A Process Algebra for Reasoning About Quantum Security

P. Adão [1,3,4]

Center for Logic and Computation, Department of Mathematics
IST, Technical University of Lisbon
Lisbon, Portugal

P. Mateus [2,3,5]

Center for Logic and Computation, Department of Mathematics
IST, Technical University of Lisbon
Lisbon, Portugal

Abstract

We present a process algebra for specifying and reasoning about quantum security protocols. Since the computational power of the protocol agents must be restricted to quantum polynomial-time, we introduce the logarithmic cost quantum random access machine (QRAM) similar to [4,6], and incorporate it in the syntax of the algebra. Probabilistic transition systems give the semantic for the process algebra. Term reduction is stochastic because quantum computation is probabilistic and, moreover, we consider a uniform scheduler to resolve non-deterministic choices. With the purpose of defining security properties, we introduce observational equivalence and quantum computational indistinguishability, and show that the latter is a congruence relation. A simple corollary of this result asserts that any security property defined via emulation is compositional. Finally, we illustrate our approach by establishing the concept of quantum zero-knowledge protocol.

Key words: Quantum process algebras, quantum polynomial-time machines, quantum zero-knowledge proofs

[1] Email: pad@math.ist.utl.pt
[2] Email: pmat@math.ist.utl.pt
[3] Partially supported by FEDER/FCT project QuantLog POCI/MAT/55796/2004.
[4] Additional support from FCT grant SFRH/BD/8148/2002.
[5] Additional support from the Anglo-Portuguese Joint Research Programme, Treaty of Windsor B22/05, 2005-2006.

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

1 Introduction

Security protocols are, in general, composed by several agents running in parallel, where each agent computes information (bounded by polynomial-time on the security parameter) and exchange it with other agents. In the context of quantum processes, the computation is bounded by quantum polynomial-time and the information exchanged is supported by qubits. In this paper, the problem of defining quantum security properties is addressed using a quantum polynomial-time process algebra. This approach is highly inspired in [13,9].

The computational model we adopt to define quantum polynomial terms is based on the logarithmic cost random access machine [4]. A hybrid model, using both classic and quantum memory, similar to [6] but with complexity assumptions, is considered and it is shown to be (polynomial-time) equivalent to a uniform family of quantum circuits (which are, by themselves, equivalent to quantum Turing machines). Such machines model the computation of each agent, receive qubits as input, and return qubits as output.

Thanks to the non-cloning theorem, quantum information cannot be copied without prior knowledge of its state. This observation imposes some design options in the process algebra, since it is necessary to know which agent possesses a qubit in order to know who can retrieve each piece of information. In order to deal with this fact, a set of agents is fixed and the qubits are partitioned among them.

Although several other approaches to quantum process algebras are already present in the literature (see [5], for instance), ours is quite original, due to the universe of application—security protocols. In our approach, process terms are divided into local and global. An agent is modelled by a local process while a protocol is modelled by a global process so, a global process corresponds to local processes running in parallel. A semantics based on probabilistic transition systems (which can be easily translated to Markov chains) is provided, and the probabilistic transitions are defined using rules and assuming a uniform scheduler to resolve non-deterministic choices.

Agent observation is defined as a probability distribution over binary words obtained by measuring on the computational basis (some of) the agent's qubits. This measurement is done at the end of a protocol run. This concept is the key ingredient to establish observational equivalence that, in the context of security protocols, is based on computational indistinguishability [14]. Intuitively, two process terms are observational equivalent for an agent if, after making all possible reductions to each process, it is impossible to distinguish (in quantum polynomial-time) the qubits of the agent on both processes. Since we internalize quantum polynomial-time machines in the process algebra language, observational equivalence is easily defined and it is shown to be a congruence relation.

One of the most successful ways for defining secure concurrent cryptographic tasks is via process emulation [1,2]. This definitional job boils down

to the following: a process realizes a cryptographic task if and only if it emulates an ideal process that is known to realize such task. Based on the notion of observational equivalence, we establish the notion of emulation for the quantum process calculus and show that it is compositional. Finally, we provide the notion of quantum zero-knowledge via process emulation.

In Section 2 the process algebra is introduced together with the logarithmic cost random access machine. Both the syntax and the semantics of the process algebra are clearly established, and the section is concluded by presenting the notion of observational equivalence. Section 3 is devoted to emulation and its composition theorem, and finally, in Section 4 quantum zero-knowledge is defined using process emulation.

2 Process Algebra

In the context of security protocols it is common to consider a security parameter $\eta \in \mathbb{N}$. In the case of quantum protocols we will also consider such parameter in order to bound the quantum complexity of the principals and adversaries. From now on, the symbol η is reserved to designate such security parameter. The role of this parameter is twofold: it bounds to a polynomial on η the number of qubits that can be sent through channels, and it bounds all the computation to quantum polynomial time (on η). We now detail these aspects culminating with the presentation of the process algebra language.

2.1 Quantum polynomial machines

The computational model we adopted to define quantum polynomial machine is based on the logarithmic cost random access machine [4] and it is quite similar to the quantum random access machine in [6]. We consider a hybrid model using both classic and quantum memory. In order to cope with a countable set of qubits qB we adopt the following Hilbert space $\mathcal{H}$ (isomorphic to $\ell^2(2^{qB})$ and $L^2(2^{qB}, \#)$) to model the quantum state (see [10,11] for a discussion on why $\mathcal{H}$ is the correct Hilbert space for modelling a countable set of qubits):

- each element is a map $|\psi\rangle : 2^{qB} \to \mathbb{C}$ such that:
  - $\mathrm{supp}(|\psi\rangle) = \{v \in 2^{qB} : |\psi\rangle(v) \neq 0\}$ is countable;
  - $\sum_{v \in 2^{qB}} ||\psi\rangle(v)|^2 = \sum_{v \in \mathrm{supp}(|\psi\rangle)} ||\psi\rangle(v)|^2 < \infty$;
- $|\psi_1\rangle + |\psi_2\rangle = \lambda v.\, |\psi_1\rangle(v) + |\psi_2\rangle(v)$;
- $z|\psi\rangle = \lambda v.\, z|\psi\rangle(v)$;
- $\langle \psi_1 | \psi_2 \rangle = \sum_{v \in V} \overline{|\psi_1\rangle(v)}\, |\psi_2\rangle(v)$.

