Relationship of Three Cryptographic Channels in the UC Framework

Waka Nagao[1] Yoshifumi Manabe[1,2] and Tatsuaki Okamoto[1,2]

[1] Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Kyoto, 606-8501 Japan
w-nagao@ai.soc.i.kyoto-u.ac.jp
[2] NTT Labs, Nippon Telegraph and Telephone Corporation
3-9-11 Midori-cho, Musashino, Tokyo, 180-8585 Japan
{manabe.yoshifumi,okamoto.tatsuaki}@lab.ntt.co.jp

Abstract. The relationship of three cryptographic channels, secure channels (SC), anonymous channels (AC) and direction-indeterminable channels (DIC), was investigated by Okamoto. He showed that the three cryptographic channels are reducible to each other, but did not consider communication schedules clearly as well as composable security. This paper refines the relationship of the three channels in the light of communication schedules and composable security. We model parties by the task-probabilistic input/output automata (PIOA) to treat communication schedules, and adopt the universally composable (UC) framework by Canetti to treat composable security. We show that a class of anonymous channels, two-anonymous channels (2AC), and DIC are reducible to each other under any schedule and that DIC and SC are reducible to each other under some types of schedules, in the UC framework with the PIOA model.
Key words: Secure Channel (SC), Two-Anonymous Channel (2AC), Direction-Indeterminable Channel (DIC), Universal Composability (UC), Probabilistic Input/Output Automaton (PIOA)

1 Introduction

One of the most important results in cryptography is the relationship among *computational* cryptographic assumptions. For example, several of the most important cryptographic primitives such as pseudo-random generators, secure bit-commitment, and secure signature schemes have been proven to exist if and only if one-way functions exist [8–11, 13].

Apart from the *computational* assumptions made in the above-mentioned works, some *physical* or *unconditionally secure* assumptions and primitives, which rely on no computational condition/assumption, are known to be essential in unconditionally secure (or information theoretically) cryptography. For example, unconditionally secure multi-party protocols can be constructed assuming *secure channels* [1, 7].

The relationship of such *physical* (unconditionally secure) assumptions for channels has been studied by [12]. The paper shows that three physical assumptions about channels are equivalent (or reducible to each other). Here, the three physical assumptions are the existence of the anonymous channel(AC), direction-indeterminable channel(DIC), and secure channel(SC).

However, paper [12] did not consider communication schedules clearly as well as composable security, although the communication schedule like synchronous or asynchronous communication is critical in the reductions of the three channels, and composable security is crucial for channels since channels are always lower level components of systems and applications.

In this paper, we refine the relationship of the three channels in the light of communication schedules and composable security.

This paper adopts the universally composable (UC) framework by Canetti [2] to treat composable security, since UC is the most powerful and well-studied framework for composable security and is flexible enough to cover the security of physical (unconditionally secure) primitives like channels.

Although parties are usually modeled by interactive Turing machines (ITMs) in the standard settings in cryptography including the UC framework [2], this paper models parties by not ITMs but by task-probabilistic input/output automata (PIOA) [3–6] to treat communication schedules. This is because: ITMs cannot treat flexible communication schedules like various types of asynchronous and nondeterministic schedules, but task PIOA is one of the most powerful models to treat a variety of communication schedules. The master schedule in task PIOA can control timing of activation among party with flexible schedule.

This paper shows that a class of anonymous channels, two-anonymous channels (2AC), and DIC are reducible to each other under any schedule in the UC framework with the PIOA model. We also show that DIC and SC are reducible to each other under some types of schedules in the UC framework with the PIOA model.

2 Preliminaries

This section introduces the basic notion of (task) Probabilistic Input/Output Automata (PIOA) and security notion of Universally Composability(UC). (See papers [6] and [2] for PIOA and UC, respectively, if you need more details.)

2.1 (Task) Probabilistic I/O Automata

We recall the basic definitions of PIOA and task-PIOA from [3, 4, 6].

Definition 1. [Probabilistic I/O Automaton (PIOA)] *Let Q, $\overline{q}$, I, O, H and D be, respectively, a countable set of states, a start state (satisfying $\overline{q} \in Q$), a countable set of input actions, a countable set of output actions, a countable set of internal actions and a transition relation satisfying $D \subseteq Q \times (I \cup O \cup H) \times Disc(Q)$, where $Disc(Q)$ is the set of discrete probability measures on Q. Let PIOA $\mathcal{P}$ be the tuple of $(Q, \overline{q}, I, O, H, D)$.*

Definition 2. [Task Probabilistic I/O Automaton (Task PIOA)] *Let $T = (\mathcal{P}, \mathcal{R})$ be a task-PIOA , where $\mathcal{P} = (Q, \overline{q}, I, O, H, D)$ is a PIOA (satisfying the transition determinism and input enabling properties), and $\mathcal{R}$ is an equivalence relation on the locally-controlled actions $L = O \cup H$.*

Execution Fragment and Trace Let q_i and a_i for $i \in \{0, 1, 2, \cdots\}$ be states and actions, respectively. We consider that an execution fragment of task-PIOA T is the following infinite or finite sequence $\alpha = q_0 a_1 q_1 a_2 \ldots$. If the sequence α is a finite sequence, the last state of α is denoted by lst(α). If α is a finite sequence with lst(α) = q_{i+1}, for each (q_i, a_{i+1}, q_{i+1}) there exists a transition $(q_i, a_{i+1}, \mu) \in D$ with $q_{i+1} \in$ supp(μ), where supp(μ) is a support of μ. If there exists an execution fragment α of an automaton P, we denote by $trace(\alpha)$ the input and output (external actions) sequence obtained from α.

In this paper, we formally model parties $P_1, \cdots, P_n$ in a protocol by task-PIOA $T_1, \cdots, T_n$. Each party P_i has a local scheduler ρ_i for the task-PIOA T_i. There exists a master scheduler M for all parties, $P_1, \cdots, P_n$ in a protocol.

Definition 3. [Local Scheduler] *Let T be a closed task-PIOA for a party P. A local scheduler, ρ, for T is defined to be a finite or infinite sequence of tasks, $t_1, t_2, \cdots$; i.e., $\rho = t_1, t_2, \cdots$. ρ specifies the executing order of tasks in T. (We often omit the explicit description of ρ in the specification of a task-PIOA if ρ is trivial from the specification.)*

Definition 4. [Master Scheduler] *A master scheduler, M, is defined to be a finite or infinite sequence of party identifiers, $i_1, i_2, \cdots$; i.e., $M = i_1, i_2, \cdots$. M grobally specifies the executing order of tasks in a protocol of $(P_1, \cdots, P_n)$ with preserving the local schedulings of all parties.*

For example, let ρ_i of party i be $t_{i1}, t_{i2}, \cdots$ $(i = 1, 2, 3)$, and $M = 1, 2, 2, 2, 3, 1, 1, 3$. Then the grobal executing order of task is $t_{11}, t_{21}, t_{22}, t_{23}, t_{31}, t_{12}, t_{13}, t_{32}$.

The master schedule is not under the control of adversary although the local schedule is under the control of adversary. In other words, the adversary can not to intervene the master schedule, but he can encumber the local schedule.

2.2 Universal Composability

UC Security Let *Env*, *Adv*, and *Sim* be an environment, an adversary, and a simulator, respectively. Let *Real* denote the output of environment *Env* when interacting with adversary *Adv* and parties *Init* and *Rec* running channel protocol π. Let *Ideal* denote the output of environment *Env* after interacting in the ideal world with simulator *Sim* and ideal functionality $\mathcal{F}$. We say that *Real* UC-realizes $\mathcal{F}$, if for any adversary $Adv \in PPT$ (probabilistic polynomial time) there exists a simulator $Sim \in PPT$ such that for any environment $Env \in PPT$, $Ideal_{\mathcal{F},Sim,Env} \approx Real_{\pi,Adv,Env}$, where $\approx$ denotes statistically indistinguishable and PPT denotes a class of polynomial-time bounded machines.

