Transferring Proofs of Zero-Knowledge Systems with Quantum Correlations

P. Mateus, F. Moura and J. Rasga
SQIG-IT and IST, Portugal
{pmat,fmoura,jfr}@math.ist.utl.pt

October 17, 2006

Abstract

The use of quantum correlations to attack security protocols is an important research line deserving growing attention. An important class of cryptographic protocols used as building blocks for several other more complex protocols is zero-knowledge proof systems. One of the properties that zero-knowledge proof systems are assumed to satisfy is that it is impossible for the verifier to show to a third party that he has interacted with the prover (*impossibility of transferring proofs*). Herein, it is shown how Bell pairs, together with tamper-proofing, can be used to break the impossibility of transferring proofs for an important class of zero-knowledge proof systems.

Keywords: Quantum correlations, quantum attacks, zero-knowledge proof systems.

1 Introduction

One of the main motivations to build quantum computers is that they are able to attack widely used classical cryptosystems, like RSA and ElGammal. Besides involved quantum algorithms, it is also believed that simple quantum correlations may be used, in a distributed setting, to develop new attacks to security protocols. In this paper, it is shown how Bell pairs, together with tamper-proofing, can be used to transfer proofs of an important class of zero-knowledge proof systems.

Roughly speaking, a zero-knowledge proof system [9] is a two party protocol, between a verifier and a prover, in which the prover wants to show to the verifier that he knows some secret without conveying any information about this secret to the verifier (*zero-knowledge property*). Zero-knowledge proof systems are used in several protocol, like identification protocols [6], e-voting protocols [4], and e-money protocols [3]. While the zero-knowledge property is the most important security aspect these systems must fulfill, in the folklore another security property is assumed to hold in zero-knowledge systems. This property is known as *impossibility of transferring the proof* and states that at the end of the protocol, the verifier can not prove to a third party that he had interacted with the prover. So, the impossibility of transferring the proof is crucial for more complex protocols, like identification protocols and electronic elections, to satisfy several anonymity properties. In identification protocols the impossibility of transferring the proof allows the anonymity of the prover in the sense that the verifier can not show to a third party that a certain prover has identified to him. In election protocols it allows abstention to be anonymous.

To prove the impossibility of transferring the proof in a particular zero-knowledge proof system it is usually assumed that the verifier has complete control over his memory space and that he can rollback to a previous state whenever he needs. In practice this constitutes a very generous assumption. To be more realistic, the access to memory by the verifier may be restricted, for instance by using tamper-proof devices.

A lot of research was pursued in understanding the power of zero-knowledge under computational restrictions, see for instance [5]. However, the goal of this paper is to present the attack assuming that all parties have the usual polynomial power (could be quantum or probabilisitic), but where the agents can use tamper-proof devices.

In this paper we show how to attack the impossibility of transferring the proof for the Goldreich, Micali and Wigderson [7] system. The importance of this class stands from the fact that all zero-knowledge proof systems can be reduced to systems similar to the Goldreich, Micali and Wigderson one, and so the attack holds for all the protocols in this class.

A crucial feature of the attack is the use of tamper-proof Bell pairs. The tamper-proofing limits the verifier only to measure the first half of the Bell pair. Moreover, during the attack, the verifier can only perform two types of measurement over that half. These restrictions prevents the verifier to rollback the memory state of the Bell pairs after performing measurements, allowing a third party to check if the verifier was interacting with a particular prover, as shown below in the paper.

The paper is organized in the following way: in Section 2 some basic concepts and results about zero-knowledge proof systems are introduced and are illustrated by the Goldreich, Micali and Wigderson [7] system. In

Section 3 the notion of quantum automaton is presented and the attack using tamper-proof quantum automata is described. Finally, in Section 4, some conclusions are drawn.

2 Basic concepts

Zero-knowledge proof systems have became one of the most important concepts in computational security since their invention by Goldwasser, Micali and Rackoff in 1985 [9]. Before presenting zero-knowledge proof systems, we need to introduce several concepts from computational complexity, like for instance interactive proof-systems. We assume that the reader knows the basics of computational complexity mainly the probabilistic oriented concepts (for a good reference see [14]).

To properly define an interactive proof system it is necessary to consider a model allowing agents restricted to probabilistic polynomial time computations. There has been some discussion about the properties this model should satisfy and consequently about its expressiveness and behavior [15, 2, 11, 10]. In this work, we introduce zero-knowledge proof systems in the same way as Goldwasser, Micali and Rackoff first introduced them in 1985 [9]. However, contrarily to what is common in this area, we will not make use of (interactive) Turing machines in proofs or when presenting algorithms since that level of detail does not bring new insights as far as this paper is concerned. Instead we will refer to the Church-Markov-Turing Postulate[1]. In the sequel we denote the set $\{0,1\}$ by 2, and given a word $x \in 2^*$, we denote the *number of bits* in x (also called *the length* of x) by $|x|$.