The inner product induces the norm $|||\psi\rangle|| = \sqrt{\langle \psi | \psi \rangle}$ and so, the distance $d(|\psi_1\rangle, |\psi_2\rangle) = |||\psi_1\rangle - |\psi_2\rangle||$. Clearly, $\{|v\rangle : v \in 2^{qB}\}$ is an orthonormal basis of $\mathcal{H}$ where $|v\rangle(v) = 1$ and $|v\rangle(v') = 0$ for every $v' \neq v$. This basis is called

the computational or logic basis of $\mathcal{H}$.

A configuration of a quantum random access machine (QRAM) is triple $\xi = (\mathbf{m}, |\psi\rangle, s)$ where $\mathbf{m} \in \mathbb{N}^{\mathbb{N}}$, $|\psi\rangle \in \mathcal{H}$ and $s \in \mathbb{N}$. The first component of the triple represents the classical memory of the machine—an infinite sequence of natural numbers, the second component represents the quantum state of the machine, and finally the third component is a counter that indicates how many (qu)bit operations are allowed.

We associate to each QRAM a positive polynomial q for bounding the number of allowed (qu)bit operations to $q(\eta)$. In this way, we force each QRAM to terminate in polynomial-time. Given a finite set of qubits at state $|\varphi\rangle$, the *initial configuration* of the QRAM is the triple $\xi_0(|\varphi\rangle) = (\mathbf{m}_0, |\varphi\rangle \otimes |\mathbf{0}\rangle, q(\eta))$, where the sequence $\mathbf{m}_0$ is such that $\mathbf{m}_0(k) = 0$ for all $k \in \mathbb{N}$ and $|\mathbf{0}\rangle$ is the unit vector in $\mathcal{H}$ such that $|\mathbf{0}\rangle(\emptyset) = 1$ (note that if $\mathcal{Q}$ is a 2^n dimension Hilbert space, then there is a canonical isomorphism between $\mathcal{H}$ and $\mathcal{Q} \otimes \mathcal{H}$, and therefore $|\varphi\rangle \otimes |\mathbf{0}\rangle \in \mathcal{Q} \otimes \mathcal{H}$ can be seen as a unit vector in $\mathcal{H}$). A QRAM receives as input a finite sequence of qubits, but since it is always possible to encode classical bits in qubits this is not a limitation.

The set of atomic commands $\mathcal{AC}$, and their associated cost is presented in the table below [6].

Number	Instruction	Computational cost
1	$R_i = n$	$\lvert n\rvert$
2	$R_i = R_j$	$\lvert R_j\rvert$
3	$R_i = R_j + R_k$	$\lvert R_j\rvert + \lvert R_k\rvert$
4	$R_i = R_j - R_k$	$\lvert R_j\rvert + \lvert R_k\rvert$
5	$R_i = R_j R_k$	$\lvert R_j\rvert \times \lvert R_k\rvert$
6	$R_i = R_j / R_k$	$\lvert R_j\rvert \times \lvert R_k\rvert$
7	$R_i = R_{R_j}$	$\lvert R_j\rvert + \lvert R_{R_j}\rvert$
8	$R_{R_i} = R_j$	$\lvert R_i\rvert + \lvert R_j\rvert$
9	$\texttt{Pauli}_X[b]$	1
10	$\texttt{Pauli}_Y[b]$	1
11	$\texttt{Pauli}_Z[b]$	1
12	$\texttt{Hadamard}[b]$	1
13	$\texttt{phase}[b]$	1
14	$\frac{\pi}{8}[b]$	1
15	$\texttt{c-not}[b_1, b_2]$	1
16	$\texttt{measure}[b] \rightarrow R_i$	1

Most of the commands above are self-explanatory, but it is worthwhile to notice that all commands are deterministic with exception of `measure`. Indeed, according to the measurement postulates of quantum mechanics (see for instance [3]), when a quantum system is measured the outcome is stochastic, and moreover the state evolves accordingly to this outcome. Note that we only

[6] We denote the number of bits required to represent a natural number n by $|n|$.

consider measurements over the computational basis, nevertheless this is not a limitation since any other qubit measurement can be performed by applying a unitary transformation before measuring the qubit over the computational basis.

The set of QRAM *commands* $\mathcal{C}$ is obtained inductively as follows:

(i) $a \in \mathcal{C}$ if $a \in \mathcal{AC}$;

(ii) $c_1; c_2 \in \mathcal{C}$ if $c_1, c_2 \in \mathcal{C}$;

(iii) $(\texttt{if}\ \ (R_n > 0)\ \ \texttt{then}\ \ c) \in \mathcal{C}$ if $c \in \mathcal{C}$;

(iv) $(\texttt{while}\ \ (R_n > 0)\ c) \in \mathcal{C}$ if $c \in \mathcal{C}$.

The *execution* of a QRAM command c is a stochastic function between configurations. Let $\Xi = \mathbb{N}^{\mathbb{N}} \times \mathcal{H} \times \mathbb{N}$ be the set of all configurations, and $\text{Prob}_{\text{fin}}(\Xi)$ be the set of all probability measures over $(\Xi, 2^{\Xi})$ such that only a finite set of configurations have probability different from 0. The execution of a QRAM command c is a map $\text{run}_c : \Xi \to \text{Prob}_{\text{fin}}(\Xi)$, and we write $[c]\ \xi \to_p \xi'$ to denote that $\Pr_{\text{run}_c(\xi)}(\xi') = p$. The execution of QRAM commands can be defined using the following rules, which are quite intuitive:

$$\frac{s \geq |n|}{[R_i = n]\ \ (\mathbf{m}, |\psi\rangle, s) \to_1 (\mathbf{m}', |\psi\rangle, s - |n|)}\ (R_i = n),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = n$;

$$\frac{s \geq |R_j|}{[R_i = R_j]\ \ (\mathbf{m}, |\psi\rangle, s) \to_1 (\mathbf{m}', |\psi\rangle, s - |R_j|)}\ (R_i = R_j),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = \mathbf{m}(j)$;

$$\frac{s \geq |R_j| + |R_k|}{[R_i = R_j + R_k]\ \ (\mathbf{m}, |\psi\rangle, s) \to_1 (\mathbf{m}', |\psi\rangle, s - (|R_j| + |R_k|))}\ (R_i = R_j + R_k),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = \mathbf{m}(j) + \mathbf{m}(k)$;

$$\frac{s \geq |R_j| + |R_k|}{[R_i = R_j - R_k]\ \ (\mathbf{m}, |\psi\rangle, s) \to_1 (\mathbf{m}', |\psi\rangle, s - (|R_j| + |R_k|))}\ (R_i = R_j - R_k),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = \max(\mathbf{m}(j) - \mathbf{m}(k), 0)$;

