Automated Verification of Equivalence on Quantum Cryptographic Protocols

Takahiro Kubota[1], Yoshihiko Kakutani[1], Go Kato[2],
Yasuhito Kawano[2] and Hideki Sakurada[2]

[1] The University of Tokyo, Tokyo, Japan
{takahiro.k11_30,kakutani}@is.s.u-tokyo.ac.jp
[2] NTT Communication Science Laboratories, Kanagawa, Japan
{kato.go,kawano.yasuhito,sakurada.hideki}@lab.ntt.co.jp

Abstract

It is recognized that security verification of cryptographic protocols tends to be difficult and in fact, some flaws on protocol designs or security proofs were found after they had been presented. The use of formal methods is a way to deal with such complexity. Especially, process calculi are suitable to describe parallel systems. Bisimilarity, which denotes that two processes behave indistinguishably from the outside, is a key notion in process calculi. However, by-hand verification of bisimilarity is often tedious when the processes have many long branches in their transitions. We developed a software tool to automatically verify bisimulation relation in a quantum process calculus qCCS and applied it to Shor and Preskill's security proof of BB84. The tool succeeds to verify the equivalence of BB84 and an EDP-based protocol, which are discussed in their proof.

1 Introduction

Security proofs of cryptographic protocols tend to be complex and difficult to verify. This fact has been recognized in the classical cryptography and there is a line of researches to deal with the complexity of verification [1, 2]. Since by-hand proofs are prone to human error and time consumption, efforts have been put into automating security proofs and verification in the classical cryptography. Security proofs are also complex in the quantum cryptography, where we must also consider attacks using entanglements. The first security proof of BB84 quantum key distribution (QKD) protocol by Mayers [3] is about 50 pages long. After that paper, researchers have been seeking simpler proofs [4, 5]. Since QKD is one of the closest application to practice in the quantum information field, it is important to examine QKD systems' security formally and make it machine-checkable.

There are several formal frameworks for quantum protocols. Feng et al. developed a process calculus qCCS [6]. In this calculus, a protocol is formalized as a configuration. A configuration is a pair of a process and a quantum state corresponding to quantum variables. Weak open bisimulation relation on qCCS configurations is defined. Weakly open bisimilar configurations have the same transitions up to internal ones and reveal the identical quantum states to the outsider (adversary) in each step. The relation is proven to be closed by parallel composition of processes. A use case of a process calculus for formal verification is to prove weak bisimilarity of implementation and specification described as configurations. qCCS has been applied to quantum teleportation, superdense coding and BB84 protocols.

To extend application of formal methods to security proofs, we applied qCCS to Shor and Preskill's security proof of BB84 [4] in the previous paper [7]. In Shor and Preskill's security proof, security of BB84 is proven to be equivalent to that of another protocol based on an entanglement distillation protocol (EDP), and then the latter is proven to be secure. We

formalized BB84 and the EDP-based protocol as configurations and proved their bisimilarity by hand. However, by-hand verification of bisimilarity is often tedious when configurations have many long branches in their transitions. In this paper, we present a software tool to verify bisimilarity of given two qCCS configurations. There are two main contributions of our work.

First, we implemented a software tool that formally verifies bisimilarity of qCCS configurations without recursive structures. A protocol possibly takes security parameters for various purposes. As for BB84, it takes two parameters: one determines the length of qubits which Alice sends to Bob in the first quantum communication, and the other is for tolerated error-rate in the eavesdropping detection step. Such parameters are passed to the verifier as symbols and are interpreted appropriately. In our verifier's framework, quantum states and operations are described as symbols. Since attacks from the adversary are also treated as symbols, our verifier can deal with unspecified attacks. To examine the behavioural equivalence, adversary's views denoted by partial traces must be checked to be equal in each step of a protocol. The verifier automatically checks the equality using user-defined equations.

Second, we applied our verifier to Shor and Preskill's security proof of BB84 [4]. The verifier takes as input two configurations denoting BB84 and the EDP-based protocol, and equations about the properties of error-correcting codes and measurement of the halves of EPR pairs. The verifier succeeds to verify the bisimilarity of the configurations of the two protocols.

The package of the verifier is available from `http://hagi.is.s.u-tokyo.ac.jp/~tk/qccs verifier.tar.gz`. It includes a user manual and and example scripts in the folders `doc` and `scripts`.