Definition 1 An *interactive proof system* (V, P) is a protocol between two agents, the *verifier* V and the *prover* P, such that given an input $x \in 2^*$:

- Each agent has access to: (i) a private oracle that returns a (uniformly) random bit; (ii) the input x (only for reading); (iii) and a private memory.
- The protocol consists of several rounds and each agent is alternately active in each round. When active, each agent performs a computation and after that, either it sends a message to the other agent or the protocol halts.

[1]The Church-Markov-Turing Postulate asserts that all the "reasonable" (classical) computational models are equivalent and inter-reducible in polynomial time (note that this reduction is not valid between classical and quantum models).

- The computation of each agent depends only of the shared input x and of the messages received meanwhile from the other agent. The total time spent by the verifier V is polynomial in $|x|$.
- When the protocol halts, V outputs a value (denoted by $(V, P)(x)$) and, either accepts x (denoted by $(V, P)(x)_{\text{yes}}$), or rejects x.

In the following, given an interactive proof system (V, P), the probability of the verifier to accept the input x is denoted by $\text{Prob}((V, P)(x)_{\text{yes}})$. Note that what makes the computation of the verifier to be probabilistic is the fact that the oracle returns random bits. Moreover, observe that in an interactive proof system only the verifier is restricted to do polynomial time computations. There is no such restriction for the prover. So, for instance, given any NP language it is not difficult to consider an interactive proof system in which the verifier accepts exactly the elements of that language, as illustrated in the next example.

Example 2 (Interactive proof system for SAT) The verifier V and the prover P share a propositional formula φ. The protocol runs as follows:

1. If the formula φ is satisfiable, then P sends to V a valuation v that satisfies φ (note that the prover is not bounded to polynomial-time computations and therefore he can find this v). If the formula is not satisfiable, then P sends a random valuation to V.
2. When V receives a valuation v, it verifies whether the valuation satisfies φ. If this is the case it accepts φ, otherwise it rejects φ. In both cases, the output of the verifier, that is, the value $(V, P)(\varphi)$, is the valuation v received from the prover.

Note that in Example 2, if $\varphi \in \text{SAT}$ then the verifier accepts φ with probability 1, and if $\varphi \notin \text{SAT}$ then the verifier accepts v with probability 0. An interactive proof system for a language is defined in a more lax way by not imposing the probabilities to be 0 or 1.

Definition 3 A *binary language $L \subseteq 2^*$ has an interactive proof system* if there exists an interactive proof system (V, P) such that:

1. if $x \in L$ then $\text{Prob}((V, P)(x)_{\text{yes}}) > \frac{2}{3}$;
2. if $x \notin L$ then $\text{Prob}((V, P')(x)_{\text{yes}}) < \frac{1}{3}$ for any prover P'.

The system (V, P) is said to be an *interactive proof system for L*.

One of the most significant results in the area of computational complexity concerning interactive proof systems was obtained by Shamir and states that PSPACE coincides with the class of languages for which there is an interactive proof system [16]. From a security point of view the concept of interactive proof system is much more useful when enriched with the notion of zero knowledge. Informally, an interactive proof system is zero knowledge if the verifier does not get any additional information by interacting with the prover. In the following, we will see how this additional property is important to obtain certain security goals. Prior to define what is zero knowledge we need to introduce the concept of *computationally indistinguishability.*

Definition 4 Two probabilistic algorithms A_1 and A_2 are *computationally indistinguishable* if, for any probabilistic polynomial-time decision algorithm T, polynomial p and input x large enough

$$|\mathrm{Prob}(T(A_1(x)) = 1) - \mathrm{Prob}(T(A_2(x)) = 1)| \leq \frac{1}{p(|x|)}.$$

The notion of *computationally indistinguishability* is important in cryptography to define several security concepts, as for instance, zero-knowledge proof systems.

Definition 5 A *binary language $L \subseteq 2^*$ has a zero-knowledge proof system* if L has an interactive proof system (V, P) such that, for any verifier V' there exists a probabilistic polynomial-time algorithm $S_{V'}$ such that, $(V', P)(x)$ is computationally indistinguishable from $S_{V'}(x)$ for any $x \in L$ large enough.

In what constitutes one of the fundamental results about zero-knowledge proof systems, in [7] it is shown that there is a zero-knowledge proof system for any language in NP provided that there is a non-uniform and indistinguishable cipher scheme. Zero-knowledge proof systems for languages in NP that are not believed to be in P can be employed to build useful security systems like for instance, identification protocols [6], e-voting protocols [4] and e-money protocols [3]. Informally theses systems allow the prover to show to the verifier that he knows a secret (a witness of an instance of the NP problem that is being considered) without revealing it. We now (re)define zero-knowledge proof systems from a cryptographic point of view.

