITM, or does not solve $\mathcal{P}$. Here $s(\mathcal{M})$ denotes the description of $\mathcal{M} \in \mathrm{ITM}$. □

The description $s : \mathrm{ITM} \to \Sigma^*$ is an injective function converting a partial ITM $\mathcal{M}$ into a description of $\mathcal{M}$, which can be efficiently used. We will not give an explicit definition here, this can however be done in a straightforward way, it should only be remembered, that elements of $\tilde{\mathbb{R}}$ can be specified by giving an algorithm calculating its digits.

Note that we have specified Ω to be an efficiently computable function, not only a function efficiently computable on quantum computers. In the case, that we allow ITM to solve our problem, and if we have a sufficiently natural definition of the problem, this does not make any difference, since Ω could simply simulate a quantum $\tilde{\Omega}$ by just constructing an universal quantum Turing machine which evaluates $\tilde{\Omega}$ and then runs its output.

On the other hand, if we have only allowed classical machines to be solutions to our problem, then we probably wanted to model a world without quantum adversaries, so restricting Ω to be classical, too, makes sense.

Until now, we have only defined what it means when we say, "a machine does not use an algorithm A". In order to formulate temporary assumptions, however, we need to say, "a machine does not use an algorithm A in phases Φ". In order to do this, we have to extract that part of an ITM, which gets used in some phases Φ.

Definition 7.4: Phase trace in an ITM
Let $\mathcal{M} = (Q, q_0, q_f, n, \delta)$ be an ITM, and $\Phi \in \mathcal{PH}$ be a set of phases. Then we construct the *trace* $\mathcal{M}^\Phi$ *of the phases* Φ *in* $\mathcal{M}$ as follows:

Let O_i, P_{r_i}, P_{f_i}, $P^{(i)}_{k_i}$ be the various parts of an quantum interaction ι, and P_q be the projection of $\mathcal{H}_{Q,\mathcal{M}}$ onto all configurations, where the current state is q.[26]

Then let $Q_T \subseteq Q$ be the set of all states $q \in Q$ satisfying, that there is a quantum interaction ι, and an $n \in \mathbb{N}_0$, such that

$$P_q \prod_{i=1}^{n} (O_i P_{r_i} P_{f_i} P^{(i)}_{k_i})(|\Psi_{0,\mathcal{M}}\rangle \otimes |\#\rangle \otimes |\#\rangle \otimes |0\rangle \otimes |0\rangle) \neq 0,$$

and such that in $f_1, \dots, f_n$ there is an f_i having the form $Ж(\mathtt{phase}, \cdot)$, and the last such f_i equals $Ж(\mathtt{phase}, \varphi)$ for some $\varphi \in \Phi$.

Then let $\mathcal{M}^\Phi$ be the partial ITM resulting from $\mathcal{M}$ by restricting it to the states in Q_T. □

In the forgoing definition, we consider all quantum interactions ι, which let $\mathcal{M}$ after the n-th time step be in phase φ for some $\varphi \in \Phi$. Then we measure the state of $\mathcal{M}$ after the n-th time step, and let Q_T contain all states, which can occur in this measurement with non-vanishing probability. Then we can restrict $\mathcal{M}$ to only those states and thus get the trace.

To clarify this definition, we compare the ITM with a normal program. Then we can run the program and mark any code, which gets executed while in phase $\varphi \in \Phi$. When we do this for any possible executions, we get all code which has influence on the behaviour in phases Φ.

We can now finish our definition of non-usage of an algorithm in certain phases:

Definition 7.5: Phase dependent non-usage of an algorithm
We say, an ITM $\mathcal{A}$ does *not use an algorithms for a problem* $\mathcal{P}$ *with respect to* Ω *up to phase* $\varphi \in \mathcal{PH}$, if and only if $\mathcal{A}^\Phi$ with $\Phi := \{\tilde{\varphi} \in \mathcal{PH} : \tilde{\varphi} \leq \varphi\}$ does not use an algorithms for $\mathcal{P}$ with respect to Ω. □

[26] The exact form of P_q depends, of course, of whether $\mathcal{M}$ is an IQTM, an ICTM, or an IQCTM.

In the preceding definition, it might seem useful also to allow some Φ which is not of the structure $\{\tilde{\varphi} \in \mathcal{PH} : \tilde{\varphi} \leq \varphi\}$. This definition, however, would not conform to our intuitive requirement any more, as the following example shall demonstrate: Assume some machine which does not use an algorithm for $\mathcal{P}$ in phases $\Phi := \{\varphi_2\}$. Then an algorithm for $\mathcal{P}$ might be written onto some tape in phase $\varphi_1 < \varphi_2$, and in phase φ_2 simply executed. Then in phase φ_2, the trace contains nothing but some universal Turing machine's program, but nevertheless it may solve the problem $\mathcal{P}$.

7.3 Constraints

Sometimes, we do not want to get security in all cases, but are contented with security under specific conditions. Some examples of such conditional security specifications would be, informally stated:

- "The protocol is secure, as long as the inputs to the parties are all different."
- "The protocol behaves as a coin toss, or all parties output $\perp$."
- "The protocol is secure, as long as the adversary does not send more than a million messages."

The first two of those problems we will handle using *constraints.* Roughly sketched, constraints are measurements, which test continuously, whether a certain condition is satisfied while executing the AQN, and, if that test fails, the given run is aborted and not considered when generating the statistics of the output of the environment. This effects, that the output distribution goes only over those runs, where the constraint is satisfied.

The third problem cannot, unfortunately, be handled using this mechanism. If we use the constraint "the adversary has not send more than a million messages", and try e.g. to implement coin tossing, then the ideal adversary could try to use the following trick to adjust the ideal coin toss to yield any desired value: Assume the adversary gets to know the result of the coin toss before the environment does. Then if the result is not as desired, the adversary sends a million dummy messages, thus aborting the run and preventing it from being counted in the environment's output statistics. So the statistics run only over the cases, where the coin toss outputs the desired result. This makes many protocols secure, which we would consider as insecure. So we cannot use constraints for formulating conditions like the above one.

We do not know any formal specification, which constraints are "harmless", so constraints should only be designed with great care.

Formally, a constraint is defined as follows:

Definition 7.6: Constraint
Let $\mathcal{N}$ be a quantum network. Then a *constraint C on $\mathcal{N}$* is a sequence $(C^{(k)})_{k\in\mathbb{N}}$ of unitary transformations on $\mathcal{H}_\mathcal{N} \otimes \mathbb{C}^\mathbb{Z} \otimes \mathbb{C}^2$, which operates read-only on $\mathcal{H}_\mathcal{N}$. $\square$

We have chosen to define the constraint not directly as a measurement, but as a transformation which operates on the network's configuration space $\mathcal{H}_\mathcal{N}$, on an auxiliary space $\mathbb{C}^\mathbb{Z}$, and on an output space $\mathbb{C}^2$. This output space will be measured and should contain $|1\rangle$ for a satisfied constraint and $|0\rangle$ for an unsatisfied one. Due to the auxiliary space $\mathbb{C}^\mathbb{Z}$, the constraint can hold arbitrarily complex information about previous time steps.

The parameter k which selects a specific unitary transformation is the security parameter.

When some constraints $C := \{C_1, \dots, C_n\}$ are given, the output of an AQN is redefined as follows: The state of the quantum network is a vector in $\mathcal{H}_\mathcal{N} \otimes \mathbb{C}^\mathbb{Z}_1 \otimes \mathbb{C}^2_1 \otimes \mathbb{C}^\mathbb{Z}_2 \otimes \mathbb{C}^2_2 \otimes \cdots \otimes \mathbb{C}^\mathbb{Z}_n \otimes \mathbb{C}^2_n$ instead of simply $\mathcal{H}_\mathcal{N}$, as it was in section 5.5. All operations given in section 5.5 still operate only on $\mathcal{H}_\mathcal{N}$. After the time evolution, but before measuring $|G\rangle$, apply $C_i^{(k)}$ on $\mathcal{H}_\mathcal{N} \otimes \mathbb{C}^\mathbb{Z}_i \otimes \mathbb{C}^2_i$ for each i. Then measure all $\mathbb{C}^2_i$ in the standard basis. If one of them contains $|0\rangle$, terminate with output $\square$. Otherwise continue as in section 5.5.

By this modified algorithm we get an output distribution $\mathrm{Out}_{k,n}(\mathcal{N})$ over $\{0, 1, \perp, \square\}$. We can now define the constrained output $\mathrm{Out}^C_{k,n}(\mathcal{N})$ as the conditional stochastic variable over $\{0, 1, \perp\}$ defined by

$$P(\mathrm{Out}_{k,n}(\mathcal{N}) = t) = P(\mathrm{Out}_{k,n}(\mathcal{N}) = t |\, \mathrm{Out}_{k,n}(\mathcal{N}) \neq \square) \qquad (t \in \{0, 1, \perp\})$$

If $P(\mathrm{Out}_{k,n}(\mathcal{N}) = \square) = 1$, we write $\mathrm{Out}^C_{k,n}(\mathcal{N}) = \perp$.

Then, in the definitions of security, we check for indistinguishability of the constrained output distributions, where we assume two output distributions to be indistinguishable, if one of them is $\perp$.

Using the foregoing redefinition, the constrained output distributions are statistics only over the runs where the constraints are satisfied. If a constraint is never satisfied, we assume indistinguishability, since if e.g. some environment would never satisfy some constraint, then this environment's output should not be counted at all.

Since all C_i are read-only on $\mathcal{H}_{Q,\mathcal{M}}$, and operate on separated spaces otherwise, they commute, so it is justified to specify a set C of constraints and not a sequence.

7.4 Passive corruption

In this section, we introduce the concept of passive corruption. This models the case, where after corruption the adversary knows the state of the corrupted party, but does not get control over it or its messages. A passive corruption may even take place several times on the same party.

When introducing passive corruption, we have to make the following changes to our definitions:

- The definition of the IAQS is to be extended by a *passive corruption operator,* which is analogous to the corruption operator, but is applied in case of passive corruptions.

- The scheduling function is to be modified thus, that the adversary may request an active or a passive corruption. In case of the passive corruption, the corrupted party is not added to the set of corrupted parties, so that it will still get the token and be able to send and receive messages.

- Instead of $\mathfrak{A}$, the set of sets of allowed corruptions, a more sophisticated mechanism is needed to specify, whether a requested corruption may still take place. There could e.g. be restrictions on how often a party may be passively corrupted, etc.

Probably only passive corruption operators which are read-only on the configuration space make sense, since otherwise the corruption would not satisfy an intuitive notion of "passiveness". Further passive corruption should only be allowed on classical machines, since otherwise interference could be destroyed and thus the machine be influenced. However, if the passive corruption operator is restricted to those parts of a machine which can be considered as classical, also a partly quantum machine could be passively corrupted. E.g. the passive corruption operator of an IAQS with classical interfaces could just export the measured data from the interfaces.

7.5 Cheat Detection

In some cases, we do not necessarily need a protocol securely implementing e.g. some primitive, we only need a protocol which does run securely *or* informs all honest parties that it ran insecurely. Lets for example consider encrypted communication. If we know after execution of the protocol whether it ran securely or not, we can use it to implement a secure encrypted communication as follows: First transfer a key.[27] Then encrypt the message to be transfered using that key, if there was no cheating during the transmission of the key. Now all the adversary can achieve, is hindering the communication, eavesdropping however is impossible.[28]

Informally we define π to be secure with cheat-detection, if π is secure under the constraint, that there is at least one uncorrupted party not outputting $\perp_c$, where $\perp_c$ stands for some previously defined message.

To define this more formally, we first need the concept of a security notion.

Definition 7.7: Security notion
A *security notion* σ is one of the security concepts defined in chapters 6 and 7, together with all its parameters except for the real protocol (i.e. e.g. the set of adversaries, the constraints, the ideal protocol, etc.)

We say *a protocol π is σ-secure under constraints C,* if when adding π to σ as the real protocol (which was the last missing parameter), and adding C to the set of constraints given in σ, the security definition given by that construct is fulfilled.

If no constraints are given, we assume the empty set. □

To clarify this formalism, we give an example: Let σ be the polynomial security with $\mathfrak{A} := X$, indistinguishability relation $\approx'$, and constraints Y, and with the ideal protocol ϱ. Then π is σ-secure under constraints C means that π is as secure as ϱ with $\mathfrak{A} := X$, indistinguishability relation $\approx'$, and constraints $Y \cup C$.

Having this, we can write down the definition of cheat-detection:

Definition 7.8: Security with cheat-detection
Let σ be a security notion and π a protocol. Then π is *σ-secure with cheat-detection $\perp_c$*, where $\perp_c \in \Sigma^*$, if π is σ-secure under the constraint, that for some uncorrupted party P it holds, that is has never sent a message $\perp_c$ to the environment. □

[27] Possible as long as the message.

[28] In fact, we have constructed encrypted communication with partial abort (see section 7.6) from encrypted communication with cheat-detection.

7.6 Security with Partial Abort

Another variant of standard security models is security with partial abort. Here we want to model protocol participants, which implement a given functionality or, if this is not possible, retreat from the protocol. For an example, assume the following situation: We have some protocol for some multi-party computation, where individual parties may come to suspect the computation of being compromised. Then they will notify the user of this (by outputting $\bot_a$) and stop inputting our outputting anything.

To achieve this concept of security, we add a new functionality $\mathcal{F}$ to the ideal communication framework, and we modify the ideal protocol thus, that on getting any message from $\mathcal{F}$, a party P will send $\bot_a$ to the environment, and afterwards P will ignore all messages from the environment (and forward them to $\mathcal{F}$), and wherever P's program tells it to send a message to the environment, it will send it to $\mathcal{F}$ instead.

The functionality $\mathcal{F}$ will, when receiving a message from the adversary containing a party's ID P, send to P an empty message. When receiving a message from a party, it is ignored, i.e. simply stored on some tape and never read again.

For a security notion σ, we will then say a protocol π is *σ-secure with partial abort,* if π is σ'-secure, where σ' is obtained from σ by following the above instructions.

We will omit a more formal definition of this notion.

7.7 Pliable Security

A generalisation of security with cheat-detection is the pliable security. Imagine some protocol, which during its run detects, what level of security may be achieved and then uses and outputs this level of security.

An example would be a protocol for encrypted and authenticated communication. This protocol may then e.g. output, that the channel is encrypted and authenticated, or that it is only authenticated, but not encrypted any more.[29]

To specify such a pliable security, we need to specify a set of security notions $\mathfrak{S}$, together with a half ordering $\le$, a phase χ, an injective mapping $f : \mathfrak{S} \to \Sigma^*$, a security model σ_0, and a set of parties $\mathcal{P}$. Then intuitively $(\mathfrak{S}, \le, \chi, f, \sigma_0, \mathcal{P})$-pliable security of a protocol π means, that the protocol π will run with some security $\sigma \in \mathfrak{S}$, and at the latest in phase χ uncorrupted parties in $\mathcal{P}$ may output, which security has been chosen. Furthermore the protocol is in any case σ_0-secure. Instead of outputting a security which is in fact fulfilled, it may also output some security $\tilde{\sigma} \le \sigma$, because $\tau \le \sigma$ shall mean that σ is a security "at least as good as" τ. Since the security notions have no canonical string representation, $f(\sigma)$ denotes which message is to be sent to the environment for security notion σ.

More formally pliable security is defined as follows:

Definition 7.9: Pliable security
Let $(\mathfrak{S}, \le)$ be a half-ordered set of security notions having a supremum for any subset, $\chi \in \mathcal{PH}$ a phase, $f : \mathfrak{S} \to \Sigma^*$ an injective mapping, $\mathcal{P} \subset I_P$ a finite set, and σ_0 a security notion or $\sigma_0 = \bot$.

Then a protocol π has $(\mathfrak{S}, \le, \chi, f, \sigma_0, \mathcal{P})$-pliable security, if the following conditions hold:

- The protocol π is σ_0-secure or $\sigma_0 = \bot$
- For each $\sigma \in \mathfrak{S}$ the protocol π is σ-secure under the constraint

$$\varphi \le \chi \ \vee \ \sigma = \sup_{\substack{P \in \tilde{\mathcal{P}} \\ \sigma_P \neq \bot}} \sigma_P.$$

  Here φ denotes the current phase, $\tilde{\mathcal{P}}$ the set of all uncorrupted parties in $\mathcal{P}$, and σ_P is such, that $f(\sigma_P)$ is the first message send from P to $\mathcal{E}$ during a phase smaller-or-equal χ formed $f(\sigma')$ with $\sigma' \in \mathfrak{S}$; it is $\sigma_P = \bot$, if no such σ_P exists. If the supremum goes over an empty set, we assume its value to be $\bot$ (note $\sigma \neq \bot$ for $\sigma \in \mathfrak{S}$, so in this case the condition is trivially fulfilled).

If $\le$ is not given, we assume $\tau \le \sigma$, if and only if being σ-secure implies being τ-secure. If σ_0 is not given, we assume $\sigma_0 = \bot$.