$$\frac{s \geq |R_j| \times |R_k|}{[R_i = R_j R_k]\ \ (\mathbf{m}, |\psi\rangle, s) \to_1 (\mathbf{m}', |\psi\rangle, s - (|R_j| \times |R_k|))}\ (R_i = R_j R_k),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = \mathbf{m}(j)\mathbf{m}(k)$;

$$\frac{s \geq |R_j| \times |R_k|}{[R_i = R_j / R_k]\ \ (\mathbf{m}, |\psi\rangle, s) \to_1 (\mathbf{m}', |\psi\rangle, s - (|R_j| \times |R_k|))}\ (R_i = R_j / R_k),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = \lfloor \mathbf{m}(j)/\mathbf{m}(k) \rfloor$;

$$\frac{s \geq |R_j| + |R_{R_j}|}{[R_i = R_{R_j}] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}', |\psi\rangle, s - (|R_j| + |R_{R_j}|))} \; (R_i = R_{R_j}),$$

where $\mathbf{m}'(k) = \mathbf{m}'(k)$ for all $k \neq i$ and $\mathbf{m}'(i) = \mathbf{m}(\mathbf{m}(j))$;

$$\frac{s \geq |R_i| + |R_j|}{[R_{R_i} = R_j] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}', |\psi\rangle, s - (|R_i| + |R_j|))} \; (R_{R_i} = R_j),$$

where $\mathbf{m}'(k) = \mathbf{m}(k)$ for all $k \neq \mathbf{m}(i)$ and $\mathbf{m}'(\mathbf{m}(i)) = \mathbf{m}(j)$;

$$\frac{s \geq 1}{[\texttt{Pauli}_X[b]] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}, |\psi'\rangle, s - 1)} \; (\texttt{Pauli}_X[b]),$$

where $|\psi'\rangle$ is obtained from $|\psi\rangle$ by applying the Pauli_X operator $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ on qubit b. Similar rules apply to the following one-qubit operators:

$$\text{Pauli}_Y \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}; \qquad \text{Pauli}_Z \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix};$$

$$\text{Hadamard } \tfrac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \qquad \text{Phase } \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}; \qquad \tfrac{\pi}{8} \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix};$$

$$\frac{s \geq 1}{[\texttt{c-not}[b_1, b_2]] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_1 (\mathbf{m}, |\psi'\rangle, s - 1)} \; (\texttt{c-not}[b_1, b_2]),$$

where $|\psi'\rangle$ is obtained from $|\psi\rangle$ by applying the control-not operator

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

on qubits b_1 and b_2;

$$\frac{s \geq 1}{[\texttt{measure}[b] \rightarrow R_i] \;\; (\mathbf{m}, |\psi\rangle, s) \rightarrow_p (\mathbf{m}', |\psi'\rangle, s - 1))} \; (\texttt{measure}[b] \rightarrow R_i = 0),$$

where $|\psi'\rangle$ is equal to $\frac{P_0|\psi\rangle}{|P_0|\psi\rangle|}$, $p = |P_0|\psi\rangle|$ (P_0 is the projector onto the subspace of $\mathcal{H}$ where qubit b takes value $|0\rangle$), $\mathbf{m}'(i) = 0$ and $\mathbf{m}'(j) = \mathbf{m}(j)$ for all $j \neq i$;

$$\frac{s \geq 1}{[\texttt{measure}[b] \to R_i] \ \ (\mathbf{m}, |\psi\rangle, s) \to_p (\mathbf{m}', |\psi'\rangle, s-1))} \ (\texttt{measure}[b] \to R_i = 1),$$

where $|\psi'\rangle$ is equal to $\frac{P_1|\psi\rangle}{|P_1|\psi\rangle|}$, $p = |P_1|\psi\rangle|$ (P_1 is the projector onto the subspace of $\mathcal{H}$ where qubit b takes value $|1\rangle$), $\mathbf{m}'(i) = 1$ and $\mathbf{m}'(j) = \mathbf{m}(j)$ for all $j \neq i$;

$$\frac{[c_1] \ (\mathbf{m}, |\psi\rangle, s) \to_{p_1} (\mathbf{m}', |\psi'\rangle, s') \quad [c_2] \ (\mathbf{m}', |\psi'\rangle, s') \to_{p_2} (\mathbf{m}'', |\psi''\rangle, s'')}{[c_1; c_2] \ (\mathbf{m}, |\psi\rangle, s) \to_{p_1 \times p_2} (\mathbf{m}'', |\psi''\rangle, s'')} \ (c_1; c_2);$$

$$\frac{\mathbf{m}(n) > 0 \qquad s \geq |R_n| \qquad [c] \ \ (\mathbf{m}, |\psi\rangle, s - |R_n|) \to_p (\mathbf{m}', |\psi'\rangle, s')}{[(\texttt{if} \ \ (R_n > 0) \ \ \texttt{then} \ \ c)] \ \ (\mathbf{m}, |\psi\rangle, s) \to_p [c] \ \ (\mathbf{m}', |\psi'\rangle, s')} \ (\texttt{if}\top);$$

$$\frac{\mathbf{m}(n) = 0}{[(\texttt{if} \ \ (R_n > 0) \ \ \texttt{then} \ \ c)] \ \ (\mathbf{m}, |\psi\rangle, s) \to_1 (\mathbf{m}, |\psi\rangle, s)} \ (\texttt{if}\bot);$$

$$\frac{\begin{array}{c}\mathbf{m}(n) > 0 \qquad s \geq |R_n| \\ [c; (\texttt{while} \ \ (R_n > 0) \ c)] \ \ (\mathbf{m}, |\psi\rangle, s - |R_n|) \to_p (\mathbf{m}', |\psi'\rangle, s')\end{array}}{[(\texttt{while} \ \ (R_n > 0) \ c)] \ \ (\mathbf{m}, |\psi\rangle, s) \to_p (\mathbf{m}', |\psi'\rangle, s')} \ (\texttt{while}\top);$$

$$\frac{\mathbf{m}(n) = 0}{[(\texttt{while} \ \ (R_n > 0) \ c)] \ \ (\mathbf{m}, |\psi\rangle, s) \to_1 (\mathbf{m}, |\psi\rangle, s - |R_n|)} \ (\texttt{while}\bot).$$

Observe, that the reduction of QRAM commands always terminate, since every computation is bounded by $q(\eta)$ (qu)bit steps. The execution of a QRAM command can be seen as a word run of a quantum automata [8], however a detailed discussion about this subject is out of the scope of this abstract.