Definition 6 Let L be a language in NP that is believed not to be in P, $x \in L$ and s a witness[2] of x. Let V (verifier) and P (prover) be two agents. P states to know a witness s of x, and V wants to check whether P really knows s. A zero-knowledge proof system for x and s is an interactive proof system (V, P) such that the following conditions hold:

- *Polynomial-time bound* – the overall time complexity of both V and P is probabilistic polynomial in the size in bits of x.
- *Soundness* – if P knows a witness s then V accepts that P has a witness s of x except with a negligible probability[3];
- *Completeness* – if P does not know a witness s then V accepts that P has a witness s of x with negligible probability;
- *Zero-knowledge* – during the protocol V does not get any new information about the secret witness s of x.
- *Impossibility of transferring the proof* – the verifier V cannot show to a third party that he was interacting with P.

Note that due to the polynomial-time bound condition in Definition 6 an agent in a zero-knowledge proof system can only exchange a polynomial (in $|x|$) number of messages, and moreover, the size of each message is also polynomially bounded over $|x|$. Furthermore, due to the zero-knowledge condition, a verifier does not obtain any new information about s, that is, it must be possible to simulate, in a computationally indistinguishable way, the interaction between the prover and an arbitrary verifier using a probabilistic polynomial-time simulator with input x.

It is worthwhile to briefly discuss and motivate each of the conditions presented in the Definition 6. The polynomial-time bound corresponds to the usual restriction in cryptography that all agents can only perform efficient computation. Soundness and completeness impose that the verifier should accept that the prover has the secret (except with a negligible error) if and only if the prover actually knows the secret (that is, a witness s of the

[2]Recall that a language $L \subseteq 2^*$ is in NP if there exists an polynomial-time decision algorithm A with two inputs, x and s, such that: (i) if $x \in L$ there exists a witness s such that $A(x, s) = 1$; (ii) if $x \notin L$ then for any witness s, $A(x, s) = 0$.

[3]A family of Bernoulli random variables $\{X_n\}_{n\in\mathbb{N}}$ is said to have negligible probability if, for any polynomial p and natural number n large enough, $\text{Prob}(X_n = 1) < \frac{1}{p(n)}$. In the context of soundness of zero-knowledge proof systems, the random variable X_n to consider takes value 1 if V does not accept that P has a witness s of x with size n.

instance x of the NP problem). The zero-knowledge property establishes that the verifier can not acquire additional information about the secret s during the run of the protocol. In particular, if the goal of the proof system is to identify the prover, i.e., the secret s identifies the prover, the zero-knowledge property guarantees that the verifier can not, after interacting with the prover, gain information to impersonate the prover. Finally, the impossibility of transferring the proof is an anonymity property. For instance, in identification protocols the impossibility of transferring the proof allows the anonymity of the prover in the sense that the verifier can not show to a third party that a certain prover has identified to him. We will see that this property can be attacked when tamper-proof (quantum) memory is available.

One of the simplest examples of a zero-knowledge proof system according to Definition 6 is the Goldreich, Micali e Wigderson system [7] which is based on the conjecture that deciding if two graphs are isomorphic is not in P although it is in NP.

Example 7 (Goldreich, Micali and Wigderson system) Suppose that both the verifier V and the prover P know two graphs G_0 and G_1 with n nodes. P says that he knows an isomorphism $\sigma : G_1 \to G_0$ (a witness that G_0 and G_1 are isomorphic). The protocol runs as follows:

1. P generates a random isomorphism $\pi : G_0 \to H$ and sends H to V;
2. V generates a random bit $b \in \{0, 1\}$ and sends it to P;
3. P sends the isomorphism $\tau = \pi \circ \sigma^b$ to V;
4. V checks if $\tau(G_b) = H$.

If the condition in Step 4 is not fulfilled, then V does not accept that P knows an isomorphism between G_0 and G_1, otherwise, Steps 1 to 4 are repeated n times, and the verifier only believes that P knows an isomorphism if in all these repetitions the condition in Step 4 holds.

It is interesting to observe that the problem chosen in Example 7, the graph isomorphism problem, is an NP problem that is not known to be NP-complete. In fact it would be expected that NP-complete problems were in general more secure than NP problems since there is a polynomial time reduction from all NP problems to NP-complete problems. Nevertheless NP-complete problems have not been widely employed in cryptography as opposed to cryptographic systems based on problems as the factorization and discrete logarithm (that belong to NP $\cap$ co-NP). The reason is that although NP-complete problems are very hard in the worst case, for some of them it is possible to find a witness for a random instance with high probability. Since the security of cryptographic systems rely on how hard it is to find those witnesses, NP problems believed to be sound to this attack, like graph isomorphism, are preferred.