Hereafter, we use the bold style, Real and Ideal, to express systems of the PIOA notion to distinguish the real and ideal world of UC security. We then use the roman style also means the task-PIOA to divide the notions between task-PIOA and UC notion.

3 Three Cryptographic Channels and Definitions

3.1 Secure Channel (SC)

A secure channel is a channel such that the initiator (message sender) and the receiver can safely transmit messages to each other without the content being retrieved by a third

party or adversary. This secure channel consists of three sessions, establish session, data sending session, and expire session: 1. The establish session creates a session between the initiator and the receiver to start message sending. 2. The data sending session sends a message to the receiver safely. 3. The expire session terminates the the existing session and clears the secret key.

Definition 5. *The code for secure channel,* F_{SC}*, is defined in Fig. 1. (*$\overline{X}$*, where (*$X \in \mathrm{Init}, \mathrm{Rec}\}$*), means that if* $X = \mathrm{Init}$ *then* $\overline{X} = \mathrm{Rec}$ *else if* $X = \mathrm{Rec}$ *then* $\overline{X} = \mathrm{Init}$*.)*

Code for Secure Channel Functionality, F_{SC}, where $X \in \{\mathrm{Init}, \mathrm{Rec}\}$ and $\mathtt{sid}_{\mathrm{SC}} = (\mathrm{Init}, \mathrm{Rec}, \mathtt{sid}'_{\mathrm{SC}})$.

State: $estcond_X \in \{\bot, \top\}$, initially $\bot$, $active \in \{\bot, \top\}$, initially $\bot$, $okcond_{\mathrm{Adv}} \in \{\bot, \top\}$, initially $\bot$, $mes \in (\{0,1\}) \cup \{\bot\}$, initially $\bot$, $ntask \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$

Transitions:

– Establish Session:

ESS1. $receive(\mathtt{Establish}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_X$ Precondition: $active = \bot$, Effect: $estcond_X := \top$, If $estcond_X = \top$ for all X then $active := \top$.

ESS2. $send(\mathtt{SID}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$ Precondition:$active = \top$, Effect: none

– Data Sending Session:

DSS1. $receive(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_X$ Precondition:$active = \top$ and $mes = \bot$, Effect: $mes := m$

DSS2. $send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, |m|)_{\mathrm{Adv}}$ Precondition: $active = \top$, $mes \neq \bot$, $okcond_{\mathrm{Adv}} = \bot$ and $m := mes$, Effect: none

DSS3. $receive(\mathtt{Response}, \mathtt{sid}_{\mathrm{SC}}, ok)_{\mathrm{Adv}}$ Precondition:$active = \top$ and $okcond_{\mathrm{Adv}} = \bot$, Effect: $okcond_{\mathrm{Adv}} := \top$

DSS4. $send(\mathtt{Receive}, \mathtt{sid}_{\mathrm{SC}}, mes)_{\overline{X}}$ Precondition: $active, mes = \top$ and $okcond_{\mathrm{Adv}} \neq \bot$, Effect: mes and $okcond_{\mathrm{Adv}} := \bot$

– Expire Session:

EXS1. $receive(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_X$ Precondition: $active \neq \bot$ and $mes = \bot$, Effect: $ntask :=$ EXS2

EXS2. $send(\mathtt{Expire}_{\mathrm{SC}}, \mathtt{sid}_{\mathrm{SC}})_{\mathrm{Adv}}$ Precondition: $ntask =$ EXS2, Effect: $active, ntask$ and $estcond_X := \bot$ for all X

Fig. 1. Code for Secure Channel Functionality, F_{SC}

3.2 Two-anonymous Channel (2AC)

An anonymous channel is one of the three cryptographic channels and is able to send some messages to the receiver from unknown senders (”anonymously”). The adversary can know the identity of the receiver and the message content, but cannot know who sent the message to the receiver. When two senders and a receiver anonymously communicate by a channel, we say the channel is a two-anonymous channel. That is, one of the two senders sends a message to the receiver. Note that two-anonymous channel can also be used when the receiver and one of the senders is the same process.

Definition 6. *The code for two-anonymous channel,* $\mathrm{F}_{\mathrm{2AC}}$*, is defined in Fig. 2.*

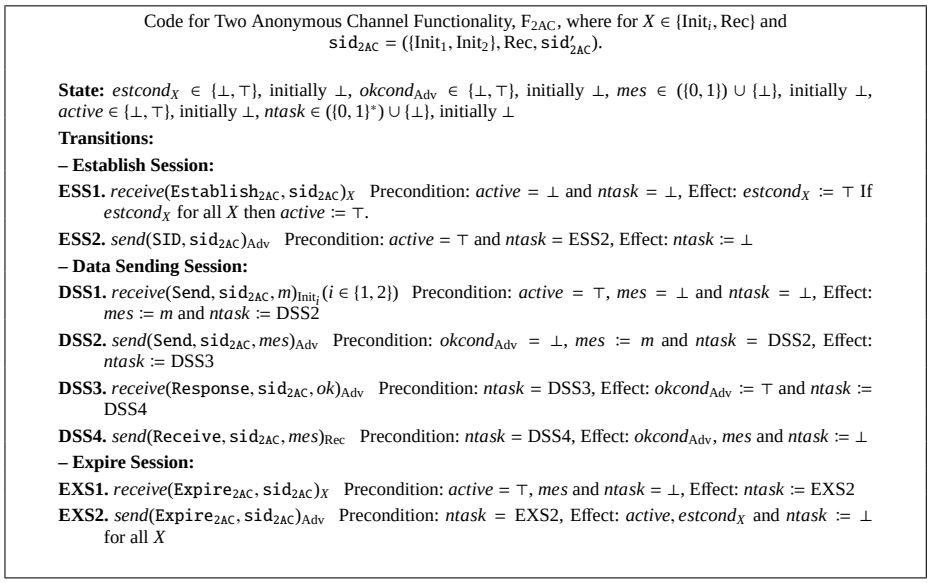

Fig. 2. Code for Two Anonymous Channel Functionality, F_{2AC}

3.3 Direction-indeterminable Channel (DIC)

A direction-indeterminable channel is one of the three cryptographic channels and is able to send some messages to the receiver direction-indeterminably. The adversary can know the identities of both parties and the transmitted message, but cannot know who the sender was. That is, the direction of message transmission is indeterminable.

Definition 7. *The code for direction-indeterminable channel,* F_{DIC}*, is defined in Fig. 3.*

3.4 Security Definitions

We define the security notion on PIOA considering the synchronous and asynchronous schedule as follows:

Definition 8. [Perfect Implementation] *Let* **Env**, **Real** *and* **Ideal** *be an environment task-PIOA, a real protocol task-PIOA system and an ideal functionality task-PIOA system, respectively. Let* sch *be the some (synchronous or asynchronous) schedule. We say that* **Real** *perfectly implements* **Ideal** *under some (synchronous or asynchronous) schedule (or* $\mathbf{Real} \leq_0^{\text{sch.}} \mathbf{Ideal}$*), if trace*(**Real**||**Env**) = *trace*(**Ideal**||**Env**) *for every environment* **Env** *under synchronous or asynchronous schedule.*

Definition 9. [Perfect Hybrid Implementation] *Let* Hybrid *be a real protocol task-PIOA system with hybrid model. We say that* Hybrid *perfectly hybrid implements* **Ideal** *under some (synchronous or asynchronous) schedule (or* $\text{Hyb.} \leq_0^{\text{sch.}} \mathbf{Ideal}$*), if trace* (**Hybrid**||**Env**) = *trace*(**Ideal**||**Env**) *for every environment* **Env** *under some (synchronous or asynchronous) schedule.*

Code for Direction-Indeterminable Channel Functionality, F_{DIC},
where $X \in \{\text{Init}, \text{Rec}\}$ and $\mathtt{sid}_{DIC} = (\{\text{Init}, \text{Rec}\}, \mathtt{sid}'_{DIC})$.