The output of a QRAM is the quantum state of a set of qubits. This output set is determined by another positive polynomial o associated to the machine. Given a security parameter η, the set of output qubits is constituted by the first $o(\eta)$ qubits.

Definition 2.1 A *quantum polynomial machine* is a triple $M = (c, q, o)$ where c is a QRAM command, q is a positive step bounding polynomial and o is a positive output polynomial. We denote the set of all these triples by QPM.

Given a quantum polynomial machine M and a security parameter η, the computation of M over state $|\psi\rangle$ is the probability distribution over the state of the first $o(\eta)$ qubits of $|\psi'\rangle$, where this distribution is defined by the execution rules $[c](\mathbf{m}_0, |\psi\rangle, q(\eta)) \to_p (\mathbf{m}', |\psi'\rangle, s')$. Hence, the computation of a QRAM

is a probability distribution over the state space of the first $o(\eta)$ qubits. It is traditional in quantum algorithms to measure all relevant qubits at the end of the computation in order to obtain a classical result (see Shor's and Grover's algorithms). However, since we use QRAM to compute quantum information that can be sent through quantum channels, we do not impose this final measurement since it may be desirable to send a superposition through a quantum channel.

The following result asserts that the QRAM model is equivalent to the usual quantum circuit computational model (a careful presentation of this result is out of the scope of this abstract).

Proposition 2.2 *For any uniform family of polynomial quantum circuits $Q = \{Q_\eta\}_{\eta\in\mathbb{N}}$, there exists a quantum polynomial machine M_Q such that the M_Q computes the same stochastic function as Q. Moreover, for any quantum polynomial machine M there exists an equivalent uniform family of polynomial quantum circuits $Q_M = \{Q_\eta\}_{\eta\in\mathbb{N}}$.*

Proof (Sketch): Note that a uniform circuit uses precisely the gates defined as quantum atomic commands of the QRAM. The construction of the circuit can be mimicked by a RAM command c. Since this construction must be polynomial in η, the program must terminate in polynomial time and therefore, there is a polynomial q to bound the number of steps, finally the output must always be a polynomial set of qubits, and therefore we are able to construct an equivalent QRAM machine.

On the other hand a QRAM program is the realization of the uniform family construction, since, for each η, a circuit can be retrieved by looking at the finite (do not forget that QRAM programs always terminate) sequence of quantum atomic gates generated by the execution of the command. The stochastic nature of the execution does not bring a problem, since gates placed after a measurement can be controlled by the outcome of that measurement. If a measurement gives the value 1 to a qubit and in that case a gate U is placed at some qubit b, then the circuit should be constructed by placing a control-U gate controlled by the measured qubit and targeted at b. □

2.2 Process algebra

As stated before, we require to know who possesses a qubit in order to know who can retrieve some piece of information. In order to deal with this fact, a qubit is considered to belong to some agent, and therefore, the set of qubits qB is partitioned among all agents. To make this more precise, a countable set $A = \{a_1, \ldots, a_k, \ldots\}$ of agents is fixed once and for all, and moreover the partition $q\mathcal{B} = \{qB_{a_i}\}_{a_i\in A}$ of qB is such that each set qB_{a_i} is countable and recursively enumerable.

Note that each qB_{a_i} has a total order (with a bottom element) induced by its recursive enumeration. The purpose of this total ordering is to reindex the qubits accessed by a QPM M when an agent a executes M. An obvious

desideratum of the system is that an agent a is restricted to compute over its own qubits qB_a, and therefore, when agent a executes a quantum polynomial machine M, this machine must have access only to the qubits in qB_a (note that if the qubits of a are entangled with other qubits, then when the former are modified so can be the latter). Therefore, if, for instance, an agent a executes a machine that consists of the command $\texttt{Pauli}_X[b]$, and if qB_a is recursively enumerated by γ, then the command effectively executed is $\texttt{Pauli}_X[\gamma(b)]$. The same procedure applies to the input and output qubits, so when a machine executed by a outputs the first $o(\eta)$ qubits, the machine is in fact outputting the qubits $\{\gamma(o(1)), \ldots, \gamma(o(\eta))\} \subset qB_a \subseteq qB$.

Communication between agents is achieved via public channels, allowing qubits to be exchanged. Clearly, this process is modelled by modifying the partition of qB. It is also convenient to allow parallelism inside an agent (that is, an agent may be constituted by several processes in parallel), for this purpose, private channels (that cannot be intercepted) allowing communication between the agent local processes are introduced. To make this assumptions clear, two countable disjoint sets of *quantum channels* are considered, the set of *global or public channels* $G = \{g_1, g_2, \ldots, g_k, \ldots\}$, and the set of *local or private channels* $L = \{l_1, l_2, \ldots, l_k, \ldots\}$. We denote by C the set $G \cup L$. All global channels can be read and written by an adversary while local channels correspond to private communication from one agent to itself. One role of the security parameter is to bound the bandwidth of the channels. Hence, we introduce a *bandwidth map* $\mathbf{bw} : C \to \mathbf{q}$, where $\mathbf{q}$ is the set of all polynomials taking positive values. Given a value η for the security parameter, a channel c can send at most $\mathbf{bw}(c)(\eta)$ qubits.

We also consider a countable set of variables $Var = \{x_1, x_2, \ldots, x_k, \ldots\}$, which are required to define qubit terms. A qubit term t is either a finite subset of qB or a variable $x \in Var$.

Finally, we present the language of processes, which is a fragment of π-calculus. Mind that the overall computation must be quantum polynomial on η and therefore we do not cope with recursion nor mobility. First, we establish the language of an agent, that we call local process language.

Definition 2.3 The *language of local processes* $\mathcal{L}$ is obtained inductively as follows:

(i) $0 \in \mathcal{L}$ (termination);

(ii) $c\langle M(t)\rangle \in \mathcal{L}$ where $M \in \mathrm{QPM}$, t is a qubit term, and $c \in C$ (output);

(iii) $c(x).Q \in \mathcal{L}$ where $c \in C$, $x \in Var$ and $Q \in \mathcal{L}$ (input);

(iv) $[M(t) = 0].Q$ where $M \in \mathrm{QPM}$, t is a qubit term, and $Q \in \mathcal{L}$ (match);

(v) $(Q_1|Q_2)$ where $Q_1, Q_2 \in \mathcal{L}$ (parallel composition);

(vi) $!_q Q$ where $Q \in \mathcal{L}$ and $q \in \mathbf{q}$ (bounded replication).