If $\mathcal{P}$ is not given, but there is some natural notion of a finite set of parties participating in the protocol, then let $\mathcal{P}$ be that set. □

This definition is in fact stricter than the intuitive formulation, since we say here, that the security reached is *exactly* the supremum of the security notions signalled to the environment, instead of, as required in the

[29]This example could, of course, be realised much simpler by creating an ideal functionality implementing both behaviours, but in some cases the differences between the security models might be more subtle, e.g. polynomial and non-polynomial security.

intuitive formulation, some security notion greater-or-equal that supremum. This may make realisation of protocols with pliable security much harder, unless $\leq$ is the equivalence or the natural half-ordering of $\mathfrak{S}$ given by the implications between security notions. We have, however, not found a way to formalise the desired definition.

7.8 Adjustable Security

Sometimes, we do not know exactly, what security is needed when designing a protocol. The security could depend on some inputs of the protocol, and the protocol would have to behave accordingly. Usually, the less strict security notions will be more efficient, so using the stricter ones will only be done in case of need.

An example for such protocols comes from human interaction. In some cases, an election may be open (i.e. the individual votes are known to the adversary), but as soon as one of the voters requests a secret election, a secret election is carried out.[30]

So we again have a set of security notions $\mathfrak{S}$, a half ordering $\leq$, a phase χ, an injective mapping f, and a security notion σ_0. The meanings of $\mathfrak{S}$, $\leq$, f, and σ_0 are as with pliable security, χ however is the phase up to which the parties need to have the input, which security notion is required. The set $\mathcal{P} \subseteq I_P$ (this time not necessarily finite) denotes the parties, which may have influence on the choice of the security notion.

Then $(\mathfrak{S}, \leq, \chi, f, \sigma_0)$-adjustable security means intuitively, that up to phase χ (inclusive) only security σ_0 is guaranteed, and after χ a security σ greater or equal to all security notions the parties $\mathcal{P}$ have got as input up to phase χ. Formally:

Definition 7.10: Adjustable security
Let $(\mathfrak{S}, \leq)$ be a half-ordered set of security notions, $\chi \in \mathcal{PH}$ a phase, $f : \mathfrak{S} \to \Sigma^*$ an injective mapping, $\mathcal{P} \subset I_P$ a finite set, σ_0 a security notion or $\sigma_0 = \perp$, and $\mathcal{P} \subseteq I_P$.

Then a protocol π has $(\mathfrak{S}, \leq, \chi, f, \sigma_0, \mathcal{P})$-adjustable security, if the following conditions hold:

- The protocol π is σ_0-secure or $\sigma_0 = \perp$.
- For each $\sigma \in \mathfrak{S}$ the protocol π is σ-secure under the constraint

$$\varphi \leq \chi \ \vee \ \sigma = \sup_{\substack{P \in \tilde{\mathcal{P}} \\ f(\sigma_P) \neq \perp}} \sigma_P.$$

  Here φ denotes the current phase, $\tilde{\mathcal{P}}$ the set of all uncorrupted parties in $\mathcal{P}$, and σ_P is such, that $f(\sigma_P)$ is the first message send from $\mathcal{E}$ to P during a phase smaller-or-equal χ formed $f(\sigma')$ with $\sigma' \in \mathfrak{S}$; it is $\sigma_P = \perp$, if no such σ_P exists. If the supremum goes over an empty set, we assume its value to be $\perp$ (note $\sigma \neq \perp$ for $\sigma \in \mathfrak{S}$, so in this case the condition is trivially fulfilled).

If $\leq$ is not given, we assume $\tau \leq \sigma$, if and only if being σ-secure implies being τ-secure, and if σ_0 is not given, we assume $\sigma_0 = \perp$.

If $\mathcal{P}$ is not given, but there is some natural notion of a set of parties participating in the protocol, then let $\mathcal{P}$ be that set. □

7.9 Corruption notification

In our model we have assumed so far, that if the adversary corrupts a party, a notification of this corruption is gotten by the environment upon its next activation. There are, however, several variants.

- *Immediate corruption notification.* This is the corruption model used so far, where the environment gets a corruption notification upon its next invocation.
- *Delayed corruption notification.* Here the environment is not immediately notified. Instead, upon entrance of the *post*-phase, the environment gets (when activated the next time) a list of all parties corrupted so far.
- *Covert corruption.* In this case, the environment is not notified at all about corruptions.

[30]See footnote 29 on page 51.

In the first two models we have, that $\mathfrak{A}$-security implies $\mathfrak{A}'$-security with $\mathfrak{A}' \subseteq \mathfrak{A}$. With covert corruptions however, this is not given. Consider the following example: Some two-party-protocol π is not 1-secure[31]. It then is, with covert corruption, 2-secure, because the ideal adversary can simulate any behaviour by corrupting all two parties.[32] So π is then 2- but not 1-secure.

Therefore the security definition should be changed thus, that $\mathfrak{A}$-security is given, if and only if the old security definition holds for any subset of $\mathfrak{A}$.

7.9.1 Adaptive vs. static security

In this section, we will sketch an example, that adaptive and static security are different for all corruption notification models given in the previous section, even for the three party case.

We assume three parties named A_1, A_2, B, and an authenticated broadcast channel. Whenever we say "choice", we mean an uniformly distributed random choice. Further let always be $\mathfrak{A} := \{\{A_1\}, \{A_2\}, \emptyset\}$.

Consider an ideal functionality $\mathcal{F}$ having the following program:

- Choose χ from $\{0, 1\}$.
- Send χ to the adversary.
- Wait for a message a from the adversary.
- If $\chi = 0$, choose r from $\{0, 1\}$ and send r to B.
- If $\chi = 1$, send a to B.

As real protocol, we take parties A_1, A_2, B, which behave as follows: The parties A_i wait for some message on the broadcast channel, and then broadcast some a_i chosen from $\{0, 1\}$.

The party B does the following:

- Choose $\gamma \in \{1, 2\}$.
- Broadcast γ.
- Wait for broadcasted messages $a_{1/2}$ from $A_{1/2}$.
- Send a_γ to the environment.

We first claim, that π is statically as secure as $\mathcal{F}$. Proof sketch:
Let some real $\mathcal{A}$ be given. Then construct the ideal adversary $\mathcal{S}$ as follows:

- Simulate $\mathcal{A}$ and the real protocol's parties.
- Whenever $\mathcal{A}$ corrupts a party, corrupt the same party.
- Pass communication between environment and corrupted parties to $\mathcal{A}$.
- If $\mathcal{F}$ sends χ to $\mathcal{A}$, simulate, that B broadcasts a message γ, given as

$$\gamma = \begin{cases} 2, & \textit{if } \chi = 0,\ A_1 \textit{ corrupted}, \\ 1, & \textit{if } \chi = 0,\ A_2 \textit{ corrupted}, \\ 1, & \textit{if } \chi = 1,\ A_1 \textit{ corrupted}, \\ 2, & \textit{if } \chi = 1,\ A_2 \textit{ corrupted}, \\ 1, & \textit{if no party is corrupted.} \end{cases}$$

- When both simulated parties have send their a_i, send a to $\mathcal{F}$, where a is what the corrupted A_i has sent if there is some, 0 otherwise.

Obviously this behaves as the real protocol, so π is statically secure. ∎

We further claim, that π is adaptively insecure. Proof sketch:
Let the real adversary run as follows:

[31] I.e. $\mathfrak{A}$ consists of all one-element-subsets of I_P.

[32] Cases may be constructed, where this is not given, if non-simulatable functionalities or parties are present, but for usual protocols this argumentation holds.

- Wait for γ from B.
- Corrupt A_γ.
- Let A_γ broadcast 1.

Let the environment simply output, what it gets from B. Then the output of the environment is always 1. However, for ideal adversary the probability of B sending 0 is at least $\frac{1}{4}$, so the same holds for the environment's output. Therefore the protocol is not adaptively secure. ■

8 Relation of classical and quantum security

An important question, which arises when introducing quantum security, is, whether the proofs of security already made for classical protocols in a classical world are all obsolete.

In this chapter, we will address this question. We will, however, not examine classical security with classical machines, but security, where all communication is classical, the machines will however be internally quantum (i.e. they are IAQS with classical interfaces). This is due to the fact, that—assuming that a quantum computer may not be simulated by a classical one with polynomial overhead—a classical adversary or environment is inferior to a quantum one in computational power, thus some protocols may get insecure just because of that additional power.

In the computationally unbounded case the difference between IAQS with classical interfaces and classical IAQS is unimportant, since e.g. an ICTM can simulate an IQCTM with exponential overhead, and a classical IAQS can simulate an IAQS with classical interfaces even without overhead.

The proposition we want to prove goes as follows:

Proposition 8.1: Relation of classical and quantum security
Let π and ϱ be protocols consisting only of classical IAQS. Let their associated ACF $\mathcal{C}_\pi$, $\mathcal{C}_\varrho$ consist only of IAQS with classical interfaces.

Then π is as secure as ϱ, assuming any combination of the following parameters:

- adaptive or static security,
- polynomial or unbounded security,
- approximative or absolute security,
- any corruption set $\mathfrak{A} \subseteq 2^{I_P}$,
- security with bounded risk,
- any indistinguishability relation $\approx$,

if π is also as secure as ϱ with respect to the same parameters, when restricting the set of adversaries and the set of environments to IQCTM. □

The set of security models for this such an implication holds, is probably greater than the set given here, but not every security model stays secure in the quantum case. E.g. if restricting the set of adversaries to IQTM \ IQCTM, we would of course get classical security, since here the set of adversaries would be empty, but quantumly the protocol might nevertheless be insecure. Further research might e.g. give some characterisations of sets of adversaries and environments, for which the proposition still holds.

Now we proceed to the proof sketch of this proposition:

For clarity, we will always write IAQS with classical interfaces with a superscript prime (e.g. $\mathcal{M}'$)

Let us first assume, that the security model comparing π and ϱ is not polynomial and not of bounded risk.

Let then $\mathcal{A}$ be any real life adversary and $\mathcal{E}$ be any environment. Let $\mathcal{N}_{0r}$ be the instantiation of $\mathcal{C}_\pi$ with $\mathcal{E}$, $\mathcal{A}$, π and $\mathfrak{A}$.

Then we can replace $\mathcal{E}$ by some $\mathcal{E}_\mathcal{A}$ and $\mathcal{A}$ by some $\mathcal{A}'$, without changing the output behaviour of the network, when defining $\mathcal{E}_\mathcal{A}$ and $\mathcal{A}'$ as follows:

Let $\mathcal{E}_\mathcal{A}$ be a machine in $\mathcal{C}_\mathcal{E}$ simulating the machines $\mathcal{A}$ and $\mathcal{E}$, where all communication between those two machines is handled internally,[33] the communication between $\mathcal{E}$ and the parties is forwarded by $\mathcal{E}_\mathcal{A}$ and all communication of $\mathcal{A}$ is rerouted via $\mathcal{A}'$, together with all necessary information on destination and possibly sender. Incoming communication, corruptions and time block increases are also rerouted via $\mathcal{A}'$. When $\mathcal{A}$ wants to transfer the token to some machine $X \in I_{\text{adv}}$, then the token is given to that machine by $\mathcal{A}'$. If $\mathcal{E}_\mathcal{A}$ is activated, $\mathcal{A}$ is activated internally, except when $\mathcal{A}$ had just requested an activation of the environment, then $\mathcal{E}$ is activated. If $\mathcal{A}'$ is activated without having instructions whom to activate next, $\mathcal{E}_\mathcal{A}$ is activated.

Let $\mathcal{A}'$ be a polynomial IQCTM, which simply does the rerouting as specified in the previous paragraph. We can choose $\mathcal{A}'$ to be an IQCTM, since every forwarded message is measured anyway by some party. The message containing the state of a corrupted machine may also be measured before forwarding, since all parties are classical, so their state may be measured.

The instantiation of $\mathcal{C}_\pi$ with $\mathcal{E}_\mathcal{A}$ and $\mathcal{A}'$ we call $\mathcal{N}_{1r}$. It is now

$$\mathrm{Out}_k(\mathcal{N}_{0r}) = \mathrm{Out}_k(\mathcal{N}_{1r}), \qquad \mathrm{Out}_{k,n}(\mathcal{N}_{0r}) = \mathrm{Out}_{k,n}(\mathcal{N}_{1r}).$$

[33] Including the faked communication between two corrupted parties, which generates messages from $\mathcal{A}$ to $\mathcal{A}$.

Since in the resulting QN all machines except $\mathcal{E}_{\mathcal{A}}$ are IQCTM, all incoming and outgoing communication of $\mathcal{A}$ may be measured. So by replacing $\mathcal{E}_{\mathcal{A}}$ by $\mathcal{E}'_{\mathcal{A}} := \mathfrak{C}_{\mathfrak{I}}(\mathcal{E}_{\mathcal{A}})$, we get the QN $\mathcal{N}_{2r}$, with

$$\mathrm{Out}_k(\mathcal{N}_{1r}) = \mathrm{Out}_k(\mathcal{N}_{2r}), \qquad \mathrm{Out}_{k,n}(\mathcal{N}_{1r}) = \mathrm{Out}_{k,n}(\mathcal{N}_{2r}).$$

Since in $\mathcal{N}_{2r}$ all machines have classical interfaces, the precondition holds and there is an ideal adversary $\mathcal{S}'$ independent of $\mathcal{E}$, such that for the instantiation $\mathcal{N}_{2i}$ of $\mathcal{C}_{\varrho}$ with $\mathcal{E}'_{\mathcal{A}}$, $\mathcal{S}'$, ϱ and $\mathfrak{A}$, we have

$$\mathrm{Out}_k(\mathcal{N}_{2r}) \approx \mathrm{Out}_k(\mathcal{N}_{2i}). \tag{20}$$

When replacing $\mathcal{E}'_{\mathcal{A}}$ by $\mathcal{E}_{\mathcal{A}}$ again, we get a QN $\mathcal{N}_{1i}$. Following a similar reasoning as above, it holds:

$$\mathrm{Out}_k(\mathcal{N}_{1i}) = \mathrm{Out}_k(\mathcal{N}_{2i}), \qquad \mathrm{Out}_{k,n}(\mathcal{N}_{1i}) = \mathrm{Out}_{k,n}(\mathcal{N}_{2i}).$$

We can now construct an ideal adversary $\mathcal{S}$, which simulates $\mathcal{S}'$ and $\mathcal{A}$, passing all of the communication of $\mathcal{A}$ through $\mathcal{S}'$, except for the communication with the environment. The details of this rerouting are analogous to those where we constructed $\mathcal{E}_{\mathcal{A}}$. Because $\mathcal{S}'$ was independent of $\mathcal{E}_{\mathcal{A}}$, the adversary $\mathcal{S}$ is independent of $\mathcal{E}$.

The resulting QN we call $\mathcal{N}_{0i}$. It is

$$\mathrm{Out}_k(\mathcal{N}_{0i}) = \mathrm{Out}_k(\mathcal{N}_{1i}), \qquad \mathrm{Out}_{k,n}(\mathcal{N}_{0i}) = \mathrm{Out}_{k,n}(\mathcal{N}_{1i}).$$

So we get overall:

$$\mathrm{Out}_k(\mathcal{N}_{0r}) \approx \mathrm{Out}_k(\mathcal{N}_{0i}). \tag{21}$$

Since $\mathcal{S}$ has polynomial running time relative to $\mathcal{A}$,[34] we have proven π to be as secure as ϱ.

We still have to consider the polynomial security and the security with bounded risk. We will first examine the polynomial security.

The restriction of the sets of adversaries and of environments to PITM does not change our proof at all, since all constructed machines will be polynomial in the running time of $\mathcal{A}, \mathcal{E} \in$ PITM, therefore the new machines will be in PITM themselves.

Then (20), (21) change to

$$\mathrm{Out}_{k,p(n)}(\mathcal{N}_{\cdot r}) \approx \mathrm{Out}_{k,p(n)}(\mathcal{N}_{\cdot i})$$

for a sufficiently large polynomial p. The rest of the argumentation stays same.

We will now prove the remaining case, security with bounded risk. Since we have proven, that security holds for any indistinguishability relation, it holds especially for $\approx'$, where $\approx'$ is the statistical indistinguishability with respect to $\{\varepsilon : \varepsilon = \delta \text{ almost everywhere}\}$ for a negligible bound δ. Here δ is chosen such, that $\approx'$-security is given in the classical case (the existence of such a δ is guaranteed by the definition of security with bounded risk). Then $\approx'$-security is also given in the quantum case, so in particular we have security with bounded risk in the quantum case. ■

[34] $\mathcal{S}$ simulates $\mathcal{S}'$ and $\mathcal{A}$. $\mathcal{A}'$ was of polynomial running time, $\mathcal{S}'$ was of polynomial running time relative to $\mathcal{A}'$, therefore it is polynomial, too.

9 Composability

In order to be useful, a security notion needs to fulfil the requirement of composability, i.e. the protocol being deemed secure in a stand-alone context must also be secure as part of a larger scheme. If we would for example specify some protocol and claim it to be secure, adding the restriction that it may only be executed once during the lifetime of the universe and that it may not be used in a world where some other protocol is running, our notion of security might be considered risible. Therefore a good security model should also provide composability.