Related Work The authors of qCCS presented the notion of symbolic bisimulation for quantum processes [8]. A purpose is to verify bisimilarity algorithmically. They proved the strong (internal actions should be simulated) symbolic bisimilarity is equivalent to the strong open bisimilarity, and actually presented an algorithm to verify symbolic ground (outsiders do not perform quantum operations adaptively) bisimilarity. Since our purpose is to apply a process calculus to security proofs where adversarial interference must be taken into account, we implemented a tool that verifies weak open bisimilarity on the basis of the original qCCS [6].

2 The Verifier

Formal Framework The verifier employs a sublanguage of qCCS, where the syntax of recursion and the use of classical data are restricted. The framework is still expressive, because our target protocols do not need recursion and classical probability distributions can be represented as quantum states denoted by diagonal operators. The syntax of the processes in the verifier is as follows.

$$P ::= \texttt{discard(}\tilde{q}\texttt{)} \mid \mathbf{c}?q.P \mid \mathbf{c}!q.P \mid \texttt{meas } b \texttt{ then } P \texttt{ saem} \mid op\,\texttt{[}\tilde{q}\texttt{]}.P \mid P||P \mid P\backslash L$$

Intuitively, `discard(`$\tilde{q}$`)` means termination of a process keeping a sequence of quantum variables $\tilde{q}$ secret. $\mathbf{c}!q.P$ sends a quantum variable q to a channel $\mathbf{c}$ and then executes P. $\mathbf{c}?q.P$ receives quantum data to a variable q from a channel $\mathbf{c}$ and then executes P. `meas` b `then` P `saem` measures a quantum bit b and executes P if the outcome is 0 or terminates if the outcome is 1. $op\,\texttt{[}\tilde{q}\texttt{]}.P$ performs a superoperator op to quantum variables $\tilde{q}$ and then executes P. $P||P'$ executes P and P' in parallel. $P\backslash L$, where L is a set of channels, executes P keeping channels in L private. Let $\mathrm{qv}(P)$ be the set of quantum variables that occur in a process P. $\mathbf{c}!q.P$, `meas` b `then` P `saem`, $op\,\texttt{[}\tilde{q}\texttt{]}.P$ and $P||P'$ are defined only if $q \notin \mathrm{qv}(P)$, $b \in \mathrm{qv}(P)$, $\tilde{q} \subset \mathrm{qv}(P)$ and $\mathrm{qv}(P) \cap \mathrm{qv}(P') = \emptyset$ respectively. In our verifier, each quantum variable is defined with its

qubit-length that is indicated by a natural number symbol declared beforehand. The adoption of natural number symbols is important to treat security parameters in the verifier. Quantum states are represented as character strings called environments. The syntax of the environments is given as follows.

$$\rho ::= X\,\texttt{[}\tilde{q}\texttt{]} \mid op\,\texttt{[}\tilde{q}\texttt{]}\,\texttt{(}\rho\texttt{)} \mid \rho * \rho \mid \texttt{proj}i\,\texttt{[}b\texttt{]}\,\texttt{(}\rho\texttt{)} \mid \texttt{Tr[}\tilde{q}\texttt{]}\,\texttt{(}\rho\texttt{)}$$

Let $\mathrm{qv}(\rho)$ be the set of quantum variables that occur in an environment ρ. $X\,\texttt{[}\tilde{q}\texttt{]}$ means that quantum variables $\tilde{q}$ are in a quantum state X. $op\,\texttt{[}\tilde{q}\texttt{]}\,\texttt{(}\rho\texttt{)}$ is a quantum state after the application of operation op to quantum variables $\tilde{q}$ in an environment ρ. $\rho * \rho'$ represents the tensor product of ρ and ρ' but $\rho * \rho'$ and $\rho' * \rho$ are identified in the verifier. $\rho * \rho'$ is defined only if $\mathrm{qv}(\rho) \cap \mathrm{qv}(\rho') = \emptyset$. $\texttt{proj}i\,\texttt{[}b\texttt{]}\,\texttt{(}\rho\texttt{)}$ means the reduced density operator $|i\rangle\langle i|_b \rho |i\rangle\langle i|_b$ with $i \in \{0, 1\}$. $\texttt{Tr[}\tilde{q}\texttt{]}\,\texttt{(}\rho\texttt{)}$ means the partial trace of ρ by $\tilde{q}$. We call a pair $\langle P, \rho\rangle$ of a process P and an environment ρ a configuration only if $\mathrm{qv}(P) \subset \mathrm{qv}(\rho)$, because quantum states of qubits occurring in P must be defined in ρ. For example, an agent that sends the halves of n EPR pairs to the outside is formalized as a configuration $\langle \texttt{c!q.discard(r)}, \texttt{EPR[q,r]*ANY[s]}\rangle$, where the quantum variables $\texttt{q}, \texttt{r}$ are of the qubit length n and the quantum variable $\texttt{s}$ is defined with an arbitrary qubit-length. $\texttt{EPR[q,r]}$ is interpreted to $(|00\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 00| + |11\rangle\langle 11|)^{\otimes n}_{\texttt{q,r}}$ and $\texttt{ANY[s]}$ is arbitrarily for adversary's density operators.