Most of the (local) process terms are intuitive. The output term $c\langle M(qB')\rangle$ means that the output of machine M, which received the finite set of qubits

qB' as input, is sent through channel c. The input term $c(x).Q$ means that a set of qubits is going to be received on c, and upon reception, x takes the value of the received qubits.

After fixing the security parameter η, we can get rid of replication by evaluating each process $!_q R$ as $q(\eta)$ copies of R in parallel. Therefore, we always assume that a process term has no replication. Now, as state before, a protocol is constituted by a set o agents running in parallel, therefore the global language (or protocol language) is quite simple:

Definition 2.4 The *language of global processes* $\mathcal{G}$ over a set of agents A is defined inductively as follows:

(i) $\mathbf{0} \in \mathcal{G}$ (global termination);

(ii) $P \| (a : Q) \in \mathcal{G}$ where $P \in \mathcal{G}$, $a \in A$ does not occur in P, and $Q \in \mathcal{L}$ (global parallel composition).

The following example uses the process language to describe the RSA cryptanalysis using Shor's algorithm.

Example 2.5 [Shor's based RSA cryptanalysis] Let p, q be primes (with η length binary expansion), and e, d integers such that $ed \equiv 1 \mod \phi(pq)$. Alice is a simple process A that knows some message w and outputs $w^e \mod pq$, where e is the public key of Bob. This dummy process can be presented as

$$(a : A(w)) := (a : g\langle w^e \mod pq \rangle).$$

Bob receives x and computes $x^d \mod pq$. This procedure can be modelled by the following process:

$$(b : B) := (b : g(x).(l\langle x^d \mod pq \rangle | l(y).0)).$$

Therefore the RSA protocol is given by the process $(a : A(w)) \| (b : B)$. Finally, we can write the "attacking" process, Eve. She factorises pq, inverts e mod $\phi(pq)$ (thus, allowing her to find d), and intercepts the message sent by Alice (on channel g). We write this process as follows:

$$(c : l_1\langle \mathrm{Shor}(pq) \rangle | l_1(y).l_2\langle \mathrm{Inv}(y, e) \rangle | g(x).l_2(z).(l_3\langle x^z \mod pq \rangle | l_3(w).0)).$$

2.3 Semantics

In order to define the semantics of a local process we need to introduce the notion of local configuration. A *local configuration* or *agent configuration* is a triple $(|\psi\rangle, qB_a, Q)$ where $|\psi\rangle \in \mathcal{H}$, $qB_a \subseteq qB$ is a countable, recursive enumerable set and $Q \in \mathcal{L}$. The first element of the local configuration is the global state of the protocol, the second element is the set of qubits the agent possesses and the last element is the local process term.

The semantics of a local process is a probabilistic transition system where the transitions are defined by rules. We use $(|\psi\rangle, qB_a, Q) \to_p (|\psi'\rangle, qB_a, Q')$

to state that, at global state $|\psi\rangle$, when agent a possesses qubits qB_a, the local process Q is reduced to Q' and global state is modified to $|\psi'\rangle$ with probability p. It is also worthwhile to observe that we use the notation $M(|\psi\rangle, qB_a, qB_1) \rightarrow_p (|\psi'\rangle, qB_2)$ to denote that the execution of the QRM M, operating on qB_a (that is, using the recursive enumeration of qB_a to reindex the position of the qubits), and receiving as input qB_1, outputs qB_2 and modifies the global state $|\psi\rangle$ to $|\psi'\rangle$ with probability p. For the case of local processes, the sets qB_1 and qB_2 are irrelevant, because the qubits owned by the agent remain the same when a local communication (LCom rule) is applied. Their functionality will be clear when we present the global rules.

$$\frac{M(|\psi\rangle, qB_a, qB_1) \rightarrow_p (|\psi'\rangle, qB_2) \quad qB_1, qB_2 \subseteq qB_a \quad |qB_2| \leq \mathbf{bw}(l)(\eta)}{(|\psi\rangle, qB_a, l(x).Q | l\langle M(qB_1)\rangle) \rightarrow_p (|\psi'\rangle, qB_a, Q^x_{qB_2})} \text{ (LCom)}$$

We also introduce the term $M;\mathrm{Meas}$ to denote the machine that, after executing M performs a measurement on the computational basis of the output qubits of M. So a match corresponds to performing a measurement on the output qubits of M and checking whether the result is the 0 word.

$$\frac{(M;\mathrm{Meas})(|\psi\rangle, qB_a, qB_1) \rightarrow_p (|\psi'\rangle, qB_2) \quad |\psi'\rangle|_{qB_2} = |\mathbf{0}\rangle}{(|\psi\rangle, qB_a, [M(qB_1) = 0].Q) \rightarrow_p (|\psi'\rangle, qB_a, Q)} \text{ (Match}\top\text{)}$$

$$\frac{(M;\mathrm{Meas})(|\psi\rangle, qB_a, qB_1) \rightarrow_p (|\psi'\rangle, qB_2) \quad |\psi'\rangle|_{qB_2} \neq |\mathbf{0}\rangle}{(|\psi\rangle, qB_a, [M(qB_1) = 0].Q) \rightarrow_p (|\psi'\rangle, qB_a, \mathbf{0})} \text{ (Match}\bot\text{)}$$

The remaining rules are self-explanatory.

$$\frac{(|\psi\rangle, qB_a, P) \rightarrow_p (|\psi'\rangle, qB_a, P')}{(|\psi\rangle, qB_a, P|Q) \rightarrow_p (|\psi'\rangle, qB_a, P'|Q)} \text{ (LLPar)}$$

$$\frac{(|\psi\rangle, qB_a, Q) \rightarrow_p (|\psi'\rangle, qB_a, Q')}{(|\psi\rangle, qB_a, P|Q) \rightarrow_p (|\psi'\rangle, qB_a, P|Q')} \text{ (LRPar)}$$

