Modular Reasoning about Differential Privacy in a Probabilistic Process Calculus

Lili Xu, Catuscia Palamidessi

Modular Reasoning about Differential Privacy in a Probabilistic Process Calculus

Lili Xu[1,2,3] and Catuscia Palamidessi[1]

[1] INRIA and LIX, École Polytechnique, France
[2] State Key Lab. of Comp. Sci., Institute of Software, Chinese Academy of Sciences
[3] Graduate University, Chinese Academy of Sciences

Abstract. Differential privacy is a notion of privacy originated from the community of statistical databases, and now widely adopted for the protection of confidential information in systems of various nature. We consider a probabilistic process calculus as a specification formalism for concurrent systems, and we establish a framework for reasoning about the degree of differential privacy provided by such systems. We give a compositional method to compute the conditional probabilities that relate the secret and the public information, and we investigate the constructs which do not decrease the degree of privacy under composition.

Keywords: Differential Privacy, Probabilistic Process Calculi, Crowds Protocol

1 Introduction

The most recent developments and usages of information technologies such as data profiling in databases, or user tracking in pervasive computing, pose serious threats to the confidential information of the users. For instance, the social networks Twitter and Flickr carefully protect their user's data by anonymization, and yet Narayanan and Smatikov [19] were able to conceive a de-anonymization algorithm which could re-identify 30% of the people who have accounts in both of them, with only a 12% error rate.

Differential privacy [11, 13, 12] is a recent approach to protect confidential information. Originally emerged from the field of statistical databases, it is now widely adopted in many other models of computations (see for instance [20, 1]), including cloud computing [22]. In general, the aim is to make available global information in some domain of interest, while the sensitive data of the particular individual are kept confidential. Unfortunately, computation on collective data may leak information about an individual.

To avoid this problem, one of the most used methods consists in introducing some *random noise* on the answer. In other words, instead of giving the *exact* answer the system gives an *approximated* answer, chosen randomly according to some probability distribution. Differential privacy is a measure of the level of protection: we say that the system is ϵ-differentially private if for every pair of *adjacent* datasets (i.e. datasets which differ for the data of an individual only), the probabilities of obtaining a certain answer differ at most by e^{ϵ}.

In this paper, we consider an extended notion of differential privacy, based on a generic notion of adjacency, and we address the problem of computing the degree ϵ of privacy provided by a system. To represent such systems we use a probabilistic version of CCS. Our main results are

- a compositionality property, stating how certain operators affect the degree of differential privacy, and
- an algorithm to compute bottom-up the channel matrix of a system, and its minimum degree of differential privacy,
- an application of the compositionally result to prove an anonymity property in an extension of Crowds. The extension consists in allowing each member to have a set of trusted users to which they can forward the message. This breaks the symmetry property of Crowds, and makes the standard analyses from the literature unapplicable. In contrast, our compositional method gives a very simple proof.

Structure of the paper. In Section 2 we review some preliminaries used throughout this paper. In Sections 3 and 4 we present the probabilistic process calculus CCS$_p$ and we define a notion of differential privacy for terms of the calculus. In section 5 we show how the differential privacy degree of a security system is affected under the compositions of the CCS$_p$ operators. In Section 6, we propose an approach for computing automatically the channel matrix (and the degree of privacy) of a finite process, in a bottom-up way. In Section 7, we apply our compositional method to show the anonymity of the Crowds protocol without the symmetry requirement. In Section 8 we discuss related work. Finally, in Section 9 we conclude and discuss some future work.

2 Preliminaries

We recall some basic definitions about probability theory and probabilistic automata.

2.1 Probability measure

Let Ω be a set. A σ-$field$ over Ω is a collection $\mathcal{F}$ of subsets of Ω containing $\emptyset$ and closed under complements and countable unions. A *probability measure* on a σ-field $\mathcal{F}$ is a function $\mu : \mathcal{F} \to [0, 1]$ such that $\mu(\Omega) = 1$ and, for each family $\{Z_i\}_{i\in\mathbb{N}}$ of pairwise disjoint elements of $\mathcal{F}$, $\mu(\bigcup_{i\in\mathbb{N}} Z_i) = \sum_{i\in\mathbb{N}} \mu(Z_i)$. A *discrete probability measure* over Ω is a probability measure whose σ-field is the powerset of Ω. We denote the set of all discrete probability measures over a set X by $Disc(X)$. For $x \in X$, we denote by $\delta(x)$ (the *Dirac measure* on x) the probability measure that assigns probability 1 to $\{x\}$.

2.2 Probabilistic automata

We recall the formalism of *probabilistic automata* [24, 25], to be used as operational semantics for the probabilistic CCS that will be introduced in the next section.

A (*simple*) *probabilistic automaton* is a tuple $\mathcal{M} = (\mathcal{P}, P_{init}, Act, \mathcal{T})$ where $\mathcal{P}$ is a set of states, $P_{init} \in \mathcal{P}$ is the *initial state*, Act is a set of labels and $\mathcal{T} \subseteq \mathcal{P} \times Act \times Disc(\mathcal{P})$ is a *transition relation*. Informally, if $(P, a, \mu) \in \mathcal{T}$ then there is a transition from the state P performing a label labeled a and then leading to a distribution μ over a set of states instead of a single state. Intuitively, the idea is that the transition in $\mathcal{T}$ is chosen nondeterministically, and the target state among the ones allowed by μ is chosen probabilistically.

A *fully probabilistic* automaton is a probabilistic automaton without nondeterminism, at each state only one transition can be chosen.

An *execution* α of a probabilistic automaton is a (possibly infinite) sequence

$$P_{init} a_0 P_1 a_1 P_2 a_2 P_3 \cdots$$

of alternating states and labels, and for each i, a transition $(P_i, a_i, \mu_i) \in \mathcal{T}$ and $p_i = \mu_i(P_{i+1}) > 0$ hold. We will use

- $exec^*(\mathcal{M})$ to represent the set of all the $\mathcal{M}$'s finite executions,
- $exec(\mathcal{M})$ to represent the set of all the executions of $\mathcal{M}$,
- and $lst(\alpha)$ to denote the last state of a finite execution $\alpha \in exec^*(\mathcal{M})$.

A *execution tree* of $\mathcal{M}$, denoted by $etree(\mathcal{M})$, is an automaton $\mathcal{M}' = (\mathcal{P}', P'_{init}, Act, \mathcal{T}')$ such that $\mathcal{P}' \subseteq exec(\mathcal{M})$, $P'_{init} = P_{init}$, and $(\alpha, a, \mu') \in \mathcal{T}'$ if and only if $(lst(\alpha), a, \mu) \in \mathcal{T}$ for some μ and $\mu'(\alpha a P) = \mu(P)$ for all P. Intuitively, $etree(\mathcal{M})$ is produced by unfolding all the executions of $\mathcal{M}$.

A *scheduler* of a probabilistic automation $\mathcal{M}$ is a function

$$\zeta : exec^*(\mathcal{M}) \to \mathcal{T}$$

such that $\zeta(\alpha) = (P, a, \mu) \in \mathcal{T}$ implies that $P = lst(\alpha)$. The idea is that a scheduler resolves the nondeterminism by selecting a transition among the ones available in $\mathcal{T}$, basing its decision on the history of the execution.

The *execution tree* of $\mathcal{M}$ *relative to the scheduler* ζ, denoted by $etree(\mathcal{M}, \zeta)$, is a fully probabilistic automaton $\mathcal{M}'' = (\mathcal{P}'', P''_{init}, Act, \