Observe that the several steps of Example 7 constitute a typical iteration of a zero-knowledge proof system. The first step, designated by *commitment*, is when the prover sends the first message to the verifier. In the second step, called *challenge*, the verifier confronts the prover by typically sending it a bit. Finally, the third step, called the *decommitment*, is when the prover will have to send a message coherent with his commitment and with the challenge sent by the verifier. It can be shown that all the zero-knowledge proof systems for NP problems are reducible to a protocol with this structure [7]. The analysis and the attack presented in Section 3 rely on this organization of the protocol (commitment, challenge and decommitment). The results can be straightforwardly generalized to any system with this organization although they are presented with respect to Example 7. It is interesting also to note that graph isomorphism seems not to be in BQP, that is, it seems to be hard to attack graph isomorphism using quantum computers with the techniques known nowadays [13].

The Goldreich, Micali e Wigderson zero-knowledge proof systems described in Example 7 enjoys the following property, as shown in [7].

Theorem 8 (Goldreich, Micali e Wigderson) The protocol of Example 7 is a zero-knowledge proof system.

The technique employed to prove that the Goldreich, Micali and Wigderson system is a zero-knowledge interactive proof system for classical adversaries has become a widely used method to show the security of several protocols [8] and a well known technique to define security features [1, 15, 11, 10]. This technique is called *simulation paradigm*. This paradigm consists in showing that a real adversary of the protocol can be simulated by an ideal adversary, that is, an adversary that can not attack the system. For zero-knowledge proof systems the proof consists in showing that any dishonest verifier can be simulated by a process that does not interact with the prover (this process is the ideal verifier). Since the proof of Theorem 8 is important to understand the results of the paper we decided to present it here.

Proof of Theorem 8: Let V' be a verifier that might be dishonest, that is, a verifier that does not follow the protocol specified in Example 7. The goal is to show that it is possible to simulate the interaction between this verifier V' with an honest prover P, without communication with the prover. First, observe that the purpose of the simulator is to generate n tuples (H, c, τ) where H is randomly generated (since the prover is honest) and c is a bit generated accordingly by V'. Observe as well that the simulator has access to the code of V'. In these conditions, a verifier V' is a family of probabilistic polynomial-time algorithms $\{V'_k\}_{k\in 1\dots n}$ where V'_k corresponds to the algorithm that V' uses to choose c in repetition k.

The algorithm V'_k receives a graph H and it may use some auxiliary data $w_k \in 2^*$ that he computed in previous repetitions. We assume that the initial data V' has about s is encoded in the string w_1. We now simulate V'_k as follows:

1. The simulator chooses $b \in \{0, 1\}$ randomly (with uniform distribution);

2. The simulator generates a random isomorphism $\tau : G_b \to H$ (with uniform distribution);

3. The simulator applies $V'_k(H, w_k)$ and verifies if the bit c computed by V'_k is b:

   (a) If $c = b$ then the simulation of repetition k is successful, since the tuple generated, (H, c, τ), has the same distribution as one created by interacting with the prover as discussed below. The auxiliary computation done by V'_k, which can be used in the next repetition, is stored in w_{k+1};

   (b) If $c \neq b$ then the simulator jumps to Step 1.

Observe that the probability of $b = c$ is always $\frac{1}{2}$ (regardless of the computation of c by V'_k), since b was chosen uniformly. In other words, one expects that the simulator will have success in 2 trials. The simulator will not have success in n trials with probability $\frac{1}{2^n}$, which is negligible.

Finally, it is straightforward to show that the sequence of tuples (H, c, τ) generated by the simulator has precisely the same distribution of the tuples generated by an interaction between V' and P. For this reasons, these sequences are computationally indistinguishable. $\square$

The simulation paradigm technique ensures the impossibility of transference of proof [7], that is, the impossibility of tracing the interaction between the verifier and the prover. It is worthwhile to discuss this result since we will show how to attack it in a realistic context. The reason why the impossibility of transference of proof holds is the following. First, note that if the verifier wants to show to a third party, say Eve, that he has been interacting with the prover then he can send her the trace of the interaction, that is, the sequence of tuples (H, c, τ). Nevertheless, as it was shown before, regardless of the behavior of the verifier, it is always possible to generate, without any communication with the prover, a sequence of tuples computationally indistinguishable from the trace of the interaction. So, if Eve has only access to the trace of the interaction, she may not be persuaded that the verifier did in fact interact with the prover, because that sequence of tuples may have been generated by the verifier without communicating with the prover. For this point to hold the verifier must have complete control over the state of the computation, that is, when the simulator used in the proof of Theorem 8 fails in Step 3, it must jump to Step 1 and this trial should be disregarded. This scenario is too generous in what concerns the memory model of the verifier. If, in the beginning of the protocol, Eve gives a tamper-proof machine to the verifier such that he can not put the machine in an arbitrary state, then it is not possible to apply the simulation paradigm. It is also interesting to note that if quantum memory is shared by Eve and the verifier it seems hard to apply the simulation paradigm. The reason for this, is that if Eve and the verifier shared entangled memory and if measurements are performed by a dishonest verifier it seems hard to rollback. Therefore, the validity of the simulation paradigm in the quantum setting was an open problem. However, in [17], Watrous showed how to solve this problem.