We proceed by presenting the global rules. A *global configuration* is a triple $(|\psi\rangle, q\mathcal{B}, P)$ where $|\psi\rangle \in \mathcal{H}$, $q\mathcal{B} = \{qB_a\}_{a\in A}$ is a partition of qB indexed by the set of agents A (where each qB_a is countable and r.e.) and $P \in \mathcal{G}$. The semantics of a global process is defined by the following rules:

$$\frac{(|\psi\rangle, qB_a, Q) \rightarrow_p (|\psi'\rangle, qB_a, Q')}{(|\psi\rangle, q\mathcal{B}, (a : Q)) \rightarrow_p (|\psi'\rangle, q\mathcal{B}, (a : Q))} \text{ (LtoG)}$$

$$\frac{M(|\psi\rangle, qB_b, qB_1) \rightarrow_p (|\psi'\rangle, qB_2) \quad qB_1, qB_2 \subseteq qB_b \quad |qB_2| \leq \mathbf{bw}(g)(\eta)}{(|\psi\rangle, q\mathcal{B}, (a : g(x).Q) \| (b : g\langle M(qB_1)\rangle)) \rightarrow_p (|\psi'\rangle, q\mathcal{B}', (a : Q^x_{qB_2}))} \text{ (GCom)}$$

where $q\mathcal{B}' = \{qB'_a\}_{a\in A}$, $qB'_a = qB_a \cup qB_2$, $qB'_b = qB_b \setminus qB_2$, and $qB'_c = qB_c$ for all $c \neq a, b$.

$$\frac{(|\psi\rangle, q\mathcal{B}, P_1) \to_p (|\psi'\rangle, q\mathcal{B}', P_1')}{(|\psi\rangle, q\mathcal{B}, P_1 \| P_2) \to_p (|\psi'\rangle, q\mathcal{B}', P_1' \| P_2)} \text{ (GLPar)}$$

$$\frac{(|\psi\rangle, q\mathcal{B}, P_2) \to_p (|\psi'\rangle, q\mathcal{B}', P_2')}{(|\psi\rangle, q\mathcal{B}, P_1 \| P_2) \to_p (|\psi'\rangle, q\mathcal{B}', P_1 \| P_2')} \text{ (GRPar).}$$

All the rules are very simple to grasp. The only non trivial rule is global communication (GCom), that makes qubits to be exchanged from one agent to another, and therefore an adjustment is required in the qubit partition.

Process term reductions are non-deterministic, in the sense that several different reductions could be chosen at some step. In order to be possible to make a quantitative analysis, this reduction should be probabilistic. For the sake of simplicity, we assume a uniform scheduler, that is, the choice on any possible reduction is done with uniform probability over all possible non-deterministic reductions. We do not present in detail the scheduler model but, in principle, any probability distribution modelled by a QPM can be used to model the scheduler policy. Finally, note that by applying local and global rules, and assuming a uniform scheduler, one can define the many step reduction $\to_p^*$ such that $(|\psi_1\rangle, q\mathcal{B}_1, P_1) \to_p^* (|\psi_n\rangle, q\mathcal{B}_n, P_n)$, whenever:

- $(|\psi_1\rangle, q\mathcal{B}_1, P_1) \to_{p_1} (|\psi_2\rangle, q\mathcal{B}_2, P_2) \to_{p_2} \cdots \to_{p_{n-1}} (|\psi_n\rangle, q\mathcal{B}_n, P_n)$;
- $p = \frac{p_1}{R_1} \times \frac{p_2}{R_2} \times \cdots \times \frac{p_{n-1}}{R_{n-1}}$ where R_i is the number of possible non-deterministic choices for $(|\psi_i\rangle, q\mathcal{B}_i, P_i)$ for all $i \in \{1, \ldots, n-1\}$;
- $(|\psi_n\rangle, q\mathcal{B}_n, P_n)$ cannot be reduced any more.

The many step reduction takes into account the scheduler choice, by weighting each stochastic reduction p_i with yet another probability $\frac{1}{R_i}$, where R_i is the number of possible non-deterministic choices at step i.

2.4 Observations and observational equivalence

At the end of a protocol, each agent $a \in A$ is allowed to measure a polynomial (in η) number of qubits in qB_a to extract information. We can always assume that these qubits are the first, say, $r(\eta)$ qubits of qB_a where r is a positive polynomial. Therefore, the many step reduction of a process term P induces a probability distribution on $2^{r(\eta)}$, where $2^{r(\eta)}$ is the set of all possible outcomes of $r(\eta)$ qubits when measured over the computational basis (that is, $2^{r(\eta)}$ is the set of all $r(\eta)$-long binary words).

Definition 2.6 Given a positive polynomial r and a global configuration $(|\psi\rangle, q\mathcal{B}, P)$, let

$$\Gamma_{(|\psi\rangle, q\mathcal{B}, P)} = \{(|\psi'\rangle, q\mathcal{B}', P') : (|\psi\rangle, q\mathcal{B}, P) \to_p^* (|\psi'\rangle, q\mathcal{B}', P') \text{ and } p > 0\}.$$

We define the *observation of an agent a* to be the family of probability measures

$$O_r^a = \{(2^{r(\eta)}, 2^{2^{r(\eta)}}, \Pr_{r(\eta)}^a)\}_{\eta \in \mathbb{N}}$$

where:

- $\Pr^a_{r(\eta)}(\{w\}) = \sum_{\gamma \in \Gamma_{(|\psi\rangle, q\mathcal{B}, P)}} p_\gamma \times |\langle w|\psi_\gamma\rangle|$;
- p_γ is such that $(|\psi\rangle, q\mathcal{B}, P) \rightarrow^*_{p_\gamma} \gamma$;
- $|\psi_\gamma\rangle$ is the first component of γ;
- $|\langle w|\psi_\gamma\rangle|$ is the probability of observing the $r(\eta)$-long binary word w by measuring the $r(\eta)$ first qubits of qB_a (qubits in possession of agent a) of $|\psi_\gamma\rangle$ in the computational basis.

Note that the summation used to compute $\Pr^a_{r(\eta)}(\{w\})$ is well defined, since $\Gamma_{(|\psi\rangle, q\mathcal{B}, P)}$ is finite. It is clear at this point,that an observation of an agent is a random $r(\eta)$-long binary word, with distribution given by $\Pr^a_{r(\eta)}$.

The notion of observational equivalence we adopt is based on computational indistinguishability as usual in the security community [13]. First, we introduce the concept of context. The set of *global contexts* $\mathcal{C}$ is defined inductively as follows: $[\,] \in \mathcal{C}$; $C[\,]\|P$ and $P\|C[\,] \in \mathcal{C}$ provided that $C[\,] \in \mathcal{C}$ and $P \in \mathcal{G}$. Given a context $C[\,]$ and a global process P, the notation $C[P]$ means that we substitute the process P for the $[\,]$ in $C[\,]$.