In this work, we distinguish three types of composability. The first kind is the simple composability. It means, that any meta-protocol π using some protocol ϱ instead of some ideal functionality $\mathcal{F}$ will not suffer any loss of security, provided ϱ securely realises $\mathcal{F}$.

The second kind of composability is the concurrent composability. Here we claim, that if π securely realises $\mathcal{F}$, than several copies of π securely realise several copies of $\mathcal{F}$.

And the third kind is the universal composability, which is a combination of the two preceding notions, and in fact an easy corollary of those. It states, that any meta-protocol π using some protocols ϱ_i instead of some ideal functionalities $\mathcal{F}_i$ will not suffer any loss of security, provided each ϱ_i securely realises each $\mathcal{F}_i$.

We have separated the notion of composability into these three parts, because the preconditions for their appliability are different.

9.1 Simple Composability

First we will define the formal operation of simple composition, which means inserting one protocol into another. The definition will not be fully formalised, for brevity.

Definition 9.1: Simple composition of protocols
Let π and $\varrho^{(i)}$ $(i = 1, \dots, n)$ be protocols, and $F_1, \dots, F_n \in I_F$. For shorter formulation let $\varrho^{(0)} := \pi$.

Let $F(i,k) :=$ Ж$(u(i), k)$ with $i = 0, \dots, n$ and $k \in I_F$, where u is as defined in definition 2.17.

Then we define $\tilde{\pi}_j$ $(j \in I_P)$ to be an IAQS constructed as follows:

The IAQS $\tilde{\pi}_j$ simulates all IAQS $\varrho_j^{(i)}$ with $i = 0, \dots, n$. The security parameter is passed to all those upon reception. Communication with the environment is routed to π_j. Messages from and to functionalities $F(i,k)$ are routed to $\varrho_j^{(i)}$ for $i = 0, \dots, n$ as coming from resp. going to functionalities F_k. Messages from and to other functionalities are stored and not used any more.

Messages from $\tilde{\pi}_j$ to F_i are given to $\varrho_j^{(i)}$ $(i \neq 0)$ as coming from the environment. Messages from $\varrho_j^{(i)}$ $(i \neq 0)$ to the environment are given to π_j as coming from F_i.

On each invocations of $\tilde{\pi}_j$, a different $\varrho_j^{(i)}$ $(i = 0, \dots, n)$ is activated. The invocations take place in the order of growing indices i, after $\varrho_j^{(n)}$, $\varrho_j^{(0)}$'s turn comes next.

The corruption operator of π_j considers $\mathcal{H}_C$ as $\mathcal{H}_{C,\bullet} \otimes \mathcal{H}_{C,0} \otimes \cdots \otimes \mathcal{H}_{C,n}$, and for $i = 0, \dots, n$ the corruption operators for $\varrho_j^{(i)}$ is executed on the configuration space of $\varrho_j^{(i)}$ and $\mathcal{H}_{C,i}$. The whole state of the simulator is swapped onto $\mathcal{H}_{C,\bullet}$.

Then the *simple composition* $\pi^{\{F_1/\varrho^{(1)}, \dots, F_n/\varrho^{(n)}\}}$ is defined to be the protocol $\tilde{\pi}$. The associated ACF $\mathcal{C}_{\tilde{\pi}}$ is $\tilde{\mathcal{C}}_\pi \oplus \mathcal{C}_{\varrho_1} \oplus \cdots \oplus \mathcal{C}_{\varrho_n}$ (see below), where $\tilde{\mathcal{C}}_\pi$ is as $\mathcal{C}_\pi$, except that $(\tilde{\mathcal{C}}_\pi)_i := \perp$ for $i \in \{F_1, \dots, F_n\}$. Here $\perp$ is a dummy machine which never sends any message. □

In this definition, we take some protocol π, which accesses some functionalities addressed $F_1, \dots, F_n$. We then construct some new protocol $\tilde{\pi} = \pi^{\{F_1/\varrho^{(1)}, \dots, F_n/\varrho^{(n)}\}}$ by replacing every call to one of these functionalities by calls to some corresponding simulated subprotocol ϱ_i. To the environment, $\tilde{\pi}$ behaves in an unchanged way, but the functionalities get a new addressing, since now our ACF does not only contain the functionalities for π, but also those for each ϱ_i. The following definition takes care of creating this new ACF and reordering the functionalities to match to preceding definition.

Definition 9.2: Simple composition of ACF
Let $\mathcal{C}^{(0)}, \dots, \mathcal{C}^{(n)}$ be ACF. Then let the *composition* $\mathcal{C}^{(0)} \oplus \cdots \oplus \mathcal{C}^{(n)}$ be the ACF $\tilde{\mathcal{C}}$ which is defined by $\tilde{\mathcal{C}}_{F(i,k)} := \mathcal{C}_k^{(i)}$, $\tilde{\mathcal{C}}_m := \perp$ $(m \notin \operatorname{im} F)$, where F is as defined in the preceding definition, and $\perp$ is a dummy machine which never sends any message.

The access structure f of $\tilde{\mathcal{C}}$ is defined as:

$$\begin{aligned} f(F(i,k), p) &:= f_{\mathcal{C}^{(i)}}(k, p) && (i = 1, \dots, n,\ k \in I_F,\ p \in I_P) \\ f(p, F(i,k)) &:= f_{\mathcal{C}^{(i)}}(p, k) && (i = 1, \dots, n,\ k \in I_F,\ p \in I_P) \end{aligned}$$

and $f := 0$ everywhere else. □

These definitions allow us to modularly construct protocols, we may specify some protocol, which uses some abstract primitive F and later on specify some real protocol to use instead. To help showing, that the protocol thus designed is secure, we provide the following composition theorem:

Proposition 9.3: Simple composition theorem
Let $\pi^{(i)}$, $\varrho^{(i)}$, σ be protocols. Let further $\mathcal{C}_\mathcal{E}$ be one of IAQS, IQTM, PIAQS, ICTM, IQCTM, or any non-empty intersection of these, and $\mathcal{C}_\mathcal{A}$ one of IAQS, IQTM, PIAQS, ICTM, IQCTM, the set of (stochastically/polynomially) non-blocking adversaries, or any non-empty intersection of these. The ACF $\mathcal{C}_\sigma$ associated to σ shall be such, that all its functionalities can be simulated by a machine in $\mathcal{C}_\mathcal{E}$.[35] Let $\mathcal{B}$ be any set of bounds. Let $F_i \in I_F$ ($i = 1, \ldots, n$, $n \in \mathbb{N}_0$) be pairwise different.

Then it is

$$\forall i \in \{1, \ldots, n\} : \pi^{(i)} \geq \varrho^{(i)} \quad \Longrightarrow \quad \sigma^{\{F_1/\pi^{(1)}, \ldots, F_n/\pi^{(n)}\}} \geq \sigma^{\{F_1/\varrho^{(1)}, \ldots, F_n/\varrho^{(n)}\}}$$

where $\geq$ means approximative or absolute security, security with respect to $\mathcal{B}$, polynomial security, security with bounded risk or any combination of these. □

Note that the two sides of the implication have different ACF, i.e. the respective associated ACF for the different protocols.

Proof sketch: We will restrict the proposition to the case $n = 1$. The case $n > 1$ is proven inductively. We then write $F := F_1$, $\pi := \pi^{(1)}$, $\varrho := \varrho^{(1)}$. Let further $\mathcal{F}$ be the functionality with ID F in $\mathcal{C}_\sigma$. When writing $\mathrm{Out}_k(\mathcal{E}, \mathcal{A}, \pi)$ we mean $\mathrm{Out}_k(\mathcal{C}_\pi(x, \mathcal{E}, \mathcal{A}, \mathfrak{A}, \pi))$, where x is *adaptive* or *static*, depending on the chosen security model $\geq$.

Assume that $\pi \geq \varrho$ and $\sigma^{F/\pi} \not\geq \sigma^{F/\varrho}$. Then there is an adversary $\mathcal{A}$, s.t.

$$\forall \mathcal{S} \in \mathcal{C}_\mathcal{A} \; \exists \mathcal{E} \in \mathcal{C}_\mathcal{E} : \qquad (\mathrm{Out}_k(\mathcal{E}, \mathcal{A}, \sigma^{F/\pi})) \not\approx (\mathrm{Out}_k(\mathcal{E}, \mathcal{S}, \sigma^{F/\varrho})). \tag{22}$$

We then construct an environment $\tilde{\mathcal{E}}$, which simulates $\mathcal{E}$, all functionalities in $\mathcal{C}_\sigma$ and all parties in σ. Communication between these machines is routed internally, the outside communication occurring is:

- Communication between the parties in σ and $\mathcal{F}$. These accesses to $\mathcal{F}$ are rewritten to accesses to the protocol outside $\tilde{\mathcal{E}}$, i.e. when σ_i sends a message to $\mathcal{F}$, it is sent to the party i outside $\tilde{\mathcal{E}}$, and vice versa.
- Communication between $\mathcal{E}$ and $\mathcal{A}$. This communication (and that of the following two points) is routed to $\tilde{\mathcal{A}}$ (the adversary, see below), together with additional messages informing $\tilde{\mathcal{A}}$ of the communication partner.
- Communication between the functionalities in $\mathcal{C}_\sigma$ and $\mathcal{A}$. See the previous point.
- Corruptions of parties in σ by $\mathcal{A}$. The corruption of parties is—from the point of view of communication—simply a message transmission, so see the previous two points.

We will not cover details of this message transmission and of the scheduling in this proof sketch.

The adversary $\tilde{\mathcal{A}}$ shall be constructed thus, that it simulates $\mathcal{A}$ and reroutes communication with $\mathcal{E}$ and functionalities in $\mathcal{C}_\sigma$ to the corresponding parts of $\tilde{\mathcal{E}}$. When corrupting a party, that party shall be corrupted and additionally the simulated party in $\tilde{\mathcal{E}}$ shall be corrupted. From the states of these two corrupted machines we construct the state, the corresponding composed machine $\sigma_i^{F/\varrho}$ in $\sigma^{F/\varrho}$ would have, assuming that the states of the two simulated machines in $\sigma_i^{F/\varrho}$ are the two states we just got as the result of our corruption. That constructed state is given to $\mathcal{A}$ as the results of the corruption. Communication with functionalities in $\mathcal{C}_\pi$ is simply passed through.

Then it is

$$\forall \mathcal{E} \in \mathcal{C}_\mathcal{E} : \qquad (\mathrm{Out}_k(\mathcal{E}, \mathcal{A}, \sigma^{F/\pi})) = (\mathrm{Out}_k(\tilde{\mathcal{E}}, \tilde{\mathcal{A}}, \pi)). \tag{23}$$

Since $\pi \geq \varrho$, there is an $\tilde{\mathcal{S}} \in \mathcal{C}_\mathcal{A}$, s.t.

$$\forall \mathcal{E} \in \mathcal{C}_\mathcal{E} : \qquad (\mathrm{Out}_k(\tilde{\mathcal{E}}, \tilde{\mathcal{A}}, \pi)) \approx (\mathrm{Out}_k(\tilde{\mathcal{E}}, \tilde{\mathcal{S}}, \varrho)). \tag{24}$$

[35] This is e.g. satisfied, if $\mathcal{C}_\sigma$ contains only a finite number of non-dummy machines and they are all in $\mathcal{C}_\mathcal{E}$.

Since otherwise $\tilde{\mathcal{S}}$ would be easily distinguishable from $\tilde{\mathcal{A}}$, it does obey with overwhelming probability (or with probability 1, in case of absolute security) the communication protocol between $\tilde{\mathcal{E}}$ and $\tilde{\mathcal{A}}$. Therefore we can replace it by an adversary, which fully conforms to that protocol. So we will w.l.o.g. assume such conformity. Then we can construct an adversary $\mathcal{S}$, which simulates $\tilde{\mathcal{S}}$ and reroutes the communication from $\tilde{\mathcal{S}}$ as follows:

- Communication with the functionalities in $\mathcal{C}_\varrho$ is simply passed through.
- Communication with any simulated machines in $\tilde{\mathcal{E}}$ (i.e. $\mathcal{E}$, parties in σ, functionalities in $\mathcal{C}_\sigma$) is routed to the corresponding real machines.
- Corruptions requested by $\tilde{\mathcal{S}}$ are executed and then—assuming that a machine in $\sigma^{F/\varrho}$ is corrupted—split into two parts, the part a corruption of the corresponding machine in ϱ would have yielded, and the part $\tilde{\mathcal{E}}$ would have simulated with respect to σ.

Then we have

$$\forall \mathcal{E} \in \mathcal{C}_\mathcal{E} : \qquad (\mathrm{Out}_k(\tilde{\mathcal{E}}, \tilde{\mathcal{S}}, \varrho)) = (\mathrm{Out}_k(\mathcal{E}, \mathcal{S}, \sigma^{F/\varrho})),$$

which gives, together with (23) and (24)

$$\forall \mathcal{E} \in \mathcal{C}_\mathcal{E} : \quad (\mathrm{Out}_k(\mathcal{E}, \mathcal{A}, \sigma^{F/\pi})) \approx (\mathrm{Out}_k(\mathcal{E}, \mathcal{S}, \sigma^{F/\varrho})),$$

in contradiction to (22).

The case of security under polynomial running time, and the case of polynomial security are analogous, simply replace Out_k by $\mathrm{Out}_{k,n}$ and quantify over all n.

In the case of security with bounded risk, the bound valid for $\pi \geq \varrho$ is valid for $\sigma^{F/\pi} \geq \sigma^{F/\varrho}$, too, due to the previously shown, so the proposition also holds for security with bounded risk. ■

The most frequent application of this theorem is the following: We have some protocol π, which, when using some functionality $\mathcal{F}$ with ID F, is in some way secure, we than want to get security also for $\pi^{F/\varrho}$, provided that ϱ is as secure as $\mathcal{F}$. For this purpose we state the following corollary, of which we omit the proof:[36]

Corollary 9.4
Let the preconditions and variables be as in proposition 9.3 and $\mathcal{F}^{(i)} := (\mathcal{C}_\sigma)_{F_i}$.

Then it is

$$\sigma \geq \tau \quad \text{and} \quad \forall i \in \{1, \ldots, n\} : \pi^{(i)} \geq \mathcal{F}^{(i)} \quad \Longrightarrow \quad \sigma^{\{F_1/\pi^{(1)}, \ldots, F_n/\pi^{(n)}\}} \geq \tau,$$

with the same meaning of the symbol $\geq$ as in proposition 9.3, with the additional constraint, that $\mathcal{B}$ is closed under addition.[37] □

9.2 Concurrent Composability

We will now define the formal operation of parallelisation, i.e. running a protocol several times in parallel. Again we will not fully formalise the definition.

Definition 9.5: Parallelisation of IAQS
Let $\mathcal{M}$ be an IAQS.

We then define the *parallelisation* $\mathcal{M}^m$ *of* $\mathcal{M}$, with $m : \mathbb{N}_0 \to \mathbb{N}_0 \cup \{\infty\}$, to be an IAQS constructed as follows:

Let k be the security parameter. Then the machine $\mathcal{M}^m$ simulates $m(k)$ copies of $\mathcal{M}$ (possibly an infinite number), indexed by $i \in \{1, \ldots, m(k)\}$ (i.e. $\mathbb{N}$ in the case $m(k) = \infty$), here denoted as $\mathcal{M}_i$, $i \in \mathbb{N}$. We call i the *session ID (SID).* of the indexed machine.

On each second invocation of $\mathcal{M}^m$ one simulated machine $\mathcal{M}_i$ is invoked, the value of i first increases from 1 to 1, then from 1 to 2, then from 1 to 3, etc., until it reaches the cycle 1 to $m(k)$, where it keeps cycling.

[36]The proof can be constructed by taking into account, that $\sigma^{\{F_1/\pi^{(1)}, \ldots, F_n/\pi^{(n)}\}}$ is essentially the same as σ, if $\pi^{(i)}$ is a dummy protocol simply rerouting messages to $\mathcal{F}^{(i)}$, and that $\leq$ is transitive, if $\mathcal{B}$ is closed under addition.

[37]I.e. for $a, b \in \mathcal{B}$ there is a $c \in \mathcal{B}$, s.t. $ab \leq c$.

Communication is routed to the outside, with the following encapsulation: When a message is to be transmitted to the outside of $\mathcal{M}^m$ from $\mathcal{M}_i$, a message containing only the index i is send to the recipient, and only after the next invocation the actual message is sent.[38]

Similarly, when a message comes in, it is taken to be an index i, and the next message from the *same* sender is taken to contain a message for $\mathcal{M}_i$ (and is not taken to be an index, of course).

A corruption gives the full state of the simulator, and evaluates the corruption operators of all $\mathcal{M}_i$ invoked so far (analogous to definition 9.1).