Theorem 9 (Watrous) The Goldreich, Micali and Wigderson system is zero-knowledge against quantum polynomial-time adversaries.

Again, in the proof of Theorem 9, it is fundamental to assume that the verifiers has complete control over his private memory. As we will show in the next section in the context of the Goldreich, Micali and Wigderson system, if the memory of the verifier is tamper-proof, the verifier is able to show to Eve that he was interacting with the prover.

3 Quantum automata and transference of proofs

In this section we will show that using *quantum tamper-proof machines*, that is, machines for which the verifier has no possibility of arbitrarily changing its internal state, it is possible to trace the interaction between the verifier and the prover. To this end, it is useful to describe a tamper-proof device as a quantum automaton. We generalize the notion of quantum automaton introduced in [12] by extending the set of possible outputs and allowing measurements as inputs.

Definition 10 A *quantum automaton* is a tuple $Q = (I, \Gamma, O, H, |\psi_0\rangle, U, A)$ where:

- I is a finite set of *transition symbols*;
- Γ is a finite set of *observation symbols*;
- $O \subseteq \mathbb{R}$ is a finite set of *output symbols*;
- H is finite dimensional Hilbert space called the *state space*;
- $|\psi_0\rangle \in H$ is a unit vector in H called the *initial state*;
- $U = \{U_i\}_{i \in I}$ is a family of unitary transformations in H called *family of transitions*;
- $A = \{A_\gamma\}_{\gamma \in \Gamma}$ is a family of observables over H called *family of observables* such that the spectrum of A_γ is contained in O for all $\gamma \in \Gamma$.

The dynamics of a quantum automaton is the following. Two types of inputs are possible: transition inputs I and observation inputs Γ. Transition inputs are associated to unitary transformations $\{U_i\}_{i \in I}$ while observation inputs are associated to observables $\{A_\gamma\}_{\gamma \in \Gamma}$. The automaton is a reactive machine, that changes state when an input is read and outputs a value when an observation input is read. The set of outputs O is a (finite) subset of the real numbers containing the spectrum of all observables A_γ of the automaton. The state of the automaton is a unit vector over the state space H, which is a space where all unitary transformations and observables are defined. The automaton starts at the initial state $|\psi_0\rangle$. The state of the automaton evolves deterministically as transition inputs are read and probabilistically when the inputs are observations, respecting the postulates of quantum mechanics. That is, when a transition input i is read, the state $|\psi\rangle$ of the automaton evolves to $U_i|\psi\rangle$ without outputting any value. If an observation input γ is read, the state $|\psi\rangle$ evolves to $P_o|\psi\rangle / \|P_o|\psi\rangle\|$ with probability $\|P_o|\psi\rangle\|$, for any eigenvalue $o \in O$ of A_γ, where P_o is the projection onto the eigenspace of A_γ associated to o. In this case, the automaton outputs o.

To illustrate the concept we describe a quantum automaton that generates pure random bits.

Example 11 (Pure random bit generator) Consider the quantum automaton $(I, \Gamma, O, H, |\psi_0\rangle, U, A)$ where:

- $I = \{t\}$;
- $\Gamma = \{o\}$;
- $O = \{-1, 1\}$;
- $H = \mathbb{C}^2$;
- $|\psi_0\rangle = |0\rangle$;
- U_t is the Hadamard transformation;
- A_o is the observable $-|0\rangle\langle 0| + |1\rangle\langle 1|$.

Recall that the Hadamard transformation is the unitary transformation that maps $|0\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Each time the sequence $t\,o$ of inputs is read the automaton generates a random bit. The bit generation works as follows. Initially, the input t associated with the Hadamard transformation is read, setting the state to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Then the observation input o is read, outputting -1 or 1 with probability $\frac{1}{2}$ each, and the state evolves to $|0\rangle$ or $|1\rangle$. For obtaining more random bits, another sequence $t\,o$ should be read. Observe that for the next random bits, when t is read, the state evolves to $\frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$. Nevertheless, when any of these states are observed by A_o the output is -1 or 1 with probability $\frac{1}{2}$.