Definition 2.7 Let P and P' be process terms. We say that P is *computationally indistinguishable by agent a* from P' if and only if for every context $C[\,]$, polynomials q and r, $|\psi\rangle \in \mathcal{H}$, partition $q\mathcal{B}$ of qB, η sufficiently large and binary word $w \in 2^{r(\eta)}$,

$$|\Pr^a_{r(\eta)}(w) - \Pr'^a_{r(\eta)}(w)| \leq \frac{1}{q(\eta)}$$

where $\Pr^a_{r(\eta)}$ is given by the observation of a for configuration $(|\psi\rangle, q\mathcal{B}, C[P])$ and $\Pr'^a_{r(\eta)}$ is given by the observation of a for configuration $(|\psi\rangle, q\mathcal{B}, C[P'])$. In such case we write $P \simeq P'$.

Two processes are computationally indistinguishable if they are indistinguishable by contexts, that is, for any input (here modelled by $|\psi\rangle$ and $q\mathcal{B}$), there is no context which can distinguish, up to a negligible function, the outputs produced. The definition above extends the classical definition of computational indistinguishability to the quantum case, since processes can be modelled by quantum polynomial machines and therefore $C[\,]$ induces the required distinguishing machine. A detailed proof of this result is out of the scope of this extended abstract.

In order to set up compositionality, the following result is of the utmost importance:

Proposition 2.8 *Computational indistinguishability is a congruence relation with respect to the parallel primitive of $\mathcal{G}$.*

Proof. Both symmetry and reflexivity are trivial to check. Transitivity follows by triangular inequality, and taking into account that $\frac{1}{2}q(n)$ is a polyno-

mial. Congruence on the global parallel operator follows by noticing that for any contexts $C[\,]$ and $C'[\,]$, $C'[C[\,]]$ is also a context. □

3 Emulation and Composition Theorem

One of the most successful ways for defining secure concurrent cryptographic tasks is via process emulation [1,2]. This definitional job boils down to the following: a process realizes a cryptographic task if and only if it emulates an ideal process that is known to realize such task. In this section, guided by the goal of defining secure functionalities, we detail the notion of emulation for the quantum process calculus defined in the previous section.

Let I be an ideal protocol that realizes (the honest part of) some secure protocol and P a process that implements the functionality specified by I. The overall goal is to show that P realizes, without flaws, (part of) the secure functionality specified by I. The goal is achieved if for any real adversary, say $(a : A)$, the process $P||(a : A)$ is computationally indistinguishable by the adversary a from the process $I||(a : B)$ for some ideal adversary $(a : B)$, where an ideal adversary is an adversary which cannot corrupt I and a real adversary is any local process for agent a. This property asserts that given a real adversary $(a : A)$, agent a cannot distinguish the information leaked by $P||(a : A)$ from the information leaked by the well behaved process $I||(a : B)$ for some ideal adversary $(a : B)$, and therefore, we infer that $P||(a : A)$ is also well behaved. This discussion leads to the concept of emulation with respect to a set of real adversaries $\mathcal{A}$ and ideal adversaries $\mathcal{B}$.

Definition 3.1 Let P and I be process terms and $\mathcal{A}$ and $\mathcal{B}$ sets of global processes where the only agent is the adversary a, then P *emulates* I with respect to $\mathcal{A}$ and $\mathcal{B}$ if and only if for all processes $(a : A) \in \mathcal{A}$ there exists a process $(a : B) \in \mathcal{B}$ such that $P||(a : A) \simeq I||(a : B)$. In such case we write $P \equiv^{a}_{\mathcal{A},\mathcal{B}} I$ and say that P is a secure implementation of I with respect to $\mathcal{A}$ and $\mathcal{B}$.

A desirable property of the emulation relation is the so called Composition Theorem. This result was first discussed informally for the classical secure computation setting in [12], and states the following: if P is a secure implementation of part I of an ideal protocol, R and J are two protocols which use the ideal protocol I as a component, and finally, R is a secure implementation of J, then R^{I}_{P} should be a secure implementation of J. This result is captured as follows:

Theorem 3.2 *Let P, I be processes, $R[\,]$ and $J[\,]$ contexts and $\mathcal{A}, \mathcal{B}$ sets of processes over agent a and $\mathcal{C}, \mathcal{D}$ sets of processes over agent b. If $R[I||(a : B)] \equiv^{b}_{\mathcal{C},\mathcal{D}} J[I||(a : B)]$ for any $(a : B) \in \mathcal{B}$ and $P \equiv^{a}_{\mathcal{A},\mathcal{B}} I$ then for any adversary $(a : A) \in \mathcal{A}$ there exists $(a : B) \in \mathcal{B}$ such that $R[Q||(a : A)] \equiv^{b}_{\mathcal{C},\mathcal{D}} J[I||(a : B)]$.*

Proof. Let $(a : A) \in \mathcal{A}$ and $(a : B) \in \mathcal{B}$ be such that $P||(a : A) \simeq I||(a : B)$. Now choose some $(b : C) \in \mathcal{C}$, clearly, $R[Q||(a : A)]||(c : C) \simeq R[I||(a : B)]||(c : C)$ since $\simeq$ is a congruence relation. Moreover, since $R[I||(a : B)] \equiv_{\mathcal{C},\mathcal{D}} J[I||(a : B)]$, there is a $(b : D) \in \mathcal{D}$ such that $R[I|(a : B)]|C \simeq J[I||(a : B)]||(b : D)$. Finally, by transitivity of $\simeq$, we have that $R[Q||(a : A)]||(b : C) \simeq J[I||(a : B)]||(b : D)$ and hence $R[Q||(a : A)] \equiv_{\mathcal{C},\mathcal{D}} J[I||(a : B)]$. □

Observe that ideal protocols are constituted by a honest part I and an ideal adversary $(a : B)$, and therefore are of the form $I||(a : B)$. This justifies why $R[I||(a : B)]$ was considered in the proposition above instead of $R[I]$. Moreover, adversaries for the functionality implemented by R and J might be different from those of I and Q, therefore, two pairs of sets of processes $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{A}$, $\mathcal{B}$ are required to model two kinds of adversaries.

4 Quantum Zero-Knowledge Proofs

An interactive proof is a two party protocol, where one agent is called the *prover* and the other is called the *verifier*. The main objective of the protocol is to let the prover convince the verifier of the validity of an assertion, however, this must be done in such a way that the prover cannot convince the verifier of the validity of some false assertion.

Any interactive proof system fulfills two properties: completeness and soundness. Completeness states that if the assertion the prover wants to convince the verifier is true, then the verifier should be convinced with probability one. On the other hand, soundness is fulfilled if the verifier cannot be convinced, up to a negligible probability, of a false assertion. Therefore, completeness and soundness allow the verifier to check whether the assertion of the prover is true or false.