When machines with invalid indices are accessed (i.e. indices greater $m(k)$, or syntactically incorrect ones), dummy machines are simulated. □

This definition takes a single machine $\mathcal{M}$, and constructs a machine $\mathcal{M}^m$ containing a whole bunch of identical copies of $\mathcal{M}$. Each incoming and outgoing communication can then be addressed to or coming from a particular copy of $\mathcal{M}$. We write a function into the superscript of the newly constructed machine $\mathcal{M}^m$, which indicates whether and to what value the number of copies is then restricted (in dependence of the security parameter). If $m = \infty$, we have an unbounded number of copies available.

This construct is useful as well for parties (when used in the context of the following definition), as well as when used with functionalities, e.g. a functionality for a single common coin toss becomes a functionality which provides a (bounded or unbounded) number of coin tosses.

When we apply this construct to parties, it is of course only sensible, if all parties undergo the same change, and if all functionalities they use are also parallelised. This is achieved by the following two definitions:

Definition 9.6: Parallelisation of protocols
Let π be a protocol. The *parallelisation* π^m *of* π, $m : \mathbb{N}_0 \to \mathbb{N}_0 \cup \{\infty\}$ is defined by $(\pi^m)_i := (\pi_i)^m$, $i \in I_P$. The associated ACF $\mathcal{C}_{\pi^m}$ is the parallelisation $(\mathcal{C}_\pi)^m$ (see below). □

Definition 9.7: Parallelisation of ACF
Let $\mathcal{C}$ be an ACF. Then the *parallelisation* $\mathcal{C}^m$ *of* $\mathcal{C}$, $m : \mathbb{N}_0 \to \mathbb{N}_0 \cup \{\infty\}$ is defined by $(\mathcal{C}^m)_i := (\mathcal{C}_i)^m$, $i \in I_F$. The access structure of $\mathcal{C}^m$ is the same as that of $\mathcal{C}$. □

Proposition 9.8: Concurrent composition theorem
Let π, ϱ be protocols. Let further $\mathcal{C}_\mathcal{E}$ be one of IAQS, IQTM, PIAQS, ICTM, IQCTM, or any non-empty intersection of these, and $\mathcal{C}_\mathcal{A}$ one of IAQS, IQTM, PIAQS, ICTM, IQCTM, the set of (stochastically/polynomially) non-blocking adversaries, or any non-empty intersection of these. Let $m : \mathbb{N}_0 \to \mathbb{N}_0$ be any function and $\mathcal{B}$ a set of bounds, which is closed under multiplication with m.[39] All of the following should be doable by a machine in $\mathcal{C}_\mathcal{E}$:

- Simulate the parties in the protocols π, ϱ.
- Simulate the functionalities in the ACF $\mathcal{C}_\pi$ and $\mathcal{C}_\varrho$ associated to π resp. ϱ.
- Simulate any machine from $\mathcal{C}_\mathcal{A}$.
- Evaluate m.

Then it is
$$\pi \geq \varrho \quad \Longrightarrow \quad \pi^m \geq \varrho^m,$$
where $\geq$ means security with respect to $\mathcal{B}$, polynomial security with respect to $\mathcal{B}$, security with bounded risk and with respect to $\mathcal{B}$ or any combination of these.

If we restrict $\geq$ to polynomial security with respect to $\mathcal{B}$, and require $\mathcal{B}$ to be closed under multiplication with polynomials, then we can omit the restriction, that $\mathcal{B}$ must be closed under multiplication with m. □

Note that the two sides of the implication have different ACF, namely the respective associated ACF for the different protocols.

Proof sketch: When writing $\mathrm{Out}_k(\mathcal{E}, \mathcal{A}, \pi)$ we mean $\mathrm{Out}_k(\mathcal{C}_\pi(x, \mathcal{E}, \mathcal{A}, \mathfrak{A}, \pi))$, where x is *adaptive* or *static*, depending on the chosen security model $\geq$.

Assume that $\pi \geq \varrho$, and let $\mathcal{A}$ be an arbitrary well-formed adversary from $\mathcal{C}_\mathcal{A}$. Our goal is to construct a well-formed adversary $\mathcal{S}$, such that
$$\forall \mathcal{E} \in \mathcal{C}_\mathcal{E} : \quad (\mathrm{Out}_k(\mathcal{E}, \mathcal{A}, \pi^m)) \approx (\mathrm{Out}_k(\mathcal{E}, \mathcal{S}, \pi^m)).$$

[38] This is due to the fact, that we cannot add that ID to the message itself without measuring parts of its content.
[39] I.e. for any $b \in \mathcal{B}$, there is a $\tilde{b} \in \mathcal{B}$, s.t. $mb \leq \tilde{b}$.

If we have allowed an unrestricted m, then let m' be the maximum number of invocations of $\mathcal{A}$. Since in this case we have polynomial security, and because of the definition of the well-formedness of the adversary, we can choose m' as a polynomial only depending on the adversary and on the polynomial limiting the number of time blocks of the QN (see the definition of security under polynomial running time, definition 6.2).

If we have instead restricted m, then let instead $m' := m$.

Note that $\mathcal{B}$ is closed under multiplication with m' in both cases.

We will now construct some auxiliary objects:

Let $\mathcal{A}_0$ be a dummy adversary that simply reroutes any messages or corruption requests. For details, see the definition of $\mathcal{A}'$ in the proof for proposition 8.1, which fulfils the same tasks.

Let then $\mathcal{S}_0$ the corresponding ideal adversary, fulfilling

$$\forall \mathcal{E} \in \mathcal{C}_\mathcal{E}: \quad (\mathrm{Out}_k(\mathcal{E}, \mathcal{A}_0, \pi)) \approx (\mathrm{Out}_k(\mathcal{E}, \mathcal{S}_0, \varrho)). \tag{25}$$

Such an $\mathcal{S}_0$ exists, since $\pi \geq \varrho$.

Let $\pi\varrho^m$ be the protocol that, similar to the composition π^m, simulates several concurrent protocols, with the difference, that for some SIDs i the machines π_i are simulated, for some SIDs i the machines ϱ_i instead. Which kind of machine is to be simulated is queried from a special functionality $\mathcal{F}^*$ (see below) upon the first activation of that copy. When accessing invalid SIDs (greater m or syntactically invalid), dummy machines are simulated as with π^m.

Let $\mathcal{AS}^m$ be an adversary constructed as follows: It simulates $\mathcal{A}$, with the slight variation, that it does not directly access functionalities or parties any more, but instead does this through copies of $\mathcal{A}_0$, which are then connected with the corresponding non-simulated copies of the protocols and their respective functionalities. Instead of simply simulating copies of $\mathcal{A}_0$, $\mathcal{AS}^m$ queries the functionality $\mathcal{F}^*$ upon the first activation of a copy of $\mathcal{A}_0$, and then, if $\mathcal{F}^*$ says, that for that particular SID a copy of ϱ is executed, $\mathcal{S}_0$ is executed instead of $\mathcal{A}_0$, so that the choice of the adversary ($\mathcal{A}_0$ or $\mathcal{S}_0$) matches the corresponding copy of the protocol.

The functionality $\mathcal{F}_i$ simulates copies of functionalities $(\mathcal{C}_\pi)_i$ or $(\mathcal{C}_\varrho)_i$, for each SID querying $\mathcal{F}^*$ in order to know, whether to simulate functionalities for π or for ϱ. The associated ACF for $\pi\varrho^m$ is then the ACF consisting of the $\mathcal{F}_i$ thus constructed. In order to do this, we have to make a small change to our scheduling function by allowing, that functionalities may communicate with $\mathcal{F}^*$.

To the associated ACF for $\pi\varrho^m$ a new functionality $\mathcal{F}^*$ is added (remapping the IDs of the other functionalities to create space and rewriting all concerned machines to use the new IDs). The functionality $\mathcal{F}^* = \mathcal{F}^*(l)$ depends on a parameter l, and operates as follows: Whenever it is queried concerning an SID i, it does the following:

- If i is invalid (greater m or syntactically invalid), return $\perp$.
- If i has already been queried by some machine, return the previously given answer.
- Increase *num* (a global counter, initialised with 0 upon the first activation of $\mathcal{F}^*$).
- If $num \leq l$, return π.
- If $num > l$, return ϱ.

To denote, given which value of l we construct $\mathcal{F}^*$, we add $\mathcal{F}^*(l)$ to the argument list of Out_k.

Using this new constructions, we get

$$\forall \mathcal{E} \in \mathcal{C}_\mathcal{E}: \quad \mathrm{Out}_k(\mathcal{E}, \mathcal{A}, \pi^m) = \mathrm{Out}_k(\mathcal{E}, \mathcal{AS}^m, \pi\varrho^m, \mathcal{F}^*(m'(k))), \tag{26}$$

since m' is constructed thus, that $\mathcal{F}^*$ cannot be invoked with more that m' different valid SIDs.

We now construct a new environment $\mathcal{E}^* = \mathcal{E}^*(l)$, depending on a parameter $l \in \mathbb{N}_0$. This environment simulates the environment $\mathcal{E}$, copies of of π resp. ϱ together with their ACFs, the adversary $\mathcal{A}$ (as modified before, i.e. using intermediate adversaries $\mathcal{A}_0$ resp. $\mathcal{S}_0$), and the copies of the adversaries $\mathcal{A}_0$, $\mathcal{S}_0$. Instead of querying a functionality $\mathcal{F}^*$, however, $\mathcal{E}^*$ decides itself, whether to use the protocol π or the protocol ϱ for a given SID (and the functionalities in $\mathcal{C}_\pi$ resp. $\mathcal{C}_\varrho$, and the adversaries $\mathcal{A}_0$ resp. $\mathcal{S}_0$). It does this according to the following rules:

- If a protocol/functionality/adversary with invalid SID (greater $m(k)$ or syntactically incorrect) is activated, simulate a dummy machine.
- If a protocol/functionality/adversary with SID i is activated, and i is the n-th SID used, and $n < l$, decide in favour of π, $\mathcal{C}_\pi$ and $\mathcal{A}_0$ for this SID.

- If a protocol/functionality/adversary with SID i is activated, and i is the n-th SID used, and $n > l$, decide in favour of ϱ, $\mathcal{C}_\varrho$ and $\mathcal{S}_0$ for this SID.
- If a protocol/functionality/adversary with SID i is activated, and i is the l-th SID used, then do not simulate the objects belonging to that SID, but route messages to them to the outside of $\mathcal{E}^*$.

Using a thus constructed environment $\mathcal{E}^*(l)$, we get for $l \geq 0$:

$$\forall \mathcal{E} \in \mathcal{C}_\mathcal{E} : \quad \mathrm{Out}_k(\mathcal{E}^*(l), \mathcal{A}_0, \pi) = \mathrm{Out}_k(\mathcal{E}, \mathcal{AS}^m, \pi\varrho^m, \mathcal{F}^*(l)) \tag{27}$$
$$\forall \mathcal{E} \in \mathcal{C}_\mathcal{E} : \quad \mathrm{Out}_k(\mathcal{E}^*(l), \mathcal{S}_0, \varrho) = \mathrm{Out}_k(\mathcal{E}, \mathcal{AS}^m, \pi\varrho^m, \mathcal{F}^*(l+1)) \tag{28}$$

We now simply define $\mathcal{S}$ by modifying $\mathcal{AS}^0$ thus, that instead of querying $\mathcal{F}^*$, it simply assumes the answer to be ϱ, which gives the equality:

$$\forall \mathcal{E} \in \mathcal{C}_\mathcal{E} : \quad (\mathrm{Out}_k(\mathcal{E}, \mathcal{S}, \varrho^m)) = (\mathrm{Out}_k(\mathcal{E}, \mathcal{AS}^m, \pi\varrho^m, \mathcal{F}^*(0))) \tag{29}$$

Then we get for all $\mathcal{E} \in \mathcal{C}_\mathcal{E}$:

$$\begin{aligned}
& \frac{1}{2} \sum_{a \in \{0,1\}} |\mathrm{Out}_k(\mathcal{E}, \mathcal{A}, \pi^m) - \mathrm{Out}_k(\mathcal{E}, \mathcal{S}, \varrho^m)| \\
\overset{(26,29)}{=} \; & \frac{1}{2} \sum_{a \in \{0,1\}} |\mathrm{Out}_k(\mathcal{E}, \mathcal{AS}^m, \pi\varrho^m, \mathcal{F}^*(m'(k))) - \mathrm{Out}_k(\mathcal{E}, \mathcal{AS}^m, \pi\varrho^m, \mathcal{F}^*(0))| \\
= \; & \frac{1}{2} \sum_{a \in \{0,1\}} \left| \sum_{l=1}^{m'(k)} \mathrm{Out}_k(\mathcal{E}, \mathcal{AS}^m, \pi\varrho^m, \mathcal{F}^*(l)) - \sum_{l=1}^{m'(k)} \mathrm{Out}_k(\mathcal{E}, \mathcal{AS}^m, \pi\varrho^m, \mathcal{F}^*(l-1)) \right| \\
\overset{(27,28)}{=} \; & \frac{1}{2} \sum_{a \in \{0,1\}} \left| \sum_{l=1}^{m'(k)} \mathrm{Out}_k(\mathcal{E}^*(l), \mathcal{A}_0, \pi) - \sum_{l=1}^{m'(k)} \mathrm{Out}_k(\mathcal{E}^*(l), \mathcal{S}_0, \varrho) \right| \\
\leq \; & \sum_{l=1}^{m'(k)} \frac{1}{2} \sum_{a \in \{0,1\}} \left| \mathrm{Out}_k(\mathcal{E}^*(l), \mathcal{A}_0, \pi) - \mathrm{Out}_k(\mathcal{E}^*(l), \mathcal{S}_0, \varrho) \right| \\
\overset{(25)}{\leq} \; & m'(k)\varepsilon(k)
\end{aligned}$$

for some $\varepsilon \in \mathcal{B}$ (possibly depending on $\mathcal{E}$ and $\mathcal{A}$). Since $\mathcal{B}$ is closed under multiplication with m', it is

$$\forall \mathcal{E} \in \mathcal{C}_\mathcal{E} : \qquad (\mathrm{Out}_k(\mathcal{E}, \mathcal{A}, \pi^m)) \approx (\mathrm{Out}_k(\mathcal{E}, \mathcal{S}, \varrho^m)),$$

so $\pi^m \geq \varrho^m$.

We still have two cases to check: First the polynomial security, then the security with bounded risk.

In the case of security under polynomial running time, the argument given above holds for any polynomial p bounding the number of time blocks (see definition 6.2), and $\mathcal{S}$ is constructed independently of that p. So the proposition does also hold for polynomial security.

In the case of security with bounded risk, the function $m' = m$ is fixed, and also ε can be chosen independently of $\mathcal{E}$ and $\mathcal{A}$, thus $m'\varepsilon$ is fixed, too, therefore security with bounded risk is also given when comparing π^m and ϱ^m. ■

9.3 Universal Composability

For the security model in [Can00b] there is a proof in that work for *universal composability*. This concept is simply the combination of the two previous composability concepts. It states, that when some protocol π using multiple copies of some functionality $\mathcal{F}$ is secure, then the same protocol π using multiple copies of some protocol ϱ implementing the functionality $\mathcal{F}$ securely is also secure. In our framework, this can easily be proven in two steps, which are here informally given: First the concurrent composability ensures (under the conditions given in proposition 9.8), that multiple copies of the protocol ϱ are a valid replacement for multiple copies of the functionality, and then because of the simple composability (see proposition 9.3 for details) we can see, that π with the copies of ϱ is as secure as π with the copies of $\mathcal{F}$.

A Formal details

A.1 Lemma 3.15

In section 3.4 we have stated the following lemma, of which we will now give the proof:

Lemma 3.15
Let $\mathcal{M}$ be a classical IAQS and ι some quantum interaction for $\mathcal{M}$ consisting of operators (O_i).

Let $M_i : \mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^2 \to \mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^2$ $(i = 1, \ldots, n+1)$ be observables with eigenspaces spanned by vectors of the standard basis of $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^2$, where $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes |0\rangle$ should be completely contained in one of the eigenspaces.

We define the mixed states

$$\varrho_1 := \ddot{M}_r \ddot{M}_f \ddot{M}^{(i)} \prod_{i=1}^{n} (\ddot{O}_i \ddot{M}_r \ddot{M}_f \ddot{M}^{(i)})\, \varrho(|\Psi_{0,\mathcal{M}}\rangle \otimes |\#\rangle \otimes |\#\rangle \otimes |0\rangle \otimes |0\rangle),$$

$$\varrho_2 := \ddot{M}_r \ddot{M}_f \ddot{M}^{(i)} \ddot{M}_{n+1} \prod_{i=1}^{n} (\ddot{O}_i \ddot{M}_r \ddot{M}_f \ddot{M}^{(i)} \ddot{M}_i)\, \varrho(|\Psi_{0,\mathcal{M}}\rangle \otimes |\#\rangle \otimes |\#\rangle \otimes |0\rangle \otimes |0\rangle),$$

in $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^2$, where M_f is the standard basis measurement of $\mathcal{H}_F$, M_r that of $\mathbb{C}^2$, and $M^{(i)}$ is the measurement $\{P_k^{(i)}, k \in \{\mathit{run}, \mathit{send}, \mathit{rec}, \mathit{world}\}\}$, with $P_k^{(i)}$ as in the definition of quantum interactions (definition 3.2). $\ddot{X}$ denotes the density matrix operator for X.