We are now able to define the tamper-proof machine that will perform the attack to the Goldreich, Micali and Wigderson system. Before, we introduce the *computational observable* as $-|0\rangle\langle 0| + |1\rangle\langle 1|$ and the *diagonal observable* as $-|\nearrow\rangle\langle\nearrow| + |\searrow\rangle\langle\searrow|$ where $|\nearrow\rangle$ is $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|\searrow\rangle$ is $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

Definition 12 A *quantum tamper-proof automaton for a graph G with n nodes* is the quantum automaton $Q_G = (I, \Gamma, O, H, |\psi_0\rangle, U, A)$ where:

- $I = \emptyset$;
- $\Gamma = \{c, d\} \times \{1, 2\} \times \{1 \dots n\}^2$;
- $O = \{-1, 1\}$;
- $H = \otimes_{k=1}^{n} \otimes_{j=1}^{n} (\mathbb{C}^2 \otimes \mathbb{C}^2)$ (space constituted by n^2 pairs of *qubits*);
- $|\psi_0\rangle = \otimes_{k=1}^{n} \otimes_{j=1}^{n} (\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$);
- The family of observables A is such that:
  - $A_{(c,1,(k,j))}$ corresponds to the computational observable over the first *qubit* of the $(k \times j)$-th pair with $k, j \in 1 \dots n$;
  - $A_{(d,1,(k,j))}$ corresponds to the diagonal observable over the first *qubit* of the $(k \times j)$-th pair with $k, j \in 1 \dots n$;
  - $A_{(c,2,(k,j))}$ corresponds to the computational observable over the second *qubit* of the $(k \times j)$-th pair with $k, j \in 1 \dots n$;
  - $A_{(d,2,(k,j))}$ corresponds to the diagonal observable over the second *qubit* of the $(k \times j)$-th pair with $k, j \in 1 \dots n$.

To perform the attack the verifier has to show to Eve that he interacted with the prover after the interaction ended[4].

In order to trace the interaction with a prover whose secret is an isomorphism between G_0 and G_1, Eve should set up a quantum tamper-proof automaton for G_0 with n nodes. This quantum automaton consists of n^2 pairs of qubits, all of them initially set at the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|\nearrow\nearrow\rangle + |\searrow\searrow\rangle)$. The automaton only accepts observation inputs, and the observables are exactly the computational and diagonal observables for each qubit (so in total there are $2 \times 2 \times n^2$ observables). We index the n^2 Bell pairs by (k, j) with $k, j \in 1 \dots n$. To understand the notation of observation inputs, the symbol $(c, 1, (k, j))$ is associated to the **c**omputational observable of the **1**st half of the (k, j) Bell pair, and similarly, $(d, 2, (k, j))$ is associated to the **d**iagonal observable of the **2**nd half of the (k, j) Bell pair.

We are now able to describe the protocol between Eve, the verifier and the honest prover, which will be used to obtain an evidence of the interaction between the verifier and the prover (this evidence is called a *trace*

[4]Note that it is always assumed that the verifier can not communicate with Eve during the interaction with the prover. Otherwise Eve would play the role of the verifier, and therefore could check herself that the prover is interacting. The attack consists of the verifier holding an evidence that he interacted with the prover after this interaction ended.

of the interaction). To this end, Eve and the verifier have to be colluded (nevertheless, notice that they do not trust each other otherwise Eve would always believe in the verifier), in the sense that Eve will give the quantum tamper-proof automaton to the verifier, which will be used to store the trace. The tamper-proofing consists on the verifier being restricted to apply only the observables $A_{(c,1,(k,j))}$ and $A_{(d,1,(k,j))}$.

Definition 13 *Protocol to trace the Goldreich, Micali and Wigderson system* for a prover knowing an isomorphism between G_0 and G_1 with n nodes.

1. Eve and the verifier agree with a hash function
$$h : \mathcal{G}_n \to \{c, d\}^n$$
where $\mathcal{G}_n$ is the set of all graphs with n nodes.

2. Eve gives the sealed automaton Q_{G_0} to V. The automaton is sealed such that V can only execute the observation operators $A_{(c,1,(k,j))}$ and $A_{(d,1,(k,j))}$.

3. V starts the interaction with P similarly to Example 7. For each repetition k:

   (a) In Step 1, V receives the graph H_k sent by P;

   (b) In Step 2, V computes $h(H_k) = e_{(k,1)} \dots e_{(k,n)}$ and applies the observables $A_{(e_{(k,j)},1,(k,j))}$ for all $j \in 1 \dots n$ obtaining the sequence of measurements $o_{(k,1)} \dots o_{(k,n)}$ with $o_{(k,j)} \in \{-1, 1\}$. Next, V computes $b_k = \oplus_{j=1}^{n}(1 + o_{(k,j)})/2$ and sends $b_k \in \{0, 1\}$ to P, where $\oplus$ denotes the xor operator.

   (c) In Step 3, V receives the isomorphism τ_k sent from P (and checks if $\tau_k(G_{b_k}) = H_k$).