State: $estcond_X \in \{\bot, \top\}$, initially $\bot$, $mes \in (\{0,1\}) \cup \{\bot\}$, initially $\bot$, $okcond_{Adv} \in \{\bot, \top\}$, initially $\bot$, $active \in \{\bot, \top\}$, initially $\bot$, $ntask \in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$

Transitions:

– Establish Session:

ESS1. $receive(\mathtt{Establish}_{DIC}, \mathtt{sid}_{DIC})_X$ Precondition: $active = \bot$ and $ntask = \bot$, Effect: $estcond_X := \top$ If $estcond_X = \top$ for all X then $active := \top$ and $ntask :=$ ESS2.

ESS2. $send(\mathtt{SID}, \mathtt{sid}_{DIC})_{Adv}$ Precondition: $active = \top$ and $ntask =$ ESS2, Effect: $ntask := \bot$

– Data Sending Session:

DSS1. $receive(\mathtt{Send}, \mathtt{sid}_{DIC}, m)_X$ Precondition: $active = \top$, mes and $ntask = \bot$, Effect: $mes := m$ and $ntask :=$ DSS2

DSS2. $send(\mathtt{Send}, \mathtt{sid}_{DIC}, m)_{Adv}$ Precondition: $okcond_{Adv} = \bot$, $mes := m$ and $ntask =$ DSS2, Effect: $ntask :=$ DSS3

DSS3. $receive(\mathtt{Response}, \mathtt{sid}_{DIC}, ok)_{Adv}$ Precondition: $ntask =$ DSS3, Effect: $okcond_{Adv} := \top$ and $ntask :=$ DSS4

DSS4. $send(\mathtt{Send}, \mathtt{sid}_{DIC}, mes)_{\overline{X}}$ Precondition: $ntask =$ DSS4, Effect: $okcond_{Adv}$, mes and $ntask := \bot$

– Expire Session:

EXS1. $receive(\mathtt{Expire}_{DIC}, \mathtt{sid}_{DIC})_X$ Precondition: $active = \top$, $mes = \bot$ and $ntask = \bot$, Effect: $ntask :=$ EXS2

EXS2. $send(\mathtt{Expire}_{DIC}, \mathtt{sid}_{DIC})_{Adv}$ Precondition: $ntask =$ EXS2, Effect: $active$, $estcond_X$ and $ntask := \bot$ for all X

Fig. 3. Code for Direction-Indeterminable Channel Functionality, F_{DIC}

4 Equivalence Between DIC and 2AC

In this section, we prove that the direction indeterminable channel (DIC) is equivalent to the two-anonymous channel (2AC) under any schedule. That is, the task-PIOA of DIC perfectly implements the task-PIOA of 2AC under any schedule. To prove this, we show two reductions of DIC to 2AC and 2AC to DIC. Here, we consider the one bit message exchange, that is, $|m| = 1$. Informally, the reduction of DIC to 2AC is proven as follows: The direction-indeterminable property is made by using two 2AC functionalities, $F^{\mathtt{I}}_{2AC}$ and $F^{\mathtt{R}}_{2AC}$. Here, the two senders of $F^{\mathtt{I}}_{2AC}$ are Init and Rec, and the receiver of $F^{\mathtt{I}}_{2AC}$ is Init. Then, the two senders of $F^{\mathtt{R}}_{2AC}$ are Init and Rec, and the receiver of $F^{\mathtt{R}}_{2AC}$ is Rec. When Init sends a message to the receiver Rec, Init sends the message by $F^{\mathtt{I}}_{2AC}$ and $F^{\mathtt{R}}_{2AC}$. That is, $F^{\mathtt{I}}_{2AC}$ forwards the message to Init and $F^{\mathtt{R}}_{2AC}$ forwards the message to Rec. The adversary cannot detect the message direction because Init and Rec receive the same message m transfered by the two 2ACs. The other reduction, 2AC to DIC, is proven as follows: First, the message sending party (Init_1 or Init_2) sends a message m to the other party by DIC. Init_1 and Init_2 then send m to the receiver Rec directly. The adversary cannot detect which is the sender because the message direction among senders Init_1 and Init_2 is indeterminable.

4.1 Reduction of DIC to 2AC

Let π_{DIC} be a protocol of direction-indeterminable channel. We assume that $M_{\pi_{DIC}}$, the master schedule of π_{DIC}, is any schedule. Let Init_{DIC} and Rec_{DIC} be the initiator's code

and receiver's code for a real system, respectively, see Fig.4, Fig.5. Let $\overline{\text{Init}_{\text{DIC}}}$ and $\overline{\text{Rec}_{\text{DIC}}}$ be the initiator's code and receiver's code for an ideal system, respectively, see Fig.7 and Fig.8. Finally, let Adv_{DIC} and Sim_{DIC} be the adversary's code and the simulator's code in Fig.6 and Fig.9, respectively. Let $\mathbf{Real_{DIC}}$ and $\mathbf{Ideal_{DIC}}$ be a direction-indeterminable channel protocol system and a direction-indeterminable channel functionality system, respectively, defined as follows:

$$\mathbf{Real_{DIC}} := \text{Init}_{\text{DIC}} \| \text{Rec}_{\text{DIC}} \| \text{Adv}_{\text{DIC}} \| \text{F}^{\text{I}}_{\text{2AC}} \| \text{F}^{\text{R}}_{\text{2AC}},$$
$$\mathbf{Ideal_{DIC}} := \overline{\text{Init}_{\text{DIC}}} \| \overline{\text{Rec}_{\text{DIC}}} \| \text{Sim}_{\text{DIC}} \| \text{F}_{\text{DIC}}.$$

Code for Initiator of Direction-Indeterminable Channel, Init_{DIC}, where $X \in \{\text{I}, \text{R}\}$,
where $\text{sid}_{\text{DIC}} = (\{\text{Init}, \text{Rec}\}, \text{sid}'_{\text{DIC}})$, $\text{sid}^{\text{I}}_{\text{2AC}} = (\{\text{Init}, \text{Rec}\}, \text{Init}, \text{sid}'^{\text{I}}_{\text{2AC}})$ and $\text{sid}^{\text{R}}_{\text{2AC}} = (\{\text{Init}, \text{Rec}\}, \text{Rec}, \text{sid}'^{\text{R}}_{\text{2AC}})$.

State: $smes, rmes \in \{0,1\}^* \cup \{\perp\}$, initially $\perp$, $ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$, $active \in \{\perp, \top\}$, initially $\perp$
Transitions:
– Establish Session:
ESS1. $in(\text{Establish}_{\text{DIC}}, \text{sid}_{\text{DIC}})_{\text{Init}}$ Precondition: $active, ntask = \perp$, Effect: $ntask := \text{ESS2}$
ESS2. $send(\text{Establish}_{\text{2AC}}, \text{sid}^{\text{X}}_{\text{2AC}})_{\text{F}^{\text{X}}_{\text{2AC}}}$ Precondition: $ntask = \text{ESS2}$ (Note that each task for $X \in \{I, R\}$ activates arbitrarily order. Hereafter, we stand by this manner.), Effect: $active := \top$ and $ntask := \perp$
– Data Sending Session:
DSS1. $in(\text{Send}, \text{sid}_{\text{DIC}}, m)_{\text{Init}}$ Precondition: $active = \top$, $smes$ and $ntask = \perp$, Effect: $smes := m$ and $ntask :=$ DSS2
DSS2. $send(\text{Send}, \text{sid}^{\text{X}}_{\text{2AC}}, m)_{\text{F}^{\text{X}}_{\text{2AC}}}$ Precondition: $m := smes$ and $ntask = \text{DSS2}$, Effect: $ntask := \perp$
DSS3. $receive(\text{Receive}, \text{sid}^{\text{I}}_{\text{2AC}}, m)_{\text{F}^{\text{I}}_{\text{2AC}}}$ Precondition: $active = \top$, $rmes$ and $ntask = \perp$, Effect: If $smes = \perp$, then $rmes := m$ and $ntask :=$ DSS4. Else $smes := \perp$ and $ntask := \perp$
DSS4. $out(\text{Receive}, \text{sid}_{\text{DIC}}, r)_{\text{Init}}$ Precondition: $r := rmes$ and $ntask = \text{DSS4}$, Effect: $rmes$ and $ntask := \perp$
– Expire Session:
EXS1. $in(\text{Expire}_{\text{DIC}}, \text{sid}_{\text{DIC}})_{\text{Init}}$ Precondition: $active = \top$ and $ntask = \perp$, Effect: $ntask := \text{EXS2}$
EXS2. $send(\text{Expire}_{\text{2AC}}, \text{sid}^{\text{X}}_{\text{2AC}})_{\text{F}^{\text{X}}_{\text{2AC}}}$ Precondition: $ntask = \text{EXS2}$, Effect: $active$ and $ntask := \perp$

Fig. 4. Code for Initiator of Direction-Indeterminable Channel, Init_{DIC}

Tasks $\overline{\text{Init}_{\text{DIC}}}$ and $\overline{\text{Rec}_{\text{DIC}}}$ relay the input messages from the environment to the ideal functionality task and relay the receive messages from the ideal functionality task to the environment as interface parties in the ideal system.