A labelled transition system based on the original one [6] is implemented in the verifier. Transitions are interpreted as interactions between configurations and the adversary. For example, the configuration above performs a transition $\langle \texttt{c!q.discard(r)}, \texttt{EPR[q,r]*ANY[s]}\rangle \xrightarrow{\texttt{c!q}} \langle \texttt{discard(r)}, \texttt{EPR[q,r]*ANY[s]}\rangle$, where the adversary can recognize that the quantum variable $\texttt{q}$ is sent through the channel $\texttt{c}$. Transitions caused by internal actions such as quantum operations, internal communications and conditional branches are labelled τ. The label τ intuitively means that the action is invisible from the outside. The transition rules of conditional branch are $\langle \texttt{meas}\ b\ \texttt{then}\ P\ \texttt{saem}, \rho\rangle \xrightarrow{\tau} \langle P, \texttt{proj0[}b\texttt{](}\rho\texttt{)}\rangle$ and $\langle \texttt{meas}\ b\ \texttt{then}\ P\ \texttt{saem}, \rho\rangle \xrightarrow{\tau} \langle \texttt{discard(}\mathrm{qv}(P)\texttt{)}, \texttt{proj1[}b\texttt{](}\rho\texttt{)}\rangle$. These may decrease the trace of the density operator denoting a quantum state. Since the trace indicates the probability to reach to a configuration, we can restore the probability from configurations to conditional branches. We identify the branches that are evoked by an identical quantum variable. Eventually, probability is absorbed in quantum states and excluded from the transition system.

The transition system of the original qCCS is probabilistic, which is caused by the measurement construction $M[q; x]$ in the syntax [6]. It measures an observable M of a quantum variable q and stores the result in a classical variable x. There are two ways to formalize quantum measurement since one can be also formalized as a superoperator. Because only $M[q; x]$ evokes a probabilistic branch, processes with different formalization of measurement are not bisimilar in general. We proposed a criteria how to select the way in our previous work [7]; we should use $M[q; x]$ if a process behaves differently according to the measurement. For example, the case where a process performs different labeled transitions is typical. Otherwise, we should formalize measurement by a superoperator. In our verifier's framework, $M[q; x]$ cannot be written independently. Instead, the syntax $\texttt{meas}\ b\ \texttt{then}\ P\ \texttt{saem}$ is the composition of $M[q; x]$ and the conditional branch in the original qCCS. This restriction and the conditions on quantum variables prevent the situation where two configurations with the different ways of formalization of measurement are not bisimilar.

Procedure to Check Bisimulation Weak bisimulation relation [6] is defined on qCCS configurations. Roughly speaking, weakly bisimilar configurations (1) perform identical labelled transitions up to τ transitions and (2) reveal identical density operators whatever operations the adversary performs to her quantum variables. The adversary's view is denoted by the partial

trace $\texttt{Tr}[\tilde{q}](\rho)$ when the global quantum state is ρ and $\tilde{q}$ are the quantum variables that are not revealed to the adversary. The verifier takes as input two configurations and user-defined equations on environments and returns *true* or *false*. It is designed to be sound with respect to the original qCCS, that is, two configurations are bisimilar if the verifier returns *true* with them and some valid equations. This is because

$$\hat{\mathcal{R}}_{eqs} = \{(\mathcal{C}, \mathcal{D}) \mid \text{The verifier returns } \textit{true}, \text{ given } \mathcal{C}, \mathcal{D}, \text{and } eqs.\}$$

is a bisimulation relation for all valid equations *eqs*. Precisely, $\hat{\mathcal{R}}_{eqs}$ is converted accordingly because $\texttt{meas}\ b\ \texttt{then}\ P\ \texttt{saem}$ is decomposed to $M[b;x]$ and a conditional branch, which performs a two-step transition in the branch with $x = 0$. The recursive procedure to verify bisimilarity is as follows. Since the transitions of the processes written in our sublanguage are finite, the procedure always terminates.