Let M be any observable satisfying for each of its eigenspaces E, that

$$(\langle\tilde{q}| \otimes \langle\tilde{f}| \otimes \langle\tilde{c}| \otimes \mathbb{1} \otimes \mathbb{1}) P_E (|q\rangle \otimes |f\rangle \otimes |c\rangle \otimes \mathbb{1} \otimes \mathbb{1}) = 0$$

for any basis states $|\tilde{q}\rangle, |q\rangle \in \mathcal{H}_{Q,\mathcal{M}}$, $|\tilde{f}\rangle, |f\rangle \in \mathcal{H}_F$, $|\tilde{c}\rangle, |c\rangle \in \mathcal{H}_C$ with $|\tilde{q}\rangle \otimes |\tilde{f}\rangle \otimes |\tilde{c}\rangle \neq |q\rangle \otimes |f\rangle \otimes |c\rangle$, where P_E is the projection onto E.

Then the outcomes of M-measuring ϱ_1 and ϱ_2 have the same probability distribution. □

We will now proceed with the proof of this lemma:

Let $r_i \in \{0, 1\}$, $f_i \in \Sigma^*$, and $k_i \in \{\mathit{run}, \mathit{send}, \mathit{rec}, \mathit{world}\}$ $(i = 1, \ldots, n)$ have arbitrary values. Then we define

$$\varrho_1^{(k)} := \prod_{i=1}^{k} (\ddot{P}_{r_{i+1}} \ddot{O}_i \ddot{P}_{r_i} \ddot{P}_{f_i} \ddot{P}_{k_i}^{(i)})\, \varrho(\underbrace{|\Psi_{0,\mathcal{M}}\rangle \otimes |\#\rangle \otimes |\#\rangle \otimes |0\rangle \otimes |0\rangle}_{=:|S\rangle})$$

$$\varrho_2^{(k)} := \prod_{i=1}^{k} (\ddot{M}_{i+1} \ddot{P}_{r_{i+1}} \ddot{O}_i \ddot{P}_{r_i} \ddot{P}_{f_i} \ddot{P}_{k_i}^{(i)}) \ddot{M}_1\, \varrho(|S\rangle)$$

where P_{r_i} denotes the projection of $\mathbb{C}^2$ onto $|r_i\rangle$, P_{f_i} denotes the projection of $\mathcal{H}_F$ onto $|f_i\rangle$, and $P_{k_i}^{(i)}$ are the projections from the quantum interaction ι (see definition 3.2).

Let

$$\Gamma := \{|q\rangle \otimes |f\rangle \otimes |c\rangle \otimes \mathbb{C}^{\mathbb{Z}} \otimes |1\rangle \subseteq \mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^2 : q, c \in \Sigma_{\mathrm{fin}}^{\mathbb{Z}}, f \in \Sigma^*\}$$
$$\cup \{|q\rangle \otimes \mathcal{H}_F \otimes \mathcal{H}_{Q,\mathcal{M}} \otimes \mathbb{C}^{\mathbb{Z}} \otimes |0\rangle \subseteq \mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^2 : q \in \Sigma_{\mathrm{fin}}^{\mathbb{Z}}\}.$$

If for two density matrices ϱ, σ there is a decomposition

$$\varrho = |\Psi^{\varrho}\rangle\langle\Psi^{\varrho}|, \qquad \sigma = \sum_{\nu} |\Psi_{\nu}^{\sigma}\rangle\langle\Psi_{\nu}^{\sigma}|$$

such that $\sum_i |\Psi_i^{\sigma}\rangle = |\Psi^{\varrho}\rangle$, and such that the sets Γ_i^{σ} are disjunct, when defined as the minimal subsets of Γ satisfying $|\Psi_i^{\sigma}\rangle \in \bigoplus_{\mathcal{H} \in \Gamma_i^{\sigma}} \mathcal{H}$, we will say that σ *is compatible to* ϱ in the context of this proof.

We will now show by induction, that $\varrho_2^{(k)}$ is compatible to $\varrho_1^{(k)}$ for $k = 1, \ldots, n$.

For $k = 0$ this is easy: $\varrho_1^{(0)} = |S\rangle$ and $\varrho_2^{(0)} = M_1|S\rangle$. Since M_1 is defined such, that all its eigenspaces are direct sums of elements of Γ, so an application of M_1 splits $|S\rangle$ into a sum of vectors $|\Psi_i\rangle$, each lying in an own $\bigoplus_{\mathcal{H} \in \Gamma_i} \mathcal{H}$ with disjunct $\Gamma_i \subseteq \Gamma$. Therefore $\varrho_2^{(0)}$ is compatible to $\varrho_1^{(0)}$.

We can now assume that $\varrho_2^{(k)}$ is compatible to $\varrho_1^{(k)}$ for some $k \in \{1, \dots, n-1\}$, and will therewith prove that $\varrho_2^{(k+1)}$ is compatible to $\varrho_1^{(k+1)}$, thus completing the induction proof.

As can be seen from the definition of Γ and $P_{r_{k+1}}, P_{f_{k+1}}, P_{k_{k+1}}$, it is $P\mathcal{H} \subseteq \mathcal{H}$ for $P \in \{P_{r_{k+1}}, P_{f_{k+1}}, P_{k_{k+1}}\}$ and any $\mathcal{H} \in \Gamma$. Therefore if ϱ is compatible to σ, so is $\ddot{P}\varrho$ to $\ddot{P}\sigma$. We get by thrice applying this result that $\varrho_2' := \ddot{P}_{r_{k+1}} \ddot{P}_{f_{k+1}} \ddot{P}^{(i)}_{k_{k+1}} \varrho_2^{(k)}$ is compatible to $\varrho_1' := \ddot{P}_{r_{k+1}} \ddot{P}_{f_{k+1}} \ddot{P}^{(i)}_{k_{k+1}} \varrho_1^{(k)}$.

We will now examine how ϱ_1' and ϱ_2' behave under application of O_{k+1}. In order to do so, we have to distinguish five cases:

1. $r_{k+1} = 1$,
2. $r_{k+1} = 0$, $k_{k+1} = \mathit{run}$,
3. $r_{k+1} = 0$, $k_{k+1} = \mathit{send}$,
4. $r_{k+1} = 0$, $k_{k+1} = \mathit{rec}$,
5. $r_{k+1} = 0$, $k_{k+1} = \mathit{world}$.

Case 1 ($r_{k+1} = 1$): In this case ϱ_1' and ϱ_2' have been projected to have $|1\rangle$ in $\mathbb{C}^2$. Therefore $P_{r_{k+2}} O_{k+1}$ operates on ϱ_1' and ϱ_2' by definition as

$$\begin{aligned} \ddot{P}_{r_{k+2}} \ddot{O}_{k+1} \varrho_{1,2}' &= \ddot{P}_{r_{k+2}} \ddot{U}_{\mathrm{FS},\mathcal{M},\mathrm{cond}} \ddot{U}_{\mathrm{F},\mathcal{M}} \ddot{U}_{\mathcal{M}} \varrho_{1,2}' \\ &= \ddot{P}_{r_{k+2}} \ddot{U}_{\mathrm{FS},\mathcal{M},\mathrm{cond}} \ddot{P}_{r_{k+2}} \ddot{U}_{\mathrm{F},\mathcal{M}} \ddot{U}_{\mathcal{M}} \varrho_{1,2}' \end{aligned} \tag{30}$$

where

$$\begin{aligned} U_{\mathcal{M}} &= \tilde{U}_{\mathcal{M}} \cdot U_{\mathrm{copy},\mathcal{M}} \\ U_{\mathrm{FS},\mathcal{M},\mathrm{cond}} &:= (U_{\mathrm{copy},\mathcal{M}} \cdot \tilde{U}_{\mathrm{FS},\mathcal{M}} \cdot U_{\mathrm{copy},\mathcal{M}}) \otimes P_0 + \mathbb{1} \otimes P_1 \\ U_{\mathrm{F},\mathcal{M}} &:= P^{\complement}_{\mathrm{F},\mathcal{M}} \otimes \mathbb{1} + P_{\mathrm{F},\mathcal{M}} \otimes X \end{aligned}$$

and $X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right) \in U(\mathbb{C}^2)$ is the bit flip. The second equality in (30) is due to the fact, that P_0 and P_1 commute, and therefore also $P_{r_{k+2}}$ and $U_{\mathrm{FS},\mathcal{M},\mathrm{cond}}$.

Since $\text{Ѫ}(q, m_1, \dots) \mapsto \text{Ѫ}(q, q, m_1, \dots)$ is injective, each element of Γ is mapped by $U_{\mathrm{copy},\mathcal{M}}$ into some unique element of Γ. Therefore $U_{\mathrm{copy},\mathcal{M}} \varrho_2'$ is compatible to $U_{\mathrm{copy},\mathcal{M}} \varrho_1'$.

Let for brevity be $\Gamma_i := \Gamma_i^{U_{\mathrm{copy},\mathcal{M}} \varrho_2'}$. Then no two different elements of $\bigcup_i \Gamma_i$ lie in the same $\Delta_{m_1,\dots}$ with

$$\Delta_{m_1,\dots} := \operatorname{span}\{|\text{Ѫ}(q, m_1, \dots)\rangle, q \in \Sigma^{\mathbb{Z}}_{\mathrm{fin}}\} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^2 \qquad (m_1, m_2, \dots \in \Sigma^{\mathbb{Z}}_{\mathrm{fin}}),$$

i.e. each element of $\bigcup_i \Gamma_i$ is uniquely defined by the content of the measurement tapes.

Since $\tilde{U}_{\mathcal{M}} \Delta_{m_1,\dots} \subseteq \Delta_{m_1,\dots}$ (by definition of $\tilde{U}_{\mathcal{M}}$, equation 15), each $\bigoplus_{\mathcal{H} \in \Gamma_i} \mathcal{H}$ is mapped by $\tilde{U}_{\mathcal{M}}$ into a $\bigoplus_{m_1,\dots \in T_i} \Delta_{m_1,\dots}$, with disjunct T_i, and since each $\Delta_{m_1,\dots}$ is a direct sum of elements of Γ, the compatibility condition is again satisfied for $\ddot{U}_{\mathcal{M}} \varrho_1' = \ddot{\tilde{U}}_{\mathcal{M}} \ddot{U}_{\mathrm{copy},\mathcal{M}} \varrho_1'$ and $\ddot{U}_{\mathcal{M}} \varrho_2' = \ddot{\tilde{U}}_{\mathcal{M}} \ddot{U}_{\mathrm{copy},\mathcal{M}} \varrho_2'$, and, with $\Gamma_i := \Gamma_i^{\ddot{U}_{\mathcal{M}} \varrho_2'}$, again no two elements of $\bigcup_i \Gamma_i$ lie in the same $\Delta_{m_1,\dots}$.

Since $\Delta_{m_1,\dots}$ is also closed under application of $U_{\mathrm{F},\mathcal{M}}$, the same argumentation as just given for $\tilde{U}_{\mathcal{M}}$ holds for $U_{\mathrm{F},\mathcal{M}}$, too.

After applying $P_{r_{k+2}}$, the compatibility is preserved as already explained above for $P_{r_{k+1}}$.

If $r_{k+2} = 1$, $U_{\mathrm{FS},\mathcal{M},\mathrm{cond}}$ operates as the identity, so the compatibility is trivially preserved. If $r_{k+2} = 0$, $U_{\mathrm{FS},\mathcal{M},\mathrm{cond}}$ operates as $U_{\mathrm{copy},\mathcal{M}} \cdot \tilde{U}_{\mathrm{FS},\mathcal{M}} \cdot U_{\mathrm{copy},\mathcal{M}}$. This also preserves the compatibility, with the same argumentation as given for $\tilde{U}_{\mathcal{M}} \cdot U_{\mathrm{copy},\mathcal{M}}$ (with the only difference, that $U_{\mathrm{copy},\mathcal{M}}$ is applied a second time after $\tilde{U}_{\mathrm{FS},\mathcal{M}}$).

We conclude, that in case 1 we have compatibility of $\ddot{O}_k \varrho_1' = \ddot{U}_{\mathrm{FS},\mathcal{M},\mathrm{cond}} \ddot{P}_{r_{k+2}} \ddot{U}_{\mathrm{F},\mathcal{M}} \ddot{U}_{\mathcal{M}} \varrho_1'$ and $\ddot{O}_k \varrho_2' = \ddot{U}_{\mathrm{FS},\mathcal{M},\mathrm{cond}} \ddot{P}_{r_{k+2}} \ddot{U}_{\mathrm{F},\mathcal{M}} \ddot{U}_{\mathcal{M}} \varrho_2'$.

Case 2 ($r_{k+1} = 0$, $k_{k+1} = \mathit{run}$): Here ϱ_1' and ϱ_2' have been projected with P_0 and $P^{(k+1)}_{\mathit{run}}$. Therefore O_{k+1} operates on ϱ_1' and ϱ_2' by definition as

$$\ddot{P}_{r_{k+2}} \ddot{O}_{k+1} \varrho_{1,2}' = \ddot{P}_{r_{k+2}} \ddot{U}_{\mathrm{FS},\mathcal{M},\mathrm{cond}} \ddot{U}_{\mathrm{F},\mathcal{M}} \ddot{U}_{\mathcal{M}} \ddot{X} \varrho_{1,2}'.$$

Since ι is well-formed, ϱ_1' lies in $\mathcal{H}_{Q,\mathcal{M}} \otimes |\#\rangle \otimes |\#\rangle \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^2$. To prove this, consider that all P_{r_i}, P_{f_i}, $P^i_{k_i}$, and the projector used in the well-formedness condition for quantum interactions commute.

Since ϱ_2' is compatible to ϱ_1', is does also lie in $\mathcal{H}_{Q,\mathcal{M}} \otimes |\#\rangle \otimes |\#\rangle \otimes \mathbb{C}^{\mathbb{Z}} \otimes \mathbb{C}^2$.

Because of $r_{k+1} = 0$, we further have that ϱ_1' and ϱ_2' both lie in $\mathcal{H}_{Q,\mathcal{M}} \otimes |\#\rangle \otimes |\#\rangle \otimes \mathbb{C}^{\mathbb{Z}} \otimes |0\rangle$.

The bit flip X maps $|q\rangle \otimes |\#\rangle \otimes |\#\rangle \otimes \mathbb{C}^{\mathbb{Z}} \otimes |0\rangle$ into $|q\rangle \otimes |\#\rangle \otimes |\#\rangle \otimes \mathbb{C}^{\mathbb{Z}} \otimes |1\rangle$, so the compatibility of $X\varrho_1'$ and $X\varrho_2'$ follows.

The rest of this case is proven as in case 1, since the $X\varrho_{1,2}'$ populate the same space as is done in case 1 by $\varrho_{1,2}'$, and the remaining parts of O_i are the same (i.e. $\ddot{U}_{\mathrm{FS},\mathcal{M},\mathrm{cond}}\ddot{U}_{\mathrm{F},\mathcal{M}}\ddot{U}_{\mathcal{M}}$).

Case 3 ($r_{k+1} = 0$, $k_{k+1} = \textit{send}$): Here O_i operates as $U_{\mathrm{copy},\mathcal{M}}\tilde{U}_{\mathrm{S},\mathcal{M}}U_{\mathrm{copy},\mathcal{M}}$. The compatibility is preserved, the proof is analogous to that for the preservation under application of $\tilde{U}_{\mathcal{M}}U_{\mathrm{copy},\mathcal{M}}$ above.

Case 4 ($r_{k+1} = 0$, $k_{k+1} = \textit{rec}$): Here O_i operates as $U_{\mathrm{copy},\mathcal{M}}\tilde{U}_{\mathrm{R},\mathcal{M}}U_{\mathrm{copy},\mathcal{M}}$. By proceeding analogously to $\tilde{U}_{\mathcal{M}}U_{\mathrm{copy},\mathcal{M}}$, we prove compatibility.

Case 5 ($r_{k+1} = 0$, $k_{k+1} = \textit{world}$): Here $\varrho_{1,2}'$ lie in $\mathcal{H}_{Q,\mathcal{M}} \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}} \otimes |0\rangle$, and O_i operates only on $\mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}}$. Thus each $\Gamma_i^{\varrho_2'}$ is of the form $|q\rangle \otimes \mathcal{H}_F \otimes \mathcal{H}_C \otimes \mathbb{C}^{\mathbb{Z}} \otimes |0\rangle$ and mapped onto itself. This preserves the compatibility and finishes this case.