Theorem 1. *Direction-indeterminable channel protocol system* $\mathbf{Real_{DIC}}$ *perfectly hybrid implements direction-indeterminable channel functionality system* $\mathbf{Ideal_{DIC}}$ *with respect to adaptive adversary under any master schedule. (A direction-indeterminable channel is reducible to a two-anonymous channel with respect to adaptive adversary under any master schedule.)*

Let ϵ_{R} and ϵ_{I} be discrete probability measures on finite executions of $\mathbf{Real_{DIC}} \| \mathbf{Env}$ and $\mathbf{Ideal_{DIC}} \| \mathbf{Env}$, respectively. We prove the Theorem 1 by showing that ϵ_{R} and ϵ_{I} satisfy the trace distribution property : $tdist(\epsilon_{\text{R}}) = tdist(\epsilon_{\text{I}})$. Here, we define correspondence R between the states in $\mathbf{Real_{DIC}} \| \mathbf{Env}$ and the states in $\mathbf{Ideal_{DIC}} \| \mathbf{Env}$. We say ($\epsilon_{\text{R}}$,

Code for Receiver of Direction-Indeterminable Channel, Rec_{DIC}, where $\texttt{sid}_{\text{DIC}} = (\{\text{Init}, \text{Rec}\}, \texttt{sid}'_{\text{DIC}})$, $\texttt{sid}^{\text{I}}_{2\text{AC}} = (\{\text{Init}, \text{Rec}\}, \text{Init}, \texttt{sid}'^{\text{I}}_{2\text{AC}})$ and $\texttt{sid}^{\text{R}}_{2\text{AC}} = (\{\text{Init}, \text{Rec}\}, \text{Rec}, \texttt{sid}'^{\text{R}}_{2\text{AC}})$.

State: $smes, rmes \in \{0,1\}^* \cup \{\perp\}$, initially $\perp$, $ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$, $active \in \{\perp, \top\}$, initially $\perp$

Transitions:

– Establish Session:

ESS1. $in(\texttt{Establish}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\text{Rec}}$ Precondition: $active$ and $ntask = \perp$, Effect: $ntask := \text{ESS2}$

ESS2. $send(\texttt{Establish}_{2\text{AC}}, \texttt{sid}^{\text{X}}_{2\text{AC}})_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $ntask = \text{ESS2}$, Effect: $active := \top$ and $ntask := \perp$

– Data Sending Session:

DSS1. $in(\text{Send}, \texttt{sid}_{\text{DIC}}, m)_{\text{Rec}}$ Precondition: $active = \top$, $smes$ and $ntask = \perp$, Effect: $smes := m$ and $ntask :=$ DSS2

DSS2. $send(\text{Send}, \texttt{sid}^{\text{X}}_{2\text{AC}}, m)_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $m := smes$ and $ntask = \text{DSS2}$, Effect: $ntask := \perp$

DSS3. $receive(\text{Receive}, \texttt{sid}^{\text{R}}_{2\text{AC}}, m)_{\text{F}^{\text{R}}_{2\text{AC}}}$ Precondition: $active = \top$, $rmes$ and $ntask = \perp$, Effect: If $smes = \perp$, then $rmes := m$ and $ntask :=$ DSS4. Else $smes := \perp$ and $ntask := \perp$.

DSS4. $out(\text{Receive}, \texttt{sid}_{\text{DIC}}, r)_{\text{Rec}}$ Precondition: $r := rmes$ and $ntask = \text{DSS4}$, Effect: $rmes, ntask := \perp$

– Expire Session:

EXS1. $in(\texttt{Expire}_{\text{DIC}}, \texttt{sid}_{\text{DIC}})_{\text{Rec}}$ Precondition: $active = \top$ and $ntask = \perp$, Effect: $ntask := \text{EXS2}$

EXS2. $send(\texttt{Expire}_{2\text{AC}}, \texttt{sid}^{\text{X}}_{2\text{AC}})_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $ntask = \text{EXS2}$, Effect: $active$ and $ntask := \perp$

Fig. 5. Code for Receiver of Direction-Indeterminable Channel, Rec_{DIC}

Code fot Adversary for Direction Indeterminable Channel, Adv_{DIC}, where $X \in \{\text{I}, \text{R}\}$, where $\texttt{sid}^{\text{I}}_{2\text{AC}} = (\{I_1, I_2\}, I_1, \texttt{sid}'_{2\text{AC}})$ and $\texttt{sid}^{\text{R}}_{2\text{AC}} = (\{I_1, I_2\}, I_2, \texttt{sid}'_{2\text{AC}})$.

State: $active \in \{\perp, \top\}$, initially $\perp$, $ntask \in (\{0,1\}^*) \cup \{\perp\}$, initially $\perp$, $smes_X \in (\{0,1\}) \cup \{\perp\}$, initially $\perp$

Transitions:

– Establish Session:

ESS1. $receive(\text{SID}, \texttt{sid}^{\text{X}}_{2\text{AC}})_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $active = \perp$, Effect:$active := \top$

– Data Sending Session:

DSS**1.** $receive(\text{Send}, \texttt{sid}^{\text{X}}_{2\text{AC}}, mes)_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $active = \top$, $ntask = \perp$, Effect: $smes_X := mes$ and $ntask :=$ DSS2

DSS**2.** $send(\text{Response}, \texttt{sid}^{\text{X}}_{2\text{AC}}, ok)_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $ntask = \text{DSS2}$, Effect: $ntask := \perp$

– Expire Session:

EXS**1.** $receive(\texttt{Expire}_{2\text{AC}}, \texttt{sid}^{\text{X}}_{2\text{AC}})_{\text{F}^{\text{X}}_{2\text{AC}}}$ Precondition: $active = \top$, Effect: $active := \perp$

– Other tasks:

This adversary makes other arbitary tasks.

Fig. 6. Code fot Adversary for Direction Indeterminable Channel, Adv_{DIC}

$\epsilon_{\text{I}}) \in R$ if and only if for every $s \in \text{supp.lst}(\epsilon_{\text{R}})$ and $u \in \text{supp.lst}(\epsilon_{\text{I}})$, all of the state correspondences in the Table 1 hold. We then prove R is a simulation relation in Lemma 1.