1. The procedure takes as input two configurations $\langle P_0, \rho_0\rangle$, $\langle Q_0, \sigma_0\rangle$ and user-defined equations on environments.
2. If P_0 and Q_0 can perform any τ-transitions of superoperator applications, they are all performed at this point. Let $\langle P, \rho\rangle$ and $\langle Q, \sigma\rangle$ be the obtained configurations.
3. Whether $\mathrm{qv}(P) = \mathrm{qv}(Q)$ is checked. If it does not hold, the procedure returns *false*.
4. Whether $\texttt{Tr}[\mathrm{qv}(P)](\rho) = \texttt{Tr}[\mathrm{qv}(Q)](\sigma)$ is checked with user-defined equations. The procedure to check the equality of quantum states are described in the next subsection. If it does not hold, the procedure returns *false*.
5. A new superoperator symbol $\mathcal{E}[\mathrm{qv}(\rho) - \mathrm{qv}(P)]$ that stands for an adversarial operation is generated.
6. For each $\langle P', \rho'\rangle$ such that $\langle P, \mathcal{E}[\mathrm{qv}(\rho) - \mathrm{qv}(P)](\rho)\rangle \xrightarrow{\alpha} \langle P', \rho'\rangle$, the procedure checks whether there exists $\langle Q', \sigma'\rangle$ such that $\langle Q, \mathcal{E}[\mathrm{qv}(\sigma) - \mathrm{qv}(Q)](\sigma)\rangle \xrightarrow{\tau*}\xrightarrow{\alpha}\xrightarrow{\tau*} \langle Q', \sigma'\rangle$ and the procedure returns *true* with the input $\langle P', \rho'\rangle$ and $\langle Q', \sigma'\rangle$. If there exists, it goes to the next step 7. Otherwise, it returns *false*.
7. For each $\langle Q', \sigma'\rangle$ such that $\langle Q, \mathcal{E}[\mathrm{qv}(\sigma) - \mathrm{qv}(Q)](\sigma)\rangle \xrightarrow{\alpha} \langle Q', \sigma'\rangle$, the procedure checks whether there exists $\langle P', \rho'\rangle$ such that $\langle P, \mathcal{E}[\mathrm{qv}(\rho) - \mathrm{qv}(P)](\rho)\rangle \xrightarrow{\tau*}\xrightarrow{\alpha}\xrightarrow{\tau*} \langle P', \rho'\rangle$ and the procedure returns *true* with the input $\langle Q', \sigma'\rangle$ and $\langle P', \rho'\rangle$. If there exists, it returns *true*. Otherwise, it returns *false*.

The step 2 prominently reduces the search space. Indeed, $\langle op_1[\tilde{q}].P || op_2[\tilde{r}].Q, \rho\rangle$ and $\langle P || Q, \mathcal{F}^{\tilde{r}}_{op_2}(\mathcal{E}^{\tilde{q}}_{op_1}(\rho))\rangle$ are bisimilar, and $\mathcal{F}^{\tilde{r}}_{op_2}(\mathcal{E}^{\tilde{q}}_{op_1}(\rho)) = \mathcal{E}^{\tilde{q}}_{op_1}(\mathcal{F}^{\tilde{r}}_{op_2}(\rho))$ holds because $\tilde{q} \cap \tilde{r} = \emptyset$ holds. Therefore, superoperators that can originally cause τ transitions are disposed.

Equality Test of Quantum States To test the equality of given two environments, the verifier checks whether they are transformed to the same form. There are two kinds of transformations, trace out and application of user-defined rules.