We have now proven, that in all five cases we have compatible $\ddot{O}_{k+1}\ddot{P}_{r_{k+1}}\ddot{P}_{f_{k+1}}\ddot{P}^{(k+1)}_{k_{k+1}}\varrho_{1,2}^{(k)}$. A further application of $P_{r_{k+2}}$ does not destroy this compatibility as already described above. So

$$\varrho_2'' := \ddot{P}_{r_{k+2}}O_{k+1}\ddot{P}_{r_{k+1}}\ddot{P}_{f_{k+1}}\ddot{P}^{(k+1)}_{k_{k+1}}\varrho_{1,2}^{(k)}$$

is compatible to $\varrho_1^{(k+1)}$.

To finish the induction step, we still need to prove that $\ddot{M}_{k+2}\varrho_2''$ is compatible to $\varrho_1^{(k+1)}$.

Each eigenspace E_j of $M := M_{k+2}$ can be written as a direct sum $\bigcup_{\mathcal{H}\in\Gamma_j'\subseteq\Gamma}\mathcal{H}$ of elements of Γ. M-measuring $\varrho_2'' = \sum_i \varrho(|\Psi_i^{\varrho_2''}\rangle)$ thus yields

$$\varrho_2^{(k+1)} = \sum_{i,j}\varrho(P_j|\Psi_i^{\varrho_2''}\rangle) =: \sum_{i,j}\varrho(|\Psi_{i,j}\rangle)$$

where P_j is the projection on E_j. Because of $\Psi_{i,j} \in \bigcup_{\mathcal{H}\in\Gamma_i\cap\Gamma_j'}\mathcal{H}$ and $\sum_{i,j}|\Psi_{i,j}\rangle = \sum_i|\Psi_i^{\varrho_2''}\rangle = |\Psi^{\varrho_1^{(k+1)}}\rangle$, we have shown that $\varrho_2^{(k+1)}$ is compatible to $\varrho_1^{(k+1)}$, since the $\Gamma_i \cap \Gamma_j$ are pairwise disjunct.

So the induction step is completed, and we have proven, that $\varrho_2^{(k)}$ is compatible to $\varrho_1^{(k)}$ for $k = 1, \ldots, n$.

Using techniques analogue to those above, we see, that $\varrho_{1,2}^* := \ddot{P}_{r_{n+1}}\ddot{P}_{f_{n+1}}\ddot{P}^{(n+1)}_{k_{n+1}}\varrho_{1,2}^{(n)}$ are compatible. Since all P_{r_i}, P_{f_i}, $P^{(i)}_{k_i}$ commute, it is

$$\varrho_1^* := \ddot{P}_{r_{n+1}}\ddot{P}_{f_{n+1}}\ddot{P}^{(n+1)}_{k_{n+1}}\prod_{i=1}^{n}(\ddot{O}_i\ddot{P}_{r_i}\ddot{P}_{f_i}\ddot{P}^{(i)}_{k_i})\,\varrho(|S\rangle),$$

$$\varrho_2^* := \ddot{P}_{r_{n+1}}\ddot{P}_{f_{n+1}}\ddot{P}^{(n+1)}_{k_{n+1}}\ddot{M}_{n+1}\prod_{i=1}^{n}(\ddot{O}_i\ddot{P}_{r_i}\ddot{P}_{f_i}\ddot{P}^{(i)}_{k_i}\ddot{M}_i)\,\varrho(|S\rangle),$$

Note that this is almost the definition of ϱ_1 resp. ϱ_2, we have only replaced the M_r-, M_f- and $M^{(i)}$-measurements by projections on a sequence of their eigenspaces. Therefore summing $\varrho_{1,2}^*$ over all possible sequences of $r_i \in \{0,1\}$, $f_i \in \Sigma^*$, $k_i \in \{\textit{run}, \textit{send}, \textit{rec}, \textit{world}\}$, we get $\varrho_{1,2}$. So if we can prove, that M-measuring of $\varrho_{1,2}^*$ gives the same generalised probability distributions, we know that this also holds for $\varrho_{1,2}$, since the generalised probability distribution of a measurement of a sum of generalised density matrices is the sum of the generalised probability distributions of the measurements of the generalised density matrices.

So we will now prove that M-measuring ϱ_1^* and ϱ_2^* gives the same generalised probability distribution and therewith finish this proof.

Let E be some eigenspace of M and P_E the projector onto E. It follows from the definition of M, that $E = \bigoplus_{\mathcal{H}\in\Gamma_E}\mathcal{H}$ for some $\Gamma_E \subseteq \Gamma$. Therefore $|\Psi_i^P\rangle := P_E|\Psi_i^{\varrho_2^*}\rangle \in \bigoplus_{\mathcal{H}\in\Gamma_E\cap\Gamma_i}\mathcal{H}$. So all $|\Psi_i^P\rangle$ are orthogonal and $\sum_i|\Psi_i^P\rangle = |\Psi^P\rangle := P_E|\Psi^{\varrho_1^*}\rangle$. If $P_{1,E}$ and $P_{2,E}$ are the generalised probabilities of measuring E on ϱ_1^* resp. ϱ_2^*, we get

$$P_{1,E} = \langle\Psi^P|\Psi^P\rangle = \sum_{i,j}\langle\Psi_i^P|\Psi_j^P\rangle = \sum_i\langle\Psi_i^P|\Psi_i^P\rangle = P_{2,E}.$$

We have shown the last pending statement, i.e. that M-measuring ϱ_1^* and ϱ_2^* yields the same generalised probability distribution, so our proof is complete. ■

A.2 Stochastic vs. hard relative polynomial running time (section 3.3.4)

In section 3.3.4 we have claimed, that if $\mathcal{M}_2$ is of stochastic polynomial running time relative to $\mathcal{M}_1$, then it is also of hard polynomial running time relative to $\mathcal{M}_1$. Of this we will now give the proof:

Assume, that $\mathcal{M}_2$ is of stochastic polynomial running time relative to $\mathcal{M}_1$, therefore there is a polynomial p, such that

$$P(T_2 \geq t) \leq P(p(T_1 + k + n) \geq t).$$

for all $k \geq 0$, $n \geq 1$, $t \geq 0$, ι_1, ι_2, where X_i is the execution transcripts of $\mathcal{M}_i$ on ι_i, and $T_i := T^{\mathrm{X}_i}_{k,n}$.

W.l.o.g. we can assume, that p has coefficients in $\mathbb{N}_0$ and is strictly monotonously increasing. Therefore we get

$$P(T_2 > t) = P(T_2 \geq t+1) \leq P(p(T_1 + k + n) \geq t+1) = P(p(T_1 + k + n) > t).$$

Let $t_i := T_{i,n,k}$ (as in the definition of hard relative polynomial running time), and $t'_2 := p(t_1 + n + k)$. We then have

$$P(T_2 > t'_2) = P(T_2 > p(t_1 + n + k)) \geq P(p(T_1 + n + k) > p(t_1 + n + k)) = P(T_1 > t_1) \stackrel{\text{def}}{=} 0.$$

Since t_2 is minimal with $P(T_2 > t_2) = 0$, we have $t_2 \leq t'_2 = p(t_1 + n + k)$, so $\mathcal{M}_2$ is of hard polynomial running time relative to $\mathcal{M}_1$. ■

A.3 Strict polynomial running time

The following definition is equivalent to the original definition of strict polynomial running time given in definition 3.6:

Definition A.2: Strict polynomial running time (variant 1)
Let $E_{k,n,t}$ be as in the definition of running time (definition 3.5).

We say an IAQS $\mathcal{M}$ has *(strict) polynomial running time* if there is a polynomial p such that for any quantum interaction ι it is

$$P(\mathrm{X} \in K_k \setminus E_{k,n,p(k+n)-1}) = 0$$

for all $n \in \mathbb{N}$, $k \in \mathbb{N}_0$ with X being the execution transcript on ι. □

Proof: Since $T^{\mathrm{X}}_{k,n} = \perp$ has probability 1 or 0 (by definition of $T^{\mathrm{X}}_{k,n}$), it is

$$\begin{aligned}
& P(T^{\mathrm{X}}_{k,n} < p(k+n) \text{ or } T^{\mathrm{X}}_{k,n} = \perp) = 1 \\
\iff\quad & P(T^{\mathrm{X}}_{k,n} \geq p(k+n), T^{\mathrm{X}}_{k,n} \neq \perp) = 0 \text{ or } P(T^{\mathrm{X}}_{k,n} = \perp) = 1 \\
\stackrel{\text{def } T^{\mathrm{X}}_{k,n}}{\iff}\quad & P(\mathrm{X} \in \bigcup_{t \in \mathbb{N} \cup \{\infty\}} E_{k,n,t} \setminus E_{k,n,p(k+n)-1}) = 0 \qquad \text{or} \\
& P(\mathrm{X} \in \bigcup_{t \in \mathbb{N} \cup \{\infty\}} E_{k,n,t}) = 0 \\
\iff\quad & P(\mathrm{X} \in \bigcup_{t \in \mathbb{N} \cup \{\infty\}} E_{k,n,t} \setminus E_{k,n,p(k+n)-1}) = 0.
\end{aligned}$$

Further it is

$$\bigcup_{t \in \mathbb{N} \cup \{\infty\}} E_{k,n,t} = E_{k,n,\infty} = K_k$$

as follows from the definition of $E_{k,n,t}$ and K_k (the latter being as in definition 3.5). ■

If we replace vanishing probability by impossibility, we get the equivalent

Definition A.3: Strict polynomial running time (variant 2)
Let $E_{k,n,t}$ be as in the definition of running time (definition 3.5).

We say an IAQS $\mathcal{M}$ has *(strict) polynomial running time* if there is a polynomial p such that for any quantum interaction ι the set

$$K_k \setminus E_{k,n,p(k+n)-1}$$

is impossible for all $n \in \mathbb{N}$, $k \in \mathbb{N}_0$ with X being the execution transcript on ι. □

Proof: It is sufficient to show, that $M_{k,n} := K_k \setminus E_{k,n,p(k+n)-1}$ is impossible if and only if is has probability 0.

If $M_{k,n}$ is impossible, it also has probability 0, as shown in section 3.2.

Let $M_{k,n}$ be of probability 0, and $(e_i) \in M_{k,n}$. Then there is a sequence of consecutive elements $(r_i)_{i=1}^m$ having the properties as described in the definition of $E_{k,n,t}$ (definition 3.5) and $m \geq p(k+n)$. Let $\tilde{m}$ be such, that $r_1, \dots, r_{p(k+n)}$ is contained in $e_1, \dots, e_{\tilde{m}}$. Then any sequence beginning with $e_1, \dots, e_{\tilde{m}}$ is *not* in $E_{k,n,p(k+n)-1}$. Further any sequence beginning with $e_1, \dots, e_{\tilde{m}}$ is in K_k, because the value of k is determined by the elements of the transcript preceding the first element formed $(\mathit{run}, \#)$ or $(\mathit{stop}, \cdot)$, and r_1 has that form.

So $P(\mathrm{X}_i = e_i, i = 1, \dots, \tilde{m}) \leq P(\mathrm{X} \in M_{k,n}) = 0$, therefore (e_i) is impossible.

Since (e_i) was an arbitrary chosen element of $M_{k,n}$, that set is impossible, too. ■

A.4 Time evolution of a quantum network

Definition A.4: Time evolution of a quantum network
Let $\mathcal{N}$ be a quantum network, then the *time evolution* $U_{\mathcal{N}}$ is constructed as follows:

Let for $G \in \Sigma^*$ be $\text{ꓘK}(a_1, \dots, a_n) = G$, $\text{ꓘK}(\texttt{sched}, \cdot, \mathit{token}, \cdot)$ be the last element of (a_i) having the form $\text{ꓘK}(\texttt{sched}, \cdot, \cdot, \cdot)$. Define then $\mathit{token}(G) := \mathit{token}$, if $\mathit{token} \in I_{\mathcal{N}}$, and set $\mathit{token}(G) := \bot$ if $\mathit{token} \notin I_{\mathcal{N}}$ or if there is no element of the required form.

Let $P_t^{(1)}$ $(t \in I_{\mathcal{N}} \cup \{\bot\})$ be the projection of $\mathcal{H}_{G,\mathcal{N}}$ onto

$$\operatorname{span}\{|G\rangle : \mathit{token}(G) = t\}.$$

Let then

$$\begin{aligned} U^{(2)} &:= \sum_{i \in I_{\mathcal{N}}} (U_{\mathcal{N}_i} P_i^{(1)}) + P_\bot^{(1)} \\ P_1^{(3)} &:= \sum_{i \in I_{\mathcal{N}}} (P_{\mathrm{F},\mathcal{N}_i} P_i^{(1)}) \\ P_0^{(3)} &:= \sum_{i \in I_{\mathcal{N}}} (P^{\complement}_{\mathrm{F},\mathcal{N}_i} P_i^{(1)}). \end{aligned}$$

Let the linear operators $U_{F_i=f}$ $(f \in \{0,1\})$ operate on $\mathcal{H}_{G,\mathcal{N}} \otimes \mathcal{H}_{H,\mathcal{N}_i}$ as

$$U_{F_i=f}|G\rangle \otimes |h\rangle := |\text{CQ}(G, \text{ꓘK}(\texttt{endconf}, f))\rangle \otimes |\text{CQ}(h, \text{ꓘK}(\texttt{endconf}, f))\rangle \qquad (G, h \in \Sigma^*)$$

and $P_f^{(3')}$ $(f = 0, 1)$ the projection of $\mathcal{H}_{G,\mathcal{N}}$ onto

$$\operatorname{span}\{|G\rangle : G \in \Sigma^*, f(G) = f\}$$

where $f(G) := 1$ if $\text{ꓘK}^{-1}(G)$ exists and the last element of $\text{ꓘK}^{-1}(G)$ having the form $\text{ꓘK}(\texttt{endconf}, \cdot)$ exists and is $\text{ꓘK}(\texttt{endconf}, 1)$, otherwise let $f(G) = 0$.

Let further $U_{\mathit{frame},i}$ $(i \in I_{\mathcal{N}})$ operate on $\mathcal{H}_{G,\mathcal{N}} \otimes \mathcal{H}_F \otimes \mathcal{H}_{H,\mathcal{N}_i}$ as

$$U_{\mathit{frame},i}|G\rangle \otimes |f\rangle \otimes |h_i\rangle := |\text{CQ}(G, \text{ꓘK}(\texttt{frame}, f))\rangle \otimes |f\rangle \otimes |\text{CQ}(h_i, \text{ꓘK}(\texttt{frame}, f))\rangle \qquad (G, f, h_i \in \Sigma^*),$$

and $U_{\mathit{frame}=f}$ $(f \in \Sigma^*)$ operate on $\mathcal{H}_{G,\mathcal{N}} \otimes \mathcal{H}_F$ as

$$U_{\mathit{frame}=f}|G\rangle \otimes |\tilde{f}\rangle := |G\rangle \otimes |\tilde{f} \ominus \mathit{frame}(G) \oplus f\rangle \qquad (G, f, \tilde{f} \in \Sigma^*),$$

where $\mathit{frame}(G)$ is defined such, that $\text{ꓘK}(\texttt{frame}, \mathit{frame}(G))$ is the last element of $\text{ꓘK}^{-1}(G)$ having the form $\text{ꓘK}(\texttt{frame}, \cdot)$, if $\text{ꓘK}^{-1}(G)$ exists, and there is an element thus formed. Let $\mathit{frame}(G) := \#$ otherwise.

Let $S := E^* \times I_{\mathcal{N}} \times \Sigma^*$ (the image range of $\mathrm{Sched}_{\mathcal{N}}$) and the operators $U_{\mathit{sched},s}$ $(s \in S)$ and U_{nosched} operate on $\mathcal{H}_{G,\mathcal{N}}$ as follows:

$$\begin{aligned} U_{\mathit{sched},(E,t,d)}|G\rangle &:= |\text{CQ}(G, \text{ꓘK}(\texttt{sched}, E', t, d))\rangle \\ U_{\mathit{nosched}}|G\rangle &:= |\text{CQ}(G, \texttt{nosched})\rangle \end{aligned}$$

Here E' is the textification of E, defined via $(e_1, \dots, e_n) := E$, $(t_i, c_{i1}, \dots, c_{im}) := e_i$, $t'_i :=$ `transmit`, `corrupt` or `control`, depending on t_i, $e'_i := (t'_i, c_{i1}, \dots, c_{im})$ and $E' := \text{ꓘK}(e'_1, \dots, e'_n)$.