Lemma 1. *The relation R defined above is a simulation relation from* $\mathbf{Real_{DIC}}||\mathbf{Env}$ *to* $\mathbf{Ideal_{DIC}}||\mathbf{Env}$. *For each step of* $\mathbf{Real_{DIC}}||\mathbf{Env}$, *the step in the establish, data sending and expire session correspond with at most two steps of* $\mathbf{Ideal_{DIC}}||\mathbf{Env}$. *This means that there is a mapping corrtasks under the relation R such that, for every* ρ, T, $|corrtasks(\rho, T)| \leq 2$, *where* ρ *is a local schedule.*

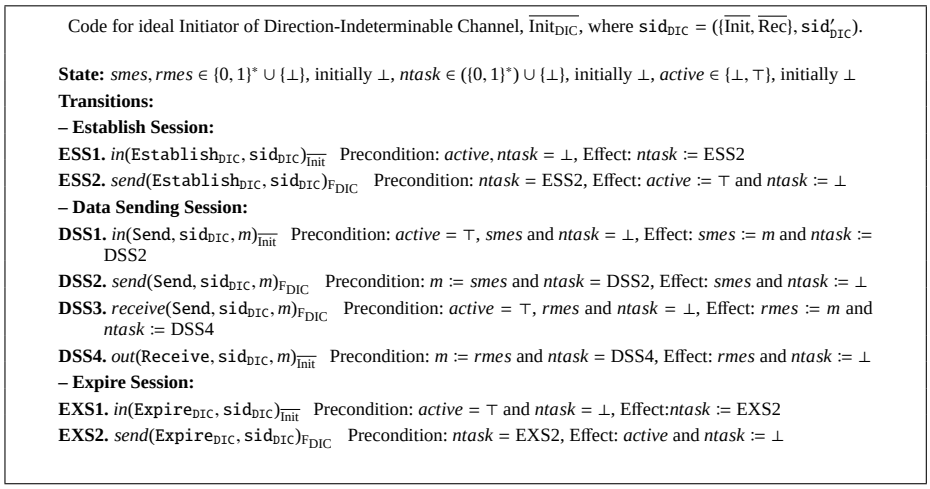

Fig. 7. Code for Initiator of Direction-Indeterminable Channel, $\overline{\text{Init}_{\text{DIC}}}$

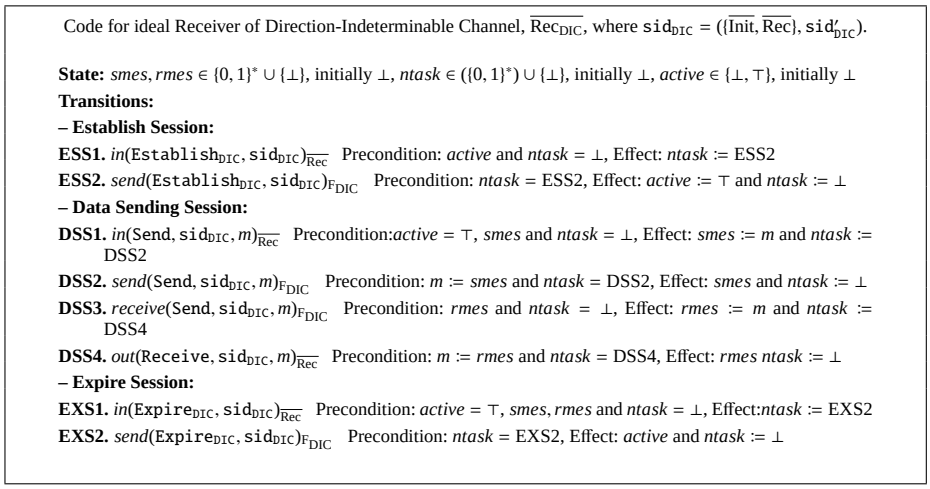

Fig. 8. Code for ideal Receiver of Direction-Indeterminable Channel, $\overline{\text{Rec}_{\text{DIC}}}$

Proof. (sketch)

We prove that R is a simulation relation from $\mathbf{Real_{DIC}}||\mathbf{Env}$ to $\mathbf{Ideal_{DIC}}||\mathbf{Env}$ using the mapping corrtasks : $R^*_{\mathbf{Real_{DIC}}||\mathbf{Env}} \times R_{\mathbf{Real_{DIC}}||\mathbf{Env}} \rightarrow R^*_{\mathbf{Ideal_{DIC}}||\mathbf{Env}}$, which is defined as follows (we write hereafter $T =_{\text{corr.}} T'$ alternating to write corrtasks$(\rho, T) = T'$.):

For any $(\rho, T) \in (R^*_{\mathbf{Real_{DIC}}||\mathbf{Env}} \times R_{\mathbf{Real_{DIC}}||\mathbf{Env}})$, the following task correspondences hold.

1. **Establish Session**
   (a) $\text{Init}_{\text{DIC}}.send(\texttt{Establish}_{\texttt{2AC}}, \texttt{sid}^{\texttt{x}}_{\texttt{2AC}})_{\text{F}^{\texttt{x}}_{\text{2AC}}} =_{\text{corr.}} \overline{\text{Init}_{\text{DIC}}}.send(\texttt{Establish}_{\texttt{DIC}}, \texttt{sid}_{\texttt{DIC}})_{\text{F}_{\text{DIC}}}$

Code for Simulator for Direction Indeterminable Channel, $\mathrm{Sim}_{\mathrm{DIC}}$, where $\mathtt{sid}_{\mathrm{DIC}} = (\{\mathrm{Init}, \mathrm{Rec}\}, \mathtt{sid}'_{\mathrm{DIC}})$.

State: *active* $\in \{\bot, \top\}$, initially $\bot$, *smes* $\in \{0,1\}^* \cup \{\bot\}$, initially $\bot$, *ntask* $\in (\{0,1\}^*) \cup \{\bot\}$, initially $\bot$

Other arbitrary variables; call "new" variables.

Transitions:

– Establish Session:

ESS1. *receive*$(\mathtt{SID}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F_{DIC}}}$ Precondition: *active* and *ntask* $= \bot$, Effect:*active* $:= \top$ This task generates the parties Init and Rec in the $\mathbf{Real_{DIC}}$ system to simulate the real world. To make establish session in the simulation world, inputs *in*$(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Init}}$ and *in*$(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Rec}}$ to Init and Rec, respectively. Finally, the parties establish the two 2ACs in the simulation world.

– Data Sending Session:

DSS1. *receive*$(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{F_{DIC}}}$ Precondition: *active* $= \top$, *ntask* $= \bot$, Effect: *smes* $:= m$ and *ntask* $:=$ DSS2

DSS2. *simulation*$(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, mes)$ Precondition: *mes* $:=$ *smes* and *ntask* $=$ DSS2, Effect: This task inputs *in*$(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{Init}}$ to Init in the simulation world. During the simulation, if the adversary in this simulation wants to output message to the environment, this simulator outputs the message after receiving from the adversary. After simulating the real world, the simulator receives *out*$(\mathtt{Receive}, \mathtt{sid}_{\mathrm{DIC}}, r)_{\mathrm{Rec}}$ from the receiver Rec and sets *ntask* $:=$ DSS3

DSS3. *send*$(\mathtt{Response}, \mathtt{sid}_{\mathrm{DIC}}, ok)_{\mathrm{F_{DIC}}}$ Precondition: *ntask* $=$ DSS3, Effect: *ntask* $:= \bot$

– Expire Session:

EXS**1.** *receive*$(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F_{DIC}}}$ Precondition: *active* $= \top$, Effect: *active* $:= \bot$

– Other tasks:

This simulator makes arbitary tasks to simulate the real world protocol system $\mathbf{Real_{DIC}}$. The tasks can be the input and output tasks with the internal tasks copied from $\mathbf{Real_{DIC}}$. Espectialy, this simulator can output the message from the adversary in the simiulating world to the environment.