4. When the protocol ends, V sends the machine to Eve together with the sequence $w = (H_1, b_1) \dots (H_n, b_n)$ of pairs (graph, bit) and the sequence $m = \tau_1 \dots \tau_n$ of isomorphisms obtained in each repetition of the protocol.

5. For each repetition $k = 1, \dots n$, Eve will do as following:

   (a) She starts by computing $h(H_k) = e_{(k,1)} \dots e_{(k,n)}$ and applies the observables $A_{(e_{(k,j)},1,(k,j))}$ and $A_{(e_{(k,j)},2,(k,j))}$ for all $j \in 1 \dots n$ obtaining the sequences $o_{(k,1)} \dots o_{(k,n)}$ and $r_{(k,1)} \dots r_{(k,n)}$, respectively, with $o_{(k,j)}, r_{(k,j)} \in \{-1, 1\}$.

(b) Then, Eve checks if $o_{(k,j)} = r_{(k,j)}$ for all $j \in 1 \ldots n$ and if b_k (received from V) is equal to $\oplus_{j=1}^{n}(1 + r_{(k,j)})/2$.

(c) Finally, Eve checks whether $\tau_k(G_{b_k}) = H_k$.

If for all repetitions the checks performed at Steps 5 b) and 5 c) hold then Eve believes that V was interacting with P.

The protocol above uses two features of quantum systems that assist to trace the interaction between the verifier and the prover.

The first feature is the fact that the outcome of a measurement of a Bell pair is a uniform Bernoulli random variable. Therefore, the measurement performed by the verifier in Step 3 b) will provide a Bernoulli random variable for b_k, that neither Eve nor the verifier will be able to predict. Note that if no randomness was employed, and the sequence $b_1 \ldots b_n$ was predetermined by Eve, she could impersonate the prover while interacting with the verifier (jeopardizing completeness as presented in Definition 6). On the other hand, if the verifier has control over the sequence $b_1 \ldots b_n$, there is no way Eve can distinguish a simulation performed solely by the verifier with the interaction between the verifier and the prover. So, it is fundamental that the output of the machine is random, and that neither Eve nor the verifier have control over the sequence $b_1 \ldots b_n$. In order for the verifier to be sure that the machine states are Bell pairs we can assume that a validity test is performed together with Eve when the quantum tamper-proof device is given to the verifier.

The second property of quantum systems that is used is entanglement. Bell pairs have the nice property that a measurement over one qubit will collapse both qubits to the same state. Since Eve does not trust the verifier, she uses Bell pairs to be sure that the measurements performed by the verifier in one half of the pairs are consistent with the measurements she performs in Step 5 a) over the second half. We can assume the second half of the qubits are always in possession of Eve, and that when the machine is given to the verifier it only contains the first half of the Bell pairs.

Finally, we show that the attack is possible.

Theorem 14 If the verifier can only change the state of the automata Q_{G_0} by executing $A_{(c,1,(k,j))}$ and $A_{(d,1,(k,j))}$, then the protocol presented in Definition 13 traces the interaction between the verifier and the prover for the Goldreich, Micali e Wigderson system.

Proof: Assume that V wants to convince Eve that he was interacting with P without interacting with him.

By Step 5 of the protocol, V has to send to Eve a sequence $w = (H_1, b_1) \dots (H_n, b_n)$. We split the repetitions in two classes, those for which the verifier did not perform all measurements as described in Step 3 b) and those for which he does all the measurements.

Suppose that at repetition k the verifier did not perform in Q_{G_0} at least one measurement $A_{(e_{(k,j)},1,(k,j))}$ (as indicated in Step 3 b) for $h(H_k) = e_{(k,1)} \dots e_{(k,n)}$ and $k \in 1, \dots n$. Then, this measurement will be performed by Eve during Step 5 a), and in that case, the probability of $\oplus_{j=1}^{n}(1 + o_j)/2$ being equal to b_k is $\frac{1}{2}$.

Thus, if there are m repetitions for which at least one measurement $A_{(e_{(k,j)},1,(k,j))}$ was not performed, the probability of the verifier cheating Eve and not being detected is at most $\frac{1}{2^m}$.