Partial trace is significant to test the equality of quantum states. For example, two different quantum states $\texttt{op1[q](X[q]*Y[r]*Z[s])}$ and $\texttt{op2[r](U[q]*V[r]*Z[s])}$ have the same partial trace $\texttt{Z[s]}$ under $\texttt{Tr[q,r]}$ for arbitrary interpretation of the superoperators $\texttt{op1}, \texttt{op2}$ and the quantum states $\texttt{X}, \texttt{Y}, \texttt{Z}, \texttt{U}, \texttt{V}$. The verifier eliminates symbols of superoperators and quantum states according to the partial trace rules below.

$$\texttt{Tr}[\tilde{q}](\mathcal{E}[\tilde{r}](\rho)) = \texttt{Tr}[\tilde{q}](\rho) \text{ if } \tilde{r} \subset \tilde{q} \quad (1), \quad \texttt{Tr}[\tilde{q}](\rho) = \texttt{Tr}[\tilde{r}](\texttt{Tr}[\tilde{s}](\rho)) \text{ if } \tilde{q} = \tilde{r} \cup \tilde{s} \quad (2),$$
$$\texttt{Tr}[\tilde{q}](\mathcal{E}[\tilde{r}](\rho)) = \mathcal{E}[\tilde{r}](\texttt{Tr}[\tilde{q}](\rho)) \text{ if } \tilde{q} \cap \tilde{r} = \emptyset \quad (3), \quad \texttt{Tr}[\tilde{q}](\rho * \rho_{\tilde{q}} * \sigma) = \rho * \sigma \quad (4).$$

If $\tilde{r} \subset \tilde{q}$ holds, `Tr[`$\tilde{q}$`](`$\mathcal{E}$`[`$\tilde{r}$`](`ρ`))` is rewritten to `Tr[`$\tilde{q}$`](`ρ`)` eliminating $\mathcal{E}$`[`$\tilde{r}$`]` by the rule (1); otherwise, by the rules (2) and (3), the trace-out symbol with quantum variables that are disjoint to targets of superoperator goes inside of it's application. After eliminating superoperators, quantum states are traced out by the rule (4).

If an objective quantum state has a part that matches to the left-hand side of a user-defined equation, the part is rewritten to the right-hand side. To apply a user-defined equation, the verifier automatically solves commutativity of superoperators or partial traces that are applied to disjoint sets of quantum variables. For example, if the objective quantum state is `Tr[q](hadamard[s] (EPR[q,r]*X[s]))` and a user defines an equation `Tr[q](EPR[q,r])=Tr[q](PROB [q,r])` (E1), the application goes as follows.

$$\begin{aligned}\texttt{Tr[q](hadamard[s](EPR[q,r]*X[s]))} &= \texttt{hadamards[s](Tr[q](EPR[q,r]*X[s]))} \quad \text{(by 5)}\\ &= \texttt{hadamard[s](Tr[q](PROB[q,r]*X[s]))} \quad \text{(by E1)}\end{aligned}$$

Since the trace-out rules may have become applicable after applications of user-defined rules, the trace-out procedure is applied again. In each opportunity to test the equality of quantum states, each user-defined equation is applied only once. This guarantees whatever rules a user defines, the equality test terminates.

3 Application to Shor and Preskill's Security Proof

The two protocols BB84 and the EDP-based protocol [4] are formalized as qCCS configurations on the basis of our previous work [7]. Readers can find the script `scripts/shor-preskill.scr` in the package. The interpretations of the symbols of the quantum states and the superoperators that are used in the formalization are also included as `doc/shor-preskill_symbols.pdf`.

On Channels As in general QKD protocols, three kinds of channels are used: public quantum channels, private classical channels and public no-interpolate classical channels. Since the syntax has channel restriction $P\backslash L$, formalization of private channels is straightforward. Public no-interpolate classical channels are realized by copying the data. If a quantum variable q where a classical value is assigned is sent through a public no-interpolate channel c, this is formalized as $...$`copy[`q`,`Q`].`c`!`q`.`d`!`Q`...`$\backslash\{..., c, ...\}$, where Q is a new quantum variable, a superoperator `copy` copies the value of q to Q and d is a new non-restricted channel. q will securely sent through the restricted channel c and an adversary obtains the same value accessing Q through a public channel d. Note that the value of q can be copied because it is classical.