Fig. 9. Code fot Simulator for Direction Indeterminable Channel, $\mathrm{Sim}_{\mathrm{DIC}}$

(b) $\mathrm{Rec}_{\mathrm{DIC}}.send(\mathtt{Establish}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{x}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{x}}_{\mathtt{2AC}}} =_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{DIC}}}.send(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F_{DIC}}}$

(c) $\mathrm{F}^{\mathtt{x}}_{\mathtt{2AC}}.send(\mathtt{SID}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}} =_{\mathrm{corr.}} \mathrm{F}_{\mathrm{DIC}}.send(\mathtt{SID}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Adv}}$

2. **Data Sending Session**

(a) $\mathrm{Init}_{\mathrm{DIC}}.send(\mathtt{Send}, \mathtt{sid}^{\mathtt{x}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{x}}_{\mathtt{2AC}}} =_{\mathrm{corr.}} \overline{\mathrm{Init}_{\mathrm{DIC}}}.send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{F_{DIC}}}$

(b) $\mathrm{Rec}_{\mathrm{DIC}}.send(\mathtt{Send}, \mathtt{sid}^{\mathtt{x}}_{\mathtt{2AC}}, m)_{\mathrm{F}^{\mathtt{x}}_{\mathtt{2AC}}} =_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{DIC}}}.send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{F_{DIC}}}$

(c) $\mathrm{F}^{\mathtt{x}}_{\mathtt{2AC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathtt{2AC}}, mes)_{\mathrm{Adv}}$
$=_{\mathrm{corr.}} \mathrm{F}_{\mathrm{DIC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{Adv}} \cdot \mathrm{Sim}_{\mathrm{DIC}}.simulation(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, mes)$

(d) $\mathrm{F}^{\mathtt{x}}_{\mathtt{2AC}}.send(\mathtt{Receive}, \mathtt{sid}_{\mathtt{2AC}}, mes)_{\mathrm{Rec}} =_{\mathrm{corr.}} \mathrm{F}_{\mathrm{DIC}}.send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, mes)_{\overline{X}}$

(e) $\mathrm{Init}_{\mathrm{DIC}}.out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{DIC}}, r)_{\mathrm{Init}} =_{\mathrm{corr.}} \overline{\mathrm{Init}_{\mathrm{DIC}}}.out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\overline{\mathrm{Init}}}$

(f) $\mathrm{Rec}_{\mathrm{DIC}}.out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{DIC}}, r)_{\mathrm{Rec}} =_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{DIC}}}.out(\mathtt{Receive}, \mathtt{sid}_{\mathrm{DIC}}, mes)_{\overline{\mathrm{Rec}}}$

(g) $\mathrm{Adv}_{\mathrm{DIC}}.send(\mathtt{Response}, \mathtt{sid}^{\mathtt{x}}_{\mathtt{2AC}}, ok)_{\mathrm{F}^{\mathtt{x}}_{\mathtt{2AC}}} =_{\mathrm{corr.}} \mathrm{Sim}_{\mathrm{DIC}}.send(\mathtt{Response}, \mathtt{sid}_{\mathrm{DIC}}, ok)_{\mathrm{F_{DIC}}}$

3. **Expire Session**

(a) $\mathrm{Init}_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{x}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{x}}_{\mathtt{2AC}}} =_{\mathrm{corr.}} \overline{\mathrm{Init}_{\mathrm{DIC}}}.send(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F_{DIC}}}$

(b) $\mathrm{Rec}_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}^{\mathtt{x}}_{\mathtt{2AC}})_{\mathrm{F}^{\mathtt{x}}_{\mathtt{2AC}}} =_{\mathrm{corr.}} \overline{\mathrm{Rec}_{\mathrm{DIC}}}.send(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F_{DIC}}}$

(c) $\mathrm{F}^{\mathtt{x}}_{\mathtt{2AC}}.send(\mathtt{Expire}_{\mathtt{2AC}}, \mathtt{sid}_{\mathtt{2AC}})_{\mathrm{Adv}} =_{\mathrm{corr.}} \mathrm{F}_{\mathrm{DIC}}.send(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Adv}}$

4. All tasks of environment **Env** in $\mathbf{Real_{DIC}}$ are correspondent with the tasks of environment in $\mathbf{Ideal_{DIC}}$.

The simulation of $\mathrm{Sim}_{\mathrm{DIC}}$ is perfectly done for establish session, data sending session and expire session with respect to the no corruption, static corruption and adaptive corruption by adversary.

	Functionality			**Receiver**	
(a)	$u.\mathrm{F}_{\mathrm{DIC}}.estcond_{\mathrm{Init}}$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.estcond_{\mathrm{Init}_i}$	(k)	$u.\overline{\mathrm{Rec}_{\mathrm{DIC}}}.smes$	$= s.\mathrm{Rec}_{\mathrm{DIC}}.smes$
(b)	$u.\mathrm{F}_{\mathrm{DIC}}.estcond_{\mathrm{Rec}}$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.estcond_{\mathrm{Rec}}$	(l)	$u.\overline{\mathrm{Rec}_{\mathrm{DIC}}}.rmes$	$= s.\mathrm{Rec}_{\mathrm{DIC}}.rmes$
(c)	$u.\mathrm{F}_{\mathrm{DIC}}.okcond_{\mathrm{Adv}}$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.okcond_{\mathrm{Adv}}$	(m)	$u.\overline{\mathrm{Rec}_{\mathrm{DIC}}}.active$	$= s.\mathrm{Rec}_{\mathrm{DIC}}.active$
(d)	$u.\mathrm{F}_{\mathrm{DIC}}.active$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.active$	(n)	$u.\overline{\mathrm{Rec}_{\mathrm{DIC}}}.ntask$	$= s.\mathrm{Rec}_{\mathrm{DIC}}.ntask$
(e)	$u.\mathrm{F}_{\mathrm{DIC}}.mes$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.mes$			
(f)	$u.\mathrm{F}_{\mathrm{DIC}}.ntask$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.ntask$			
	Initiator			**Adversary** and **Env**	
(g)	$u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.smes$	$= s.\mathrm{Init}_{\mathrm{DIC}}.smes$	(o)	$u.\mathrm{Sim}_{\mathrm{DIC}}$	$= s.\mathrm{Adv}_{\mathrm{DIC}}$
(h)	$u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.rmes$	$= s.\mathrm{Init}_{\mathrm{DIC}}.rmes$	(p)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.*$	$= s.\mathrm{F}^{\mathtt{x}}_{\mathrm{2AC}}.*$
(i)	$u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.active$	$= s.\mathrm{Init}_{\mathrm{DIC}}.active$	(q)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Init}_{\mathrm{DIC}}.*$	$= s.\mathrm{Init}_{\mathrm{DIC}}.*$
(j)	$u.\overline{\mathrm{Init}_{\mathrm{DIC}}}.ntask$	$= s.\mathrm{Init}_{\mathrm{DIC}}.ntask$	(r)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Rec}_{\mathrm{DIC}}.*$	$= s.\mathrm{Rec}_{\mathrm{DIC}}.*$
			(s)	$u.\mathrm{Sim}_{\mathrm{DIC}}.\mathrm{Adv}_{\mathrm{DIC}}.*$	$= s.\mathrm{Adv}_{\mathrm{DIC}}.*$
			(t)	$u.\mathrm{Env}$	$= s.\mathrm{Env}$

Note that $ntask \in \{\mathrm{ESS1}, \mathrm{ESS2}, \mathrm{DSS1}, \mathrm{DSS2}, \mathrm{DSS3}, \mathrm{DSS4}, \mathrm{EXS1}, \mathrm{EXS2}\}$, $i \in \{1, 2\}$ and $X \in \{I, R\}$.

Table 1. State correspondence : Reduction of DIC to 2AC

1. **No corruption**

   First, in the establish session, environment Env sends the establish session message $in(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\overline{\mathrm{Init}}}$ and $in(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\overline{\mathrm{Rec}}}$ to the initiator $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ and the receiver $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$, respectively. They send the establish session messages $send(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$. They send $send(\mathtt{SID}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Adv}}$ to the $\mathrm{Sim}_{\mathrm{DIC}}$. After $\mathrm{Sim}_{\mathrm{DIC}}$ receives the message, $\mathrm{Sim}_{\mathrm{DIC}}$ generates the parties Init and Rec in his simulation world to make the real world situation which Init and Rec exchange messages by using $\mathrm{F}_{\mathrm{2AC}}$. $\mathrm{Sim}_{\mathrm{DIC}}$ then make establish session in the simulation world. That is, he inputs two messages, $in(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Init}}$ and $in(\mathtt{Establish}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Rec}}$, to Init and Rec, respectively. Finally, the parties establish two 2ACs in the simulation world.

   Next, in the data sending session, Env sends the message $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\overline{\mathrm{Init}}}$ (or $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\overline{\mathrm{Rec}}}$) to $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ (or $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$). $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ sends $send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$. $\mathrm{F}_{\mathrm{DIC}}$ then sends $send(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{Adv}}$ to $\mathrm{Sim}_{\mathrm{DIC}}$. After receiving the message, $\mathrm{Sim}_{\mathrm{DIC}}$ executes $simulation(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, mes)$ to mimic the data sending session of the real world. That is, he inputs the message $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{Init}}$ (or $in(\mathtt{Send}, \mathtt{sid}_{\mathrm{DIC}}, m)_{\mathrm{Rec}}$) to Init and Rec in the simulation world.