Then let

$$
\begin{aligned}
U^{(4)} &:= \sum_{i\in I_{\mathcal{N}}} (U_{F_i=1} P_1^{(3)} P_i^{(1)} + U_{F_i=0} P_0^{(3)} P_i^{(1)}) + P_\perp^{(1)},\\
U^{(6)} &:= \sum_{i\in I_{\mathcal{N}}} (U_{\mathrm{FS},\mathcal{N}_i} P_1^{(3')} P_i^{(1)} + P_0^{(3')} P_i^{(1)}) + P_\perp^{(1)},\\
U^{(7)} &:= \sum_{i\in I_{\mathcal{N}}} (U_{\mathit{frame},i} P_1^{(3')} P_i^{(1)} + P_0^{(3')} P_i^{(1)}) + P_\perp^{(1)},\\
U^{(8)} &:= \sum_{i\in I_{\mathcal{N}}} (U_{\mathit{frame}=\#} P_1^{(3')} P_i^{(1)} + P_0^{(3')} P_i^{(1)}) + P_\perp^{(1)}.
\end{aligned}
$$

Let $\tilde{P}_s^{(9)}$ $(s \in S)$ be the projection of $\mathcal{H}_{G,\mathcal{N}}$ onto

$$\operatorname{span}\{|G\rangle : \operatorname{Sched}_{\mathcal{N}}(G) = s\}$$

and

$$
\begin{aligned}
P_s^{(9)} &:= \sum_{i\in I_{\mathcal{N}}} (\tilde{P}_s^{(9)} P_1^{(3')} P_i^{(1)} + P_0^{(3')} P_i^{(1)}) + \tilde{P}_s^{(9)} P_\perp^{(1)} \qquad (s \in S),\\
U^{(10)} &:= \sum_{i\in I_{\mathcal{N}}} (\sum_{s\in S} (U_{\mathit{sched},s} P_s^{(9)} P_1^{(3')} P_i^{(1)}) + U_{\mathit{nosched}} P_0^{(3')} P_i^{(1)}) + \sum_{s\in S} U_{\mathit{sched},s} P_s^{(9)} P_\perp^{(1)}.
\end{aligned}
$$

Let *sched*(G) be the last element of $Ж^{-1}$ having the form $Ж(\texttt{sched}, \cdot)$ or $\texttt{nosched}$. Let *sched*$(G) := \perp$, if there is no such element. Then $P_E^{(11)}$ $(E \in E^*)$ projects $\mathcal{H}_{G,\mathcal{N}}$ onto

$$\operatorname{span}\{|G\rangle : G, t, d \in \Sigma^*, \mathit{sched}(G) = Ж(\texttt{sched}, E', t, d)\},$$

where E' is the textification of E, as described above.

The operator $P_\perp^{(11)}$ projects onto

$$\operatorname{span}\{|G\rangle : \mathit{sched}(G) = \texttt{nosched} \text{ or } \mathit{sched}(G) = \perp\}.$$

Then it is $U_E^{(11)} := \prod_{\nu=1}^n U_{e_\nu}^{ev}$ with $(e_1, \ldots, e_n) := E$ for an event list $E \in E^*$ and

$$
\begin{aligned}
U^{(11)} &:= \sum_{E\in E^*} (U_E^{(11)} P_E^{(11)}) + P_\perp^{(11)},\\
U^{ev}_{(\mathit{transmit},s,d,f)} &:= U_{\mathrm{R},\mathcal{N}_d} U'_{\mathit{frame}=f} U_{\mathrm{S},\mathcal{N}_s},\\
U^{ev}_{(\mathit{control},d,f)} &:= U_{\mathrm{R},\mathcal{N}_d} U'_{\mathit{frame}=f},\\
U^{ev}_{(\mathit{corrupt},c,t,f)} &:= U_{\mathrm{R},\mathcal{N}_c} U_{\mathrm{corr},\mathcal{N}_t} U'_{\mathit{frame}=f,\mathit{hist}_i}
\end{aligned}
$$

Here $U'_{\mathit{frame}=f,\mathit{hist}_i}$ operates on $\mathcal{H}_F \otimes \mathcal{H}_{H,\mathcal{N}_i}$ as

$$U'_{\mathit{frame}=f,\mathit{hist}_i} |\tilde{f}\rangle \otimes |h\rangle := |\tilde{f} \oplus Ж(f, h)\rangle \qquad (f, \tilde{f}, h \in \Sigma^*)$$

and $U'_{\mathit{frame}=i}$ operates on $\mathcal{H}_F$ as

$$U'_{\mathit{frame}=i} |\tilde{f}\rangle := |\tilde{f} \oplus f\rangle \qquad (f, \tilde{f} \in \Sigma^*).$$

We finally get the time evolution $U_{\mathcal{N}}$ as

$$U_{\mathcal{N}} := U^{(11)} U^{(10)} U^{(8)} U^{(7)} U^{(6)} U^{(4)} U^{(2)}. \qquad \square$$

We will now compare this formal definition with the algorithm given in section 4.2 and show, that both do the same.

Step 1 does not modify the state of the network, therefore there is no corresponding unitary operator $U^{(1)}$. It is formulated as setting i to some value, this can also be interpreted as a branch, where the program takes another path for each possible value of i, each of those paths executing its program as if the variable i would hold that given value. To implement such branches, we need to have a projection onto those states, which lead into a given path. Therefore we have constructed $P_i^{(1)}$, which goes into the branch conditioned by "the last datum formed ... had *token* = i", and for the branch "there is no such datum", we have $P_\perp^{(1)}$. Now each step whose behaviour depends on this branch, has to be of the form $\sum_{i\in I_\mathcal{N}}(U_iP_i^{(1)}) + U_\perp P_\perp^{(1)}$, where U_i, $U_\perp$ are the actions to be taken in dependence of the respective outcomes of the branch.

Step 2 is to evaluate $U_{\mathrm{S},\mathcal{N}_i}$. Since $U_{\mathrm{S},\mathcal{N}_i}$ may be different for each i, we have to split the execution using $P_i^{(1)}$. If $P_\perp^{(1)}$ applies, we do nothing (i.e. we apply the identity), since the instruction in 1 was to skip steps 2–9.

In step 3 we again have no real action, but only a branch depending on the result of applying the end configuration measurement $P_{\mathrm{F},\mathcal{N}_i}$ on $\mathcal{N}_i$. Again this is formulated in the algorithm as an assignment to a variable, and again we reformulate it as a branch between paths "$F_i = 0$" and "$F_i = 1$". And this measurement we also have to split depending on the branch done in step 1.

In step 4 we execute $U_{F_i=f}$ for each path in which we have $F_i = f$ $(f = 0, 1)$, and do nothing if $P_\perp^{(1)}$ applies, i.e. if we are in the path skipping steps 2–9. The operator $U_{F_i=f}$ just appends the information Ж(`endconf`, f) to $|G\rangle$ and $|\mathrm{hist}_i\rangle$.

For step 5 we have not constructed any operators, since it is only a branch depending on the value of F_i, and that branch is already realised by $P_k^{(3)}$. We only have to pay attention, that all further steps do nothing if $F_i = 0$. In step 5 there does also appear an instruction "append `nosched` to $|G\rangle$", but this instruction we will only execute in step 10, we may choose to do such, since in the intermediate steps there is no action on that execution path. And we choose to do so, since this makes the proof, that the time evolution is unitary, much easier, since it ensures that every step by itself is already unitary.

For step 6 we have constructed $U^{(6)}$ quite analogous to $U^{(2)}$, except that it does nothing, if the machine has not terminated ($F_i = 0$).

There is an important point: Since we now change the machines configuration by application of $U_{\mathrm{FS},\mathcal{N}_i}$, and in particular whether the machine is in an end configuration, we may no longer use $P^{(3)}$ to get the value of the variable F_i, since $P^{(3)}$ uses $P_{\mathrm{F},\mathcal{N}_i}$ to decide depending on the machine's state and may thus yield other results now. Therefore we define the operator $P^{(3')}$ to be used in place of $P^{(3)}$ from now on.[40] This projection finds out, which is the value of F_i by looking at $|G\rangle$, where we have stored Ж(`endconf`, F_i) in step 4.

In step 7 we have $U^{(7)}$, which calls $U_{frame,i}$ depending on the value of i, and only if we are not in a path which skips this step. $U_{frame,i}$ just appends the content of $|\mathrm{frame}\rangle$ to $|G\rangle$ and $|\mathrm{hist}_i\rangle$.

In step 8 we set $|\mathrm{frame}\rangle$ to $|\#\rangle$, provided that we do not skip this step. To achieve this, we define an operator $U_{frame=f}$, which sets $|\mathrm{frame}\rangle$ to f by using the $\oplus$- and the $\ominus$-operations: Since *frame*(G) holds, due to step 7, the same value as $|\mathrm{frame}\rangle$, for $|\mathrm{frame}\rangle = |\tilde f\rangle$ it is $\tilde f \ominus \mathit{frame}(G) \oplus f = f$, so we can in fact write f onto $|\mathrm{frame}\rangle$. With the special case of $f = \#$ we can thus construct $U^{(8)}$.

We now come to step 9. Note that here the branch begun in step 1 merges again, therefore we will execute this step for the case that $F_i = 1$ and for the case that finding the token in step 1 failed (projector $P_\perp^{(1)}$). As in step 1, this variable assignment is realised by a branch over all possible values for that variable (called s resp. (***events*, *token*, *data***)), so we only define a projector $P_s^{(9)}$ to be used in the following steps.

Then step 10 writes the value of s, i.e. of the output of $\mathrm{Sched}_\mathcal{N}$ calculated in the preceding step, onto $|G\rangle$ using the according syntax, except when we are in the path with $F_i = 0$, where we write *nosched* (this was delayed from step 5). This is done using the operators $U_{sched,s}$ and $U_{nosched}$ respectively.

For step 11 we have to split in a path for each possible event list returned by $\mathrm{Sched}_\mathcal{N}$, and in a path for the case, that we have not executed $\mathrm{Sched}_\mathcal{N}$ at all, due to previous branches. We cannot, however, use $P_t^{(1)}$ any more to decide in which branch we are, since we have modified the content of $|G\rangle$ in a way which changes the output of *token*(G). Therefore we define the operator $P_\perp^{(11)}$, which checks whether we have written `nosched` in step 10. In this case we abstain from any actions.

Otherwise we split depending of the data stored on $|G\rangle$ into a path for each event list output by $\mathrm{Sched}_\mathcal{N}$. We cannot use $P^{(9)}$ for this, since first this operator uses $P^{(1)}$ and second we have modified the content of $|G\rangle$, i.e. the input to $\mathrm{Sched}_\mathcal{N}$. Therefore we define $P_E^{(11)}$, which projects onto those paths, where we have got the event list E, depending on what was stored on $|G\rangle$.

In each path we execute $U_E^{(11)}$, which again executes $U_{e_\nu}^{ev}$ for each event e_ν in E. Depending of the type of e_ν, we have three definitions for $U_{e_\nu}^{ev}$, which should be straightforward, except for the operators $U'_{frame=f}$ and $U'_{frame=f,hist_i}$ still to be explained: They write f resp. the concatenation of f and $|\mathrm{hist}_i\rangle$ onto $|\mathrm{frame}\rangle$. This can be done, because $|\mathrm{frame}\rangle$ has the content $|\#\rangle$.

[40] We already use this new projection in this step to ensure that $U^{(6)}$ is unitary.

We get the final time evolution operator $U_{\mathcal{N}}$ by applying all the unitary evolutions for the single steps.

We will now proceed and prove, that $U_{\mathcal{N}}$ is unitary. In order to do this, we will prove it for each of the factors $U^{(2)}$, $U^{(4)}$, $U^{(6)}$, $U^{(7)}$, $U^{(8)}$, $U^{(10)}$, and $U^{(11)}$.

Any operator of the form $U = \sum_i U_i P_i$ is unitary, if the U_i are unitary, and if the P_i are pairwise commuting projections with $\sum_i P_i = \mathbb{1}$ and $P_i P_j = 0$ $(i \neq j)$, which do also commute with the U_i, since

$$U^\dagger U = \sum_{i,j} P_i^\dagger U_i^\dagger U_j P_j = \sum_{i,j} U_i^\dagger P_i^\dagger P_j U_j = \sum_i U_i^\dagger P_i U_i = \sum_i P_i = \mathbb{1}.$$

This condition is satisfied by $U^{(2)}$:[41] The $P_t^{(1)}$ $(t \in I_{\mathcal{N}} \cup \{\perp\})$ satisfy the pairwise commutation, since they project onto pairwise orthogonal subspaces by construction, and the $U_{\mathcal{N}_i}$ commute with the $P_t^{(1)}$, since the $U_{\mathcal{N}_i}$ operate only on $\mathcal{H}_{Q,\mathcal{M}}$, while the $P_t^{(1)}$ operate on $\mathcal{H}_{Q,\mathcal{N}_i}$. So $U^{(2)}$ is unitary.

We will now examine $U^{(4)}$. It is

$$\sum_{i \in I_{\mathcal{N}}} (P_1^{(3)} P_i^{(1)} + P_0^{(3)} P_i^{(1)}) + P_\perp^{(1)} = \underbrace{(P_1^{(3)} + P_0^{(3)})}_{=\sum_{i \in I_{\mathcal{N}}} P_i^{(1)}} \sum_{i \in I_{\mathcal{N}}} P_i^{(1)} + P_\perp^{(1)} = \mathbb{1}.$$

It commutes $P_{\mathrm{F},\mathcal{N}_i}$ with $P_t^{(1)}$, since they operate on $\mathcal{H}_{Q,\mathcal{N}_i}$ resp. $\mathcal{H}_{G,\mathcal{N}}$. The commutator of $P_{\mathrm{F},\mathcal{N}_i}$ and $P_{\mathrm{F},\mathcal{N}_j}$ vanishes, too, because they operate on the different tensor factors $\mathcal{H}_{Q,\mathcal{N}_i}$ and $\mathcal{H}_{Q,\mathcal{N}_j}$ if they are different. The $P_t^{(1)}$ commute, as already mentioned above.

Since the $P_k^{(3)}$ are in the ring generated by the $P_i^{(1)}$ and the $P_{\mathrm{F},\mathcal{N}_i}$, they do also commute with the $P_t^{(1)}$. So the pairwise commutation is given in the case of $U^{(4)}$.

It commutes $U_{F_i=f}$ with $P_{\mathrm{F},\mathcal{N}_i}$, since they operate on different tensor factors $\mathcal{H}_{G,\mathcal{N}} \otimes \mathcal{H}_{H,\mathcal{N}_i}$ and $\mathcal{H}_{Q,\mathcal{N}_i}$. Further does $U_{F_i=f}$ also commute with $P_t^{(1)}$, due to the following: It is $\texttt{token}(G) = t$ if and only if $\texttt{token}(\mathbb{Q}(G, \mathbb{K}(\texttt{endconf}, f))) = t$, so for any basis state $|G\rangle \otimes |h\rangle \in \mathcal{H}_{G,\mathcal{N}} \otimes \mathcal{H}_{H,\mathcal{N}_i}$ it is

$$\begin{aligned} U_{F_i=f} P_t^{(1)} |G\rangle \otimes |h\rangle &= U_{F_i=f} \delta |G\rangle \otimes |h\rangle \\ &= \delta |\mathbb{Q}(G, \mathbb{K}(\texttt{endconf}, f))\rangle \otimes |\mathbb{Q}(h, \mathbb{K}(\texttt{endconf}, f))\rangle = P_t^{(1)} U_{F_i=f} |G\rangle \otimes |h\rangle, \end{aligned}$$

where we set $\delta := 1$ if $\texttt{token}(G) = t$, $\delta := 0$ otherwise.

So the $U_{F_i=f}$ commute with all $P_t^{(1)}$ and all $P_k^{(3)}$, therefore $U^{(4)}$ is unitary.

The $P_k^{(3')}$ and the $P_t^{(1)}$ all commute pairwise, since they are all projections onto spaces spanned by basis vectors. They do also commute with $U_{\mathrm{FS},\mathcal{N}_i}$, since the latter operates on $\mathcal{H}_{Q,\mathcal{N}_i} \otimes \mathcal{H}_F$, while the projections operate on $\mathcal{H}_{G,\mathcal{N}}$.

Further it is $\sum_{i \in I_{\mathcal{N}}} (P_1^{(3')} P_i^{(1)} + P_0^{(3')} P_i^{(1)}) + P_\perp^{(1)} = \mathbb{1}$, so $U^{(6)}$ is unitary.

For $U^{(7)}$, the projections commute and sum up to $\mathbb{1}$ as already shown above. And that $U_{frame,i}$ commutes with those projections is shown analogous to the commutation of $U_{F_i=f}$ and $P_t^{(1)}$. So $U^{(7)}$ is unitary.

Since the $P_k^{(3')}$ and the $P_t^{(1)}$ are all projections onto basis vectors and operate only on $\mathcal{H}_{G,\mathcal{N}}$, while $U_{frame=\#}$ is read-only on $\mathcal{H}_{G,\mathcal{N}}$, they do all commute, so $U^{(8)}$ is unitary.

To prove, that $U^{(10)}$ is unitary, we cannot use the above recipe, since $U_{sched,s}$ and $U_{nosched}$ do not commute with $P_k^{(1)}$ nor with $P_s^{(9)}$.[42] So we will first introduce some projections $P_t^{(1')}$ $(t \in I_{\mathcal{N}} \cup \{\perp\})$ and $P_s^{(9')}$ $(s \in S)$, that satisfy

$$UP_t^{(1)} = P_t^{(1')} U \qquad \text{and} \qquad UP_s^{(9)} = P_s^{(9')} U.$$

for $U \in \{U_{sched,s}, U_{nosched}\}$.