Now, we deal with the remaining $n - m$ repetitions. We will split these repetitions in two, those in which the verifier did not perform the measurements accordingly to $h(H_k)$ and those where the measurements were made accordingly to $h(H_k)$. For the first case, it is straightforward to see that the probability of Eve detecting that V has cheated her, for each repetition, is $\frac{1}{2}$. Assuming that there are t of these repetitions, the probability of V not being detected is $\frac{1}{2^t}$. For the last case, we assume that there are $n - m - t$ repetitions. For each such repetition k, V obtains a sequence $o_{(k,1)} \dots o_{(k,n)}$. Since $o_{(k,j)} \in \{-1, 1\}$ is a uniform Bernoulli random variable and $b_k = \oplus_{j=1}^{n}(1+o_{(k,j)})/2$, we have that $\mathrm{Prob}(b_k = 1) = \frac{1}{2}$. Thus, assuming that V does not know an isomorphism between G_0 and G_1, the probability of V choosing H_k such that he can find an isomorphism with G_{b_k} is $\frac{1}{2^{n-m-t}}$. Considering all the $n - m - t$ graphs, the number of graphs N for which the verifier will not be able to construct the isomorphism is a random variable with binomial distribution with parameters $\frac{1}{2}$ and $n - m - t$. For the verifier not to be detected while cheating, he has to change the N graphs in the trace to send to Eve. Since the probability of finding collisions for h is negligible, we assume that the image by h of each replaced graph differs at least one bit from the image by h of the original graph. In these conditions, the probability of Eve not detecting that V is substituting the N graphs in Step 5 is $\frac{1}{2^N}$.

Combining the results of all repetitions, the probability of V not being detected is given by $\frac{1}{2^{N+m+t}}$. The expected value of $N + m + t$ is minimum when $m = t = 0$ and takes the value $n/2$, leading to probability $\frac{1}{2^{n/2}}$ that is negligible in n. □

4 Conclusions

It is important to notice that the quantum tamper-proof device can be almost simulated by a classical tamper-proof machine, but several problems arise. Firstly, randomness has to be replaced by pseudo-randomness, and therefore, if either Eve or the verifier know the seed of the pseudo-random generator they will be able to predict the sequence $b_1 \dots b_n$, which will not be acceptable by the other part. Secondly, to mimic the entanglement one requires a write-once/read-many memory over the tamper-proof devices, which is not the usual setting. Although eventually not impossible, it is not straightforward to come up with a classical machine that fulfills all these requirements.

Another important observation comes from the fact that we only show how to attack the Goldreich, Micali e Wigderson system. However, it is more or less obvious to adapt the attack to zero-knowledge proof systems based on the commitment, challenge, response structure, which account for the majority of the zero-knowledge proof systems. Moreover, it has been shown that all zero-knowledge proof systems for NP problems are reducible to systems with these structure [7].

It is not obvious how to adapt the Goldreich, Micali e Wigderson system, presented in Example 7, in order to make it robust to the attack herein reported. A naive idea is to consider a variant of that protocol where, just after Step 2, the verifier and the prover toss a coin and disregard the repetition if, for instance, tails come out. This new step requires an additional protocol to toss coins. However, this is not sufficient because the coin tossing protocol can also be encoded in a tamper-proof device and Eve would know when the repetition should be discarded or not. Therefore, it is still an open question how to modify the protocol in order to make it robust to tamper-proof based attacks.

Acknowledgments

This work was partially supported by FCT and EU FEDER, namely via CLC POCTI (Research Unit 1-601), QuantLog project POCI/MAT/55796/2004 and within the recent QSec initiative of SQIG-IT. F. Moura was also supported by the FCT and EU FEDER postdoctoral fellowship BPD/14064/2003.

References

[1] R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[2] R. Canetti, I. Damagrd, S. Dziembowski, Y. Ishai, and T. Malkin. On adaptive vs. non-adaptive security of multiparty protocols. In *EUROCRYPT'01*, pages 262–279. Springer, 2001.

[3] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *Crypto'88*, volume 403 of *LNCS*, pages 319–327. Springer, 1990.

[4] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In W. Fumy, editor, *EuroCrypt'97*, volume 1233 of *LNCS*, pages 103–118. Springer, 1997.

[5] C. Dwork and L. Stockmeyer. Finite state verifiers i: the power of interaction. *J. ACM*, 39(4):800–828, 1992.

[6] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, 1987.

[7] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):690–728, 1991.

[8] S. Goldwasser. Multi party computations: past and present. In *PODC'97*, pages 1–6. ACM Press, 1997.

[9] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, 18(1):186–208, 1989.

[10] P. Mateus, J. Mitchell, and A. Scedrov. Composition of cryptographic protocols in a probabilistic polynomial-time process calculus. In R. Amadio and D. Lugiez, editors, *CONCUR'03*, volume 2761 of *LNCS*, pages 327–349. Springer-Verlag, 2003.

[11] J. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for analysis of cryptographic protocols. *Theoretical Computer Science*, to appear.

[12] C. Moore and J. P. Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science*, 206:275–306, 2000.

[13] C. Moore, A. Russell, and J. Schulman. The symmetric group defies strong Fourier sampling. In *FOCS'05*, pages 479–490, 2005.

[14] C. H. Papadimitriou. *Complexity Theory.* Addison-Wesley, 1994.

[15] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *CCS'00*, pages 245–254, 2000.

[16] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[17] J. Watrous. Zero-knowledge against quantum attacks. In *STOC'06*, pages 296–305. ACM Press, 2006.