   Finally, in the expire session, Env sends the messages $in(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\overline{\mathrm{Init}}}$ and $in(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\overline{\mathrm{Rec}}}$ to $\overline{\mathrm{Init}_{\mathrm{DIC}}}$ and $\overline{\mathrm{Rec}_{\mathrm{DIC}}}$, respectively. They relay the message $send(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{F}_{\mathrm{DIC}}}$ to $\mathrm{F}_{\mathrm{DIC}}$. After receiving $send(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Adv}}$ from $\mathrm{F}_{\mathrm{DIC}}$, $\mathrm{Sim}_{\mathrm{DIC}}$ expires the session in the simulation world. That is, he inputs the messgae $in(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Init}}$ and $in(\mathtt{Expire}_{\mathrm{DIC}}, \mathtt{sid}_{\mathrm{DIC}})_{\mathrm{Rec}}$ to Init and Rec in the simulation world.

2. **Static corruption**

In this case, the advesary corrupt some parties before the protocol starts. This case also is simulated by the simulator, but the direction of message sending does not conceal to the advesary.

3. **Adaptive corruption**
   In this case, the advesary corrupt some parties when he want to do so. This case also is simulated by the simulator, but the direction of message sending does not conceal to the advesary after he corrupts some parties.

As a result, the simulation is perfectly done because $\mathrm{Sim}_{\mathrm{DIC}}$ can simulate the real world from the information message through $\mathrm{F}_{\mathrm{DIC}}$. The tasks of the real world perfectly correspond with the the tasks of ideal world. That is,

$$\mathbf{Real_{DIC}}||\mathbf{Env} \;\; \text{Hyb.} \leq_0^{\mathtt{M}_{\pi_{\mathrm{DIC}}}} \mathbf{Ideal_{DIC}}||\mathbf{Env}.$$

The task sequence of the system $\mathbf{Real_{DIC}}||\mathbf{Env}$ are perfectively corresponded with the task sequence of the system $\mathbf{Ideal_{DIC}}||\mathbf{Env}$ under the schedule $M_{\pi_{\mathrm{DIC}}}$. Formally, to prove that R is simulation relation from $\mathbf{Real_{DIC}}||\mathbf{Env}$ to $\mathbf{Ideal_{DIC}}||\mathbf{Env}$, we need to show R satisfies start condition and step condition for each corresponding tasks, but we omit to mention it due to the paper limitation. See full paper that will be available soon.

4.2 Reduction of 2AC to DIC

Let $\pi_{2\mathrm{AC}}$ be a protocol of two-anonymous channel. We assume that $M_{\pi_{2\mathrm{AC}}}$, the master schedule of $\pi_{2\mathrm{AC}}$, is any schedule. Let $\mathrm{Init}_{2\mathrm{AC}}$ and $\underline{\mathrm{Rec}_{2\mathrm{AC}}}$ be the initiator's code and receiver's code for a real system, respectively. Let $\overline{\mathrm{Init}_{2\mathrm{AC}}}$ and $\overline{\mathrm{Rec}_{2\mathrm{AC}}}$ be the initiator's code and receiver's code for an ideal system, respectively. Finally, let $\mathrm{Adv}_{2\mathrm{AC}}$ and $\mathrm{Sim}_{2\mathrm{AC}}$ be the adversary's code and the simulator's code, respectively. Let $\mathbf{Real_{2AC}}$ and $\mathbf{Ideal_{2AC}}$ be a two-anonymous channel protocol system and a two-anonymous channel functionality system, respectively, defined as follows:

$$\mathbf{Real_{2AC}} \coloneqq \underline{\mathrm{Init}_{2\mathrm{AC}}}||\underline{\mathrm{Rec}_{2\mathrm{AC}}}||\mathrm{Adv}_{2\mathrm{AC}}||\mathrm{F}_{\mathrm{DIC}},$$
$$\mathbf{Ideal_{2AC}} \coloneqq \overline{\mathrm{Init}_{2\mathrm{AC}}}||\overline{\mathrm{Rec}_{2\mathrm{AC}}}||\mathrm{Sim}_{2\mathrm{AC}}||\mathrm{F}_{2\mathrm{AC}}.$$

Tasks $\overline{\mathrm{Init}_{2\mathrm{AC}}}$ and $\overline{\mathrm{Rec}_{2\mathrm{AC}}}$ relay the input messages from the environment to the ideal functionality task and relay the messages received from the ideal functionality task to the environment as interface parties in the ideal system. Several codes for each tasks are omitted in this paper, see full paper version.

Theorem 2. *Two-anonymous channel protocol system* $\mathbf{Real_{2AC}}$ *perfectly hybrid implements two-anonymous channel functionality system* $\mathbf{Ideal_{2AC}}$ *with respect to adaptive adversary under any master schedule. (An anonymous channel is reducible to a direction-indeterminable channel with respect to adaptive adversary under any master schedule.)*

The proof of theorem 2 is described like theorem 1. We omit the proof in this paper, see the full paper version.

5 Equivalence Between DIC and SC

In this section, we prove that the direction indeterminable channel (DIC) is equivalent to secure channel (SC) under a specific type of some schedules. That is, the task-PIOA of DIC perfectly implements task-PIOA of SC under an asynchronous schedule. To prove this, we show two reductions of SC to DIC and one of DIC to SC. Here, we consider the one bit message exchange, that is, $|m| = 1$. Informally, the reduction of SC to DIC is proven as follows: To make the channel between Init and Rec secure, the parties exchange a random bit (as a secret shared key) by DIC. The encrypted message by the shared key is exchanged using a public channel. The communication is done not by a DIC channel but by a public channel. When the next message sending is occured, party restart from key exchange. Here, the key exchange is done under the master schedule. After the key exchange, the cipher text generated by the secret key is sent. The other reduction of DIC to SC is proven as follows: the parties Init and Rec exchange two messages by SC. The one is the message m which the sender wants to send. The other message is a dummy message to conceal the message direction. That is, sender Init sends message m and the receiver sends dummy message s under a specific type of schedules by M. We make a random message s by $\mathrm{F}_{\mathrm{SRC}}$. Note that, the adversary cannot know the direction of message because the messages are exchanged under a specific type of schedules. In this section, we need to consider the schedules (key exchange schedule and message exchange schedule) to avoid some infromation to adversary. In the UC framework, all schedule is under control of adversary. So, we use task PIOA framework.

5.1 Reduction of SC to DIC

Let n be the number of parties. Let $M_{psync}(t_1^*, \cdots, t_n^*)$ and $M_{rasync}(t_1^*, \cdots, t_n^*)$ be master schedules, respectively, where t_i^* is a task in party P_i.

Definition 10. **[$M_{psync}(t_1^*, \cdots, t_n^*)$]** *Let t_i^* be a task in party P_i. Let $ptask(t_i^*)$ be the task just before t_i^* in the local scheduler ρ_i. For example, let $\rho_i = t_{i1}, t_{i2}, t_{i3}$ for party P_i. Then $ptask(t_{i3})$ is the task t_{i2}.*

- *1. Alignment property: After the master scheduler M activates $ptask(t_i^*)$, M does not activate P_i until all of $ptask(t_1^*), \cdots, ptask(t_n^*)$ are scheduled. This situation say that M satisfies the alinment property for the specified tasks $t_1^*, \ldots, t_n^*$.*
- *2. Random executing property: The master scheduler, M, grobally executes the specified tasks, $t_1^*, \ldots, t_n^*$ in a random order. Note that the other tasks are not scheduled until all of the specified tasks, $t_1^*, \ldots, t_n^*$, finish executing.*

$M_{psync}(t_1^*, \cdots, t_n^*)$ is defined to be a master schedule such that a master scheduler M satisfies the avobe mentioned two properties for the specified tasks $t_1^*, \ldots, t_n^*$.