Zero-knowledge is a property of the prover (strategy). Consider the following informal notion of (quantum) computational zero-knowledge strategy, which corresponds to the straightforward lifting to the quantum setting of the classical version:

Definition 4.1 A prover strategy S is said to be *quantum computational zero-knowledge* over a set L if and only if for every quantum polynomial-time verifier strategy, V there exists quantum polynomial-time algorithm M such that $(S, V)(l)$ is (quantum) computationally indistinguishable from $M(l)$ for all $l \in L$, where (S, V) denotes the output of the interaction between S and V.

The main application of zero-knowledge proof protocols in the cryptographic setting is in the context of a user U that has a secret and is supposed to perform some steps, depending on the secret. The problem is how can other users assure that U has carried out the correct steps without U disclosing its

secret. Zero-knowledge proof protocols (ZKP) can be used to satisfy these conflicting requirements.

Zero-knowledge essentially embodies that the verifier cannot gain more knowledge when interacting with the prover than by running alone a quantum polynomial time program (using the same input in both cases). That is, running a the verifier in parallel with the prover should be indistinguishable of some quantum polynomial time program.

Actually, the notion of (quantum computational) zero-knowledge proofs can be captured through emulation very easily. Assuming that a proof strategy $S(x)$ and verifier $V(x)$ are modelled as terms of the process algebra, it is actually possible to model the interaction between p and v by the process $(p : S)||(v : V)$. Denote by $\mathcal{L}^v(l)$ the set of all process terms for the verifier $(v : V)^x_l$, that is, any process term $(v : V)$ where the free variable x was replaced by the binary word l. We have the following characterization:

Proposition 4.2 *A process term $(p : S)$ denoting a proof strategy is computational zero-knowledge for L if and only if $(p : S)^x_l \equiv^v_{\mathcal{L}^v(l),\mathcal{L}^v(l)} 0$, for all $l \in L$.*

Proof (Sketch): *Notice that the ZKP resumes to impose that for all $(v : V)^x_l$ there is a process $(v : V')^x_l$ such that $(p : S)^x_l||(v : V)^x_l \simeq 0||(v : V')^x_l$. Since the semantics of a local process can be modelled by a QPM, and moreover $0||(v : V')^x_l$ can model any QPM, the characterization proposed in this proposition is equivalent to Definition 4.1.* □

So, a process $(p : S)$ models a quantum zero-knowledge strategy if, from the point of view of the verifier, it is impossible to distinguish the final result of the interaction with $(p : S)$ from the interaction with the 0 process. A clear corollary of Theorem 3.2 is that, quantum zero-knowledge is compositional.

It is simple to adapt the emulation approach to several other quantum security properties, like quantum secure computation, authentication and so on.

5 Conclusions

The contributions of this paper are multiple. First, we introduced a process algebra for specifying and reasoning about (quantum) security protocols. To restrict the computation power of the agents to quantum polynomial-time, we introduced the logarithm cost quantum random access machine, and incorporated it in the process language. Due to the special aspects of quantum information, qubits were assumed to be partitioned among agents, and the (quantum) computation of an agent was restricted to its own qubits.

Second, we defined observational equivalence and quantum computational indistinguishability for the process algebra at hand, and showed that the latter is a congruence relation. Moreover, we obtained a simple corollary of this

result: security properties defined via emulation are compositional.

Finally, we illustrated the definition of a security property via emulation with the concept of quantum zero-knowledge. It is however straightforward to adapt this approach to several other quantum security properties, like quantum secure computation.

Acknowledgement

The authors wish to express their gratitude to the regular participants in the QCI Seminar at CLC, specially to Amílcar Sernadas and João Rasga, who gave very useful feedback.

References

[1] Abadi, M. and A. D. Gordon, *A calculus for cryptographic protocols: The Spi Calculus*, Information and Computation **148** (1999), pp. 1–70, full version available as SRC Research Report 149, January 1998.

[2] Canetti, R., *Universally composable security: A new paradigm for cryptographic protocols*, in: *42nd IEEE Symposium on Foundations of Computer Science (FOCS)* (2001), pp. 136–145, full version available at IACR ePrint Archive, Report 2000/067.

[3] Cohen-Tannoudji, C., B. Diu and F. Laloë, "Quantum Mechanics," John Wiley, 1977.

[4] Cook, S. A. and R. A. Reckhow, *Time bounded random access machines*, Journal of Computer and System Sciences **7** (1973), pp. 354–375.

[5] Gay, S. J. and R. Nagarajan, *Communicating quantum processes*, in: J. Palsberg and M. Abadi, editors, *Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)* (2005), pp. 145–157.

[6] Knill, E., *Conventions for quantum pseudocode*, Technical Report LAUR-96-2724, Los Alamos National Laboratory (1996).

[7] Lincoln, P., J. Mitchell, M. Mitchell and A. Scedrov, *A probabilistic polynomial-time framework for protocol analysis*, in: M. Reiter, editor, *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS)* (1998), pp. 112–121.

[8] Martins, A. M., P. Mateus and A. Sernadas, *Minimization of quantum automata*, Technical report, CLC, Department of Mathematics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal (2005), submitted for Publication.

[9] Mateus, P., J. C. Mitchell and A. Scedrov, *Composition of cryptographic protocols in a probabilistic polynomial-time process calculus*, in: R. Amadio and

D. Lugiez, editors, *CONCUR 2003 - Concurrency Theory*, Lecture Notes in Computer Science **2761** (2003), pp. 327–349.

[10] Mateus, P. and A. Sernadas, *Reasoning about quantum systems*, in: J. J. Alferes and J. A. Leite, editors, *Proceedings of the 9th European Conference on Logics in Artificial Intelligence*, Lecture Notes in Artificial Intelligence **3229** (2004), pp. 239–251.

[11] Mateus, P. and A. Sernadas, *Weakly complete axiomatization of exogenous quantum propositional logic*, Technical report, CLC, Department of Mathematics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal (2005), submitted for Publication. http://arxiv.org/abs/math.LO/0503453.

[12] Micali, S. and P. Rogaway, *Secure computation*, in: J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, Lecture Notes in Computer Science **576** (1991), pp. 392–404.

[13] Mitchell, J. C., A. Ramanathan, A. Scedrov and V. Teague, *A probabilistic polynomial-time calculus for analysis of cryptographic protocols (preliminary report)*, Electronic Notes in Theoretical Computer Science **45** (2001), pp. 1–31.

[14] Yao, A. C., *Theory and applications of trapdoor functions*, in: *23rd IEEE Symposium on Foundations of Computer Science (FOCS)* (1982), pp. 80–91.