Projections with these properties are given by

$$\begin{aligned} \operatorname{im} P_t^{(1')} &= \operatorname{span}\{|\mathbb{Q}(G,h)\rangle : G, h \in \Sigma^*, \mathit{token}(G) = t\} && (t \in I_{\mathcal{N}} \cup \{\perp\}), \\ \operatorname{im} P_s^{(9')} &= \operatorname{span}\{|\mathbb{Q}(G,h)\rangle : G, h \in \Sigma^*, \mathrm{Sched}_{\mathcal{N}}(G) = s\} && (s \in S). \end{aligned}$$

[41] Note, that the U_i corresponding to $P_\perp^{(1)}$ is the identity.

[42] In fact, $U_{nosched}$ commutes with $P_k^{(1)}$, but this is of no matter for our proof.

Since $P_i^{(1)}$, $P_i^{(1')}$, $P^{(3')}$, $P^{(9)}$ and $P^{(9')}$ are all projections onto spans of basis vectors, they do all commute. Because of this, each two different projections out of $P_i^{(1')}P_1^{(3')}P_s^{(9')}$, $P_i^{(1')}P_0^{(3')}$, $P_\perp^{(1')}P_s^{(9')}$ have vanishing product.

It is

$$\sum_{s\in S} P_s^{(9)}P_1^{(3')}P_i^{(1)} = \underbrace{\sum_{s\in S}\tilde{P}_s^{(9)}}_{=\mathbb{1}} P_1^{(3')}P_i^{(1)} = P_1^{(3')}P_i^{(1)}$$

and similarly $\sum_{s\in S} P_s^{(9)}P_\perp^{(1)} = P_\perp^{(1)}$, so

$$\sum_{i\in I_\mathcal{N}}(\sum_{s\in S}(P_s^{(9)}P_1^{(3')}P_i^{(1)}) + P_0^{(3')}P_i^{(1)}) + \sum_{s\in S}P_s^{(9)}P_\perp^{(1)} = \sum_{i\in I_\mathcal{N}}(P_1^{(3')}P_i^{(1)} + P_0^{(3')}P_i^{(1)}) + P_\perp^{(1)} = \mathbb{1}.$$

We can now write $U := U^{(10)}$ as $\sum_\nu U_\nu P_\nu$ with unitary U_ν and with $U_\nu P_\nu = P'_\nu U_\nu$, where $P'_\nu P'_\mu = 0$ for $\mu \neq \nu$, and where $\sum_\nu P_\nu = \mathbb{1}$. Therefore it is

$$U^\dagger U = \sum_{\mu,\nu} P_\mu^\dagger U_\mu^\dagger U_\nu P_\nu = \sum_{\mu,\nu} U_\mu^\dagger P_\mu'^\dagger P'_\nu U_\nu = \sum_\nu U_\nu^\dagger P_\nu'^\dagger P'_\nu U_\nu = \sum_\nu P_\nu^\dagger U_\nu^\dagger U_\nu P_\nu = \sum_\nu P_\nu = \mathbb{1},$$

so $U^{(10)}$ is unitary.

For $U^{(11)}$ we can again use our recipe, since the $P_E^{(11)}$, $P_\perp^{(11)}$ sum up to $\mathbb{1}$, multiply mutually to 0, commute with each other and with $U^{(11)}$, since the latter does not operate on $\mathcal{H}_{G,\mathcal{N}}$, while the projections operate only on $\mathcal{H}_{G,\mathcal{N}}$. So $U^{(11)}$ is unitary.

Since all factors of $U_\mathcal{N}$ are unitary, $U_\mathcal{N}$ itself is unitary, too. ∎

A.5 $\ddot{M}_G\ddot{U}\ddot{M}_G = \ddot{M}_G\ddot{U}$ (section 5.5)

In section 5.5 we have stated the following:

Let $U \in U(\mathcal{H}\otimes\mathcal{H}_{G,\mathcal{N}})$ operate injectively on $\mathcal{H}_{G,\mathcal{N}}$, i.e. for any basis state $|G\rangle \in \mathcal{H}_{G,\mathcal{N}}$ the value of G is known, when one of the G_i in $U|\Psi\rangle\otimes|G\rangle = \sum_i \alpha_i|\Psi_i\rangle|G_i\rangle$ is known. Let further M_G be the measurement of $\mathcal{H}_{G,\mathcal{N}}$ in the standard basis. Then $\ddot{M}_G\ddot{U}\ddot{M}_G = \ddot{M}_G\ddot{U}$.

This can be proven as follows: Any state in $\mathcal{H}\otimes\mathcal{H}_{G,\mathcal{N}}$ can be written as $\sum_i \beta_i|\Phi_i\rangle|G_i\rangle$ with $\beta_i \in \mathbb{C}$, different basis vectors $|G_i\rangle \in \mathcal{H}_{G,\mathcal{N}}$, and vectors $|\Phi_i\rangle \in \mathcal{H}$. Then we get

$$\begin{aligned}
\varrho\Big(\sum_i \beta_i|\Phi_i\rangle|G_i\rangle\Big) &\xrightarrow{\ddot{M}_G} \sum_i |\beta_i|^2\varrho(|\Phi_i\rangle|G_i\rangle)\\
&\xrightarrow{\ddot{U}} \sum_i |\beta_i|^2\varrho\Big(\sum_j \alpha_{ij}|\Phi_{ij}\rangle|G_{ij}\rangle\Big)\\
&\xrightarrow{\ddot{M}_G} \sum_i |\beta_i|^2\sum_j|\alpha_{ij}|^2\varrho(|\Phi_{ij}\rangle|G_{ij}\rangle).
\end{aligned}$$

Since all the $|G_i\rangle$ were different, the $|G_{ij}\rangle$ were different, too (injectivity of U). Therefore the application of $\ddot{M}_G$ has exchanged the summation and the application of ϱ. The same applies for

$$\begin{aligned}
\varrho\Big(\sum_i \beta_i|\Phi_i\rangle|G_i\rangle\Big) &\xrightarrow{\ddot{U}} \varrho\Big(\sum_i\beta_i\sum_j \alpha_{ij}|\Phi_{ij}\rangle|G_{ij}\rangle\Big)\\
&\xrightarrow{\ddot{M}_G} \sum_i |\beta_i|^2\sum_j|\alpha_{ij}|^2\varrho(|\Phi_{ij}\rangle|G_{ij}\rangle).
\end{aligned}$$

So $\ddot{M}_G\ddot{U}\ddot{M}_G = \ddot{M}_G\ddot{U}$ for all pure states, thus also for all mixed states. ∎

A.6 Lemma 3.30

In section 3.5.1 we have stated the following

Lemma 3.30: Local well-formedness condition for IQTM

An IQTM $\mathcal{M} = (Q, q_0, q_f, n, \delta)$ is well-formed, if and only if the following conditions are fulfilled for δ:

- Unit length: $\|\delta(q, \sigma_1, \ldots, \sigma_n)\| = 1$ for all $q \in Q$, $\sigma_1, \ldots, \sigma_n \in \Sigma$.
- Orthogonality: $\delta(q, \sigma_1, \ldots, \sigma_n) \perp \delta(\tilde{q}, \tilde{\sigma}_1, \ldots, \tilde{\sigma}_n)$ for any $q, \tilde{q} \in Q$, $\sigma_i, \tilde{\sigma}_i \in \Sigma$ with $(q, \sigma_1, \ldots, \sigma_n) \neq (\tilde{q}, \tilde{\sigma}_1, \ldots, \tilde{\sigma}_n)$.
- Separability: For any $I \subseteq \{1, \ldots, n\}$, $I \neq \emptyset$ it is: For $\sigma_i, \tilde{\sigma}_i, \sigma'_i, \tilde{\sigma}'_i \in \Sigma$, $q, \tilde{q} \in Q$, $d_i, \tilde{d}_i \in \{L, R\}$ $(i < n)$, $d_n, \tilde{d}_n \in \{L, R, U, D\}$, $d_i \neq \tilde{d}_i$, it is

$$\sum_{q'} \sum \delta(q, (\sigma_i)_i, (\underline{\sigma}'_i)_i, q', (\underline{d}_i)_i)^\dagger \delta(\tilde{q}, (\tilde{\sigma}_i)_i, (\underline{\tilde{\sigma}}'_i)_i, q', (\underline{\tilde{d}}_i)_i) = 0$$

  where the second sum runs over all $\underline{\sigma}'_i, \underline{\tilde{\sigma}}'_i \in \Sigma$, $\underline{d}_i, \underline{\tilde{d}}_i \in \{L, R\}$ $(i < n)$, $\underline{d}_n, \underline{\tilde{d}}_n \in \{L, R, U, D\}$, which satisfy for $i \in I$: $\sigma'_i = \underline{\sigma}'_i$, $\tilde{\sigma}'_i = \underline{\tilde{\sigma}}'_i$, $d_i = \underline{d}_i$, $\tilde{d}_i = \underline{\tilde{d}}_i$; and for $i \notin I$: $\underline{\sigma}'_i = \underline{\tilde{\sigma}}'_i$, $\underline{d}_i = \underline{\tilde{d}}_i$.

Here δ is used as in the definition of the time evolution (definition 3.21). □

To prove, that from the local well-formedness follows well-formedness, we have to show for any two different basis states $|\Psi\rangle, |\Phi\rangle \in \mathcal{H}_{Q,\mathcal{M}}$ of length 1, that $U_\mathcal{M}|\Psi\rangle \perp U_\mathcal{M}|\Phi\rangle$ and $\|U_\mathcal{M}|\Psi\rangle\| = 1$. If any of those states does not have the form required in the definition of $U_\mathcal{M}$, this is easily verifiable, so we can assume w.l.o.g. two states

$$|\Psi\rangle = |\rlap{/}A(q, u(r_m), u(r), \mu, u(h_1), t_1, u(h_2), t_2, \ldots)\rangle$$

and

$$|\Phi\rangle = |\rlap{/}A(q^*, u(r^*_m), u(r^*), \mu^*, u(h^*_1), t^*_1, u(h^*_2), t^*_2, \ldots)\rangle.$$

For convenience, we consider vectors $|\rlap{/}A(a_1, a_2, \ldots)\rangle \in \mathcal{H}_{Q,\mathcal{M}}$ as $|a_1\rangle \otimes |a_2\rangle \otimes \cdots \in \mathcal{H}_{Q,\mathcal{M},1} \otimes \mathcal{H}_{Q,\mathcal{M},2} \otimes \cdots$.

That $\|U_\mathcal{M}|\Psi\rangle\| = 1$, follows from the definition of $U_\mathcal{M}$ and the unit-length-condition.

If $r_m \neq r^*_m$, then $U_\mathcal{M}|\Psi\rangle$ and $U_\mathcal{M}|\Phi\rangle$ are orthogonal, since they are composed from basis vectors which have $u(r_m)$ resp. $u(r^*_m)$ in $\mathcal{H}_{Q,\mathcal{M},2}$. So we can assume w.l.o.g. $r_m = r^*_m$. With the same argumentation we also get w.l.o.g. $\mu = \mu^*$, and $h_i = h^*_i$ for $i \geq n$, $i \neq n + r$, and of course $t_i = t^*_i$ for $i \geq n + r_m$.

Now if $|\Psi\rangle$ and $|\Phi\rangle$ differ at most in q, t_{i,h_i} $(i = 1, \ldots, n-1, n+r)$, orthogonality of $U_\mathcal{M}|\Psi\rangle$ and $U_\mathcal{M}|\Phi\rangle$ follows from the orthogonality of δ (note $r_m \geq 3$). So we will now assume, that $|\Psi\rangle$ and $|\Phi\rangle$ differ somewhere else.

If for some $i \in \{1, \ldots, n-1\}$ it is $h^*_i \notin \{h_i + 2, h_i, h_i - 2\}$, the images of $|\Psi\rangle$ and $|\Phi\rangle$ will be orthogonal, since they are composed of basis vectors, which have in the subspace containing h_i, h^*_i the contents $u(h_i + L)$ or $u(h_i + R)$, resp. $u(h^*_i + L)$ or $u(h^*_i + R)$, which are different. So we assume $h^*_i \in \{h_i + 2, h_i, h_i - 2\}$ for $i \in \{1, \ldots, n-1, n+r\}$.

Similarly we assume $(h^*_{n+r^*}, r^*) \in \{(h_{n+r} \pm 2, r), (h_{n+r^*}, r \pm 2), (h_{n+r}, r)\} \pmod{(0, r_m)}$.

Let now

$$I := \{i \in \{1, \ldots, n-1\}, h_i \neq h^*_i\} \cup \begin{cases} \{n\}, & \text{if } h_{n+r} \neq h^*_{n+r} \texttt{ or } r \neq r^*, \\ \emptyset, & \text{otherwise.} \end{cases}$$

If $I = \emptyset$, there must be some tape symbols other than t_{i,h_i}, t^*_{i,h^*_i} that differ between $|\Psi\rangle$ and $|\Phi\rangle$, these symbols will also differ after applying $U_\mathcal{M}$, resulting in $U_\mathcal{M}|\Psi\rangle \perp U_\mathcal{M}|\Phi\rangle$. So we will assume $I \neq \emptyset$.

It is

$$\begin{aligned}
&\langle\Psi|U^\dagger_\mathcal{M} U_\mathcal{M}|\Phi\rangle \\
&\quad = \sum \delta(q, (t_{i,h_i})_{i=1}^{n-1}, t_{n+r,h_{n+r}}, q', (\tau_i)_{i=1}^n, (d_i)_{i=1}^n)^\dagger \delta(q^*, (t^*_{i,h^*_i})_{i=1}^{n-1}, t^*_{n+r^*,h^*_{n+r^*}}, q'^*, (\tau^*_i)_{i=1}^n, (d^*_i)_{i=1}^n) \cdot \\
&\langle \rlap{/}A(q', u(r_m), u(r + \bar{d}_n \bmod r_m), \mu, (u(h_i + \tilde{d}_i), \tilde{t}_i)_i) | \rlap{/}A(q'^*, u(r_m), u(r^* + \bar{d}^*_n \bmod r_m), \mu, (u(h^*_i + \tilde{d}^*_i), \tilde{t}^*_i)_i)\rangle,
\end{aligned}$$

where the meanings of the ˜ and the ¯ are analogous to the use in the definition of $U_\mathcal{M}$. The sum runs over all $q', q'^* \in Q$, $\tau_1, \ldots, \tau_n, \tau^*_1, \ldots, \tau^*_n \in \Sigma$, $d_1, \ldots, d_{n-1}, d^*_1, \ldots, d^*_{n-1} \in \{L, R\}$, $d_n, d^*_n \in \{L, R, U, D\}$.

The scalar product inside the sum is non-zero (thus equal 1), if and only if the following conditions are satisfied:

- At least one of the scalar products is non-zero.
- It is $q' = q'^*$.

- For $i \in I$, $i \neq n$, we have or $d_i = R$, $d_i^* = L$ or $d_i = L$, $d_i^* = R$, depending only on how h_i and h_i^* relate, i.e. independent of the summation variables.
- For $i \notin I$, $i \neq n$, we have $d_i = d_i^*$.
- For $n \in I$, we have $d_n, d_n^* \in \{(L, R, U, D)\}$, $d_n \neq d_n^*$. Again the values of d_n, d_n^* are independent of the summation variables.
- For $n \notin I$, we have and $d_i = d_i^*$.
- For $i \in I$, $i \neq n$, it is $t_{i,h_i^*} = \tau_i^*$ and $t_{i,h_i}^* = \tau_i$.
- For $n \in I$, it is $t_{n+r^*,h_{n+r^*}^*} = \tau_i^*$ and $t_{n+r,h_{n+r}}^* = \tau_i$.
- If $i \notin I$, then $\tau_i = \tau_i^*$.

When we remove all summands, in which these conditions are not fulfilled, we get a sum of the same value, which is empty or has the form of the sum in the separability condition. In both cases the sum vanishes, so $U_{\mathcal{M}}|\Psi\rangle \perp U_{\mathcal{M}}|\Phi\rangle$.

The reverse direction of this proof is left to the reader. ∎

B Keyword Index

C Symbol Index

D References

[BV93] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 11–20. ACM Press, 1993.

[Can00a] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 13(1):143–202, 2000.

[Can00b] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. `http://eprint.iacr.org/2000/067` and ECCC TR 01-24. Extended abstract appears in 42nd FOCS, 2000.

[CGS02] Claude Crépeau, Daniel Gottesman, and Adam Smith. Secure multi-party quantum computation. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 643–652. ACM Press, 2002.

[PW94] Birgit Pfitzmann and Michael Waidner. A General Framework for Formal Notions of "Secure" Systems, Hildesheimer Informatik-Berichte 11/94. Technical report, Universität Hildesheim, 1994. Available at `http://www.semper.org/sirene/publ/PfWa_94FormalItsecIB.ps.gz`.

[Smi01] Adam Smith. Multi-party quantum computation. Master's thesis, Department of Electrical Engineering and Computer Science, MIT, August 2001. Available as quant-ph/0111030.