Definition 11. **[$M_{rasync}(t_1^*, \cdots, t_n^*, k)$]** *Let k be a integer. Let t_i^* be a task specified by ρ_i for party P_i. Let c_i be the number of times t_i^* is scheduled by M. M schedules the task acctivations of $t_1^*, \cdots, t_n^*$ so that $|c_i - c_j| \leq k$ for all i, j in a random order.*

We need to consider like chernov bound property if we treat this master schedule to use the message exchange or key exchange among party.

Let π_{SC} be a protocol of secure channel. Let $M_{\pi_{\mathrm{SC}}}$ be $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F_{DIC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F_{DIC}}})$. Let $\mathrm{Init_{SC}}$ and $\mathrm{Rec_{SC}}$ be the initiator's code and receiver's code for a real system, respectively. Let $\overline{\mathrm{Init_{DIC}}}$ and $\overline{\mathrm{Rec_{DIC}}}$ be the initiator's code and receiver's code for an ideal system, respectively. Finally, let $\mathrm{Adv_{SC}}$, $\mathrm{Sim_{SC}}$ and $\mathrm{F_{SRC}}$ be the adversary's code, the simulator's code and the random bit generator's code, respectively. Let $\mathbf{Real_{SC}}$ and $\mathbf{Ideal_{SC}}$ be a secure channel protocol system and a secure channel functionality system, respectively, defined as follows:

$$\mathbf{Real_{SC}} := \mathrm{Init_{SC}}||\mathrm{Rec_{SC}}||\mathrm{Adv_{SC}}||\mathrm{F_{SRC}}||\mathrm{F_{DIC}},$$
$$\mathbf{Ideal_{SC}} := \overline{\mathrm{Init_{SC}}}||\overline{\mathrm{Rec_{SC}}}||\mathrm{Sim_{SC}}||\mathrm{F_{SC}}.$$

Tasks $\overline{\mathrm{Init_{SC}}}$ and $\overline{\mathrm{Rec_{SC}}}$ relay the input messages from the environment to the ideal functionality task and relay the receive messages from the ideal functionality task to the environment, respectively, as interface parties in the ideal system. Several codes for each tasks are omitted in this paper, see full paper version.

Theorem 3. *Secure channel protocol system* $\mathbf{Real_{SC}}$ *perfectly hybrid implements secure channel functionality system* $\mathbf{Ideal_{SC}}$ *with respect to adaptive adversary under a master schedule* $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, s)_{\mathrm{F_{DIC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F_{DIC}}})$. *(A secure channel is reducible to a direction-indeterminable channel with respect to adaptive adversary under a master schedule* $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}s)_{\mathrm{F_{DIC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathtt{DIC}}, t)_{\mathrm{F_{DIC}}})$.*)*

The proof of theorem 3 is described like theorem 1. We omit the proof in this paper, see the full paper version. The master schedule can be M_{rasync} instead of M_{psync}.

5.2 Reduction of DIC to SC

Let π'_{DIC} be a protocol of direction-indeterminable channel. Let $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F_{SC}}}, send(\mathtt{Send}, \mathtt{sid}_{\mathrm{SC}}, m)_{\mathrm{F_{SC}}})$ be the master schedule for π'_{DIC}.

Let $\mathrm{Init'_{DIC}}$ and $\mathrm{Rec'_{DIC}}$ be the initiator's code and receiver's code for a real system, respectively. Let $\overline{\mathrm{Init'_{DIC}}}$ and $\overline{\mathrm{Rec'_{DIC}}}$ be the initiator's code and receiver's code for an ideal system, respectively. Finally, let $\mathrm{Adv'_{DIC}}$ and $\mathrm{Sim'_{DIC}}$ be the adversary's code and the simulator's code, respectively. Let $\mathbf{Real'_{DIC}}$ and $\mathbf{Ideal'_{DIC}}$ be a direction-indeterminable channel protocol system and a direction-indeterminable channel functionality system defined, respectively, as follows:

$$\mathbf{Real'_{DIC}} := \mathrm{Init'_{DIC}}||\mathrm{Rec'_{DIC}}||\mathrm{Adv'_{DIC}}||\mathrm{F_{SRC}}||\mathrm{F_{SC}},$$
$$\mathbf{Ideal'_{DIC}} := \overline{\mathrm{Init'_{DIC}}}||\overline{\mathrm{Rec'_{DIC}}}||\mathrm{Sim'_{DIC}}||\mathrm{F_{DIC}}.$$

Tasks $\overline{\mathrm{Init'_{DIC}}}$ and $\overline{\mathrm{Rec'_{DIC}}}$ relay the input messages from the environment to the ideal functionality task and relay the receive messages from the ideal functionality task to the environment, respectively, as interface parties in the ideal system. Several codes for each tasks are omitted in this paper, see full paper version.

Theorem 4. *Direction-indeterminable channel protocol system* $\mathbf{Real'}_{\mathbf{DIC}}$ *perfectly hybrid implements direction-indeterminable channel functionality system* $\mathbf{Ideal'}_{\mathbf{DIC}}$ *with respect to adaptive adversary under a master schedule* $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{SC}, m)_{F_{SC}}, send(\mathtt{Send}, \mathtt{sid}_{SC}, m)_{F_{SC}})$. *(A direction-indeterminable channel is reducible to a secure channel with respect to adaptive adversary under a master schedule* $M_{psync}(send(\mathtt{Send}, \mathtt{sid}_{SC}, m)_{F_{SC}}, send(\mathtt{Send}, \mathtt{sid}_{SC}, m)_{F_{SC}})$.*)*

The proof of theorem 4 is described like theorem 1. We omit the proof in this paper, see the full paper version.

6 Conclusion

This paper studied the relationship of the three cryptographic channels, secure channels (SC), two-anonymous channels (2AC) and direction-indeterminable channels (DIC), by considering communication schedules and composable security. For this purpose, we adopted the universally composable (UC) framework with the task-probabilistic input/output automata (PIOA) model. We showed that the three channels are reducible to each other under some types of schedules in the UC framework with the PIOA model.

References

1. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. Proc. of STOC, pp.1–10, 1988.
2. R. Canetti. Universally Composable Security: A New paradigm for Cryptographic Protocols. 42nd FOCS, 2001. IACR ePrint Archive 2000/067, http://eprint.iacr.org.
3. R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Taskstructured probabilistic I/O automata. Proc. of WODES'06, 2006.
4. R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Time-bounded task-PIOAs: a framework for analyzing security protocols. Proc. of DISC'06, 2006.
5. R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Using probabilistic I/O automata to analyze an oblivious transfer protocol. Technical Report MIT-CSAIL-TR-2006-046, CSAIL, MIT, 2006. This is the revised version of Technical Reports MIT-LCS-TR-1001a and MIT-LCS-TR-1001.
6. R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Using Task-Structured Probabilistic I/O Automata to Analyze an Oblivious Transfer Protocol. This is a revised version of Technical Report MIT-CSAIL-TR-2006-046, http://eprint.iacr.org.
7. D. Chaum, C. Crépeau, and I. Damgård. Multiparty Unconditionally Secure Protocols. Proc. of STOC, pp.11–19, 1988.
8. J.Håstad. Pseudo-Random Generators under Uniform Assumptions. Proc. of STOC, 1990.
9. R. Impagliazzo, L. Levin and M. Luby. Pseudo-Random Number Generation from One-Way Functions. Proc. of STOC, pp.12–24, 1989.
10. M.Naor. Bit Commitment Using Pseudo-Randomness. Proc. of Crypto'89, LNCS 435, Springer–Verlag, pp.128–136, 1990.
11. M.Naor, and M. Yung. Universal One-Way Hash Functions and Their Cryptographic Applications. Proc. of STOC, pp.33–43, 1989.
12. T.Okamoto. On the Relationship among Cryptographic Physical Assumptions. ISAAC'93, LNCS 762, Springer Verlag, pp. 369-378, 1993.
13. J. Rompel. One-Way Functions are Necessary and Sufficient for Secure Signature. Proc. of STOC, pp.387–394, 1990.