Equations for the Proof We defined 10 equations in the verifier. Equations are interpreted to those on density operators and proven correct. For example, the equation below tells the verifier that the halves of EPR pairs are indistinguishable from the uniform distribution. This is used to check that the adversary's views are identical in the EDP-based protocol and BB84 before Alice in the EDP-based protocol measures her halves of the check qubits.

```
equation E7
  Tr[q1_A](EPR[q1_A,q2_A]) = Tr[q1_A](PROB[q1_A,q2_A])
end
```

The user-defined rule above is interpreted to an equation $\mathrm{tr}_{\{\texttt{q1\_A}\}}(|00\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 00| + |11\rangle\langle 11|)^{\otimes n}_{\texttt{q1\_A,q2\_A}} = \mathrm{tr}_{\{\texttt{q1\_A}\}}(|00\rangle\langle 00| + |11\rangle\langle 11|)^{\otimes n}_{\texttt{q1\_A,q2\_A}}$. The other rules are related to CSS codes, partial traces and measurement of quantum states. We obtained them by formalizing the equality of quantum states that are discussed in the original proof [4].

It is actually natural to express quantum states as symbols with some equations in inputs. In cryptographic protocols, the dimension of quantum states and superoperators cannot be fixed

as a certain number as they depend on security parameters. In addition, qubits going through public channels are potentially interpolated by the adversary. Since arbitrary computation by the adversary must be considered, it should be described as an unspecified superoperator. Therefore, quantum states are difficult to be expressed as concrete matrices, which can be numerically analyzed. Although quantum states are treated symbolically, automatic calculation of partial traces is still possible focusing on occurrence of quantum variables in environments.
Experiment Result We ran the verifier with the input of `shor-preskill.scr`. We used a note PC with Intel Core i5 CPU M 460 @ 2.53GHz and 1GB memory. The transition tree of the EDP-based protocol has 621 nodes and 165 paths, and that of BB84 has 588 nodes and 165 paths. The verifier checked the bisimilarity of the two protocols in 15.98 seconds. The recursive procedure was called 951 times.

4 Conclusions

We presented a software tool to check bisimulation of qCCS configurations and applied it to a security proof of BB84 [4]. In security proofs, equivalence on protocols is often discussed. It can be described as bisimulation relation but it is nearly impossible to check by hand if state transitions of processes have many long branches. In addition, the equality of an adversary's view between two protocols has to be checked in each step. An adversary's view is calculated from a global quantum states, which is possibly denoted by a huge matrix. The verifier does exhausting parts of proofs of behavioral equivalence, namely, it checks the correspondence of all state transitions up to invisible ones and an adversary's view up to equations. On the other hand, a user only have to examine the correctness of formalization of protocols and validity of equations that he defines. It could be difficult to find all appropriate equations for a proof immediately. The verifier is also able to show environments and/or processes when the check procedure returns *false*. With the information, a user can modify equations to input.
Future Work We plan to define probabilistic bisimilarity, which denotes that configurations behave indistinguishably up to some probability. This will be helpful to prove "indistinguishability up to negligible probability" of protocols. It is quite common argument in the cryptography.

References

[1] S. Halevi. A plausible approach to computer-aided cryptographic proofs. Cryptology ePrint Archive, Report 2005/181, 2005.

[2] B. Blanchet, A. D. Jaggard, A. Scedrov, and J.-K. Tsay. Computationally sound mechanized proofs for basic and public-key kerberos. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 87–99, 2008.

[3] D. Mayers. Unconditional security in quantum cryptography. *J. ACM*, 48:351–406, May 2001.

[4] P. W. Shor and J. Preskill. Simple proof of security of the bb84 quantum key distribution protocol. *Phys. Rev. Lett.*, 85(2):441–444, Jul 2000.

[5] H.-K. Lo and H. F. Chau. Unconditional security of quantum key distribution over arbitrarily long distances. *Phys. Rev. Lett.*, 283(5410):2050–2056, Mar 1999.

[6] Y. Deng and Y. Feng. Open bisimulation for quantum processes. In *Theoretical Computer Science*, volume 7604 of *LNCS*, pages 119–133. 2012.

[7] T. Kubota, Y. Kakutani, G. Kato, Y. Kawano, and H. Sakurada. Application of a process calculus to security proofs of quantum protocols. In *Proceedings of WORLDCOMP/FCS2012*, Jul 2012.

[8] Y. Feng, Y. Deng, and M. Ying. Symbolic bisimulation for quantum processes. arXiv preprint arXiv:1202.3484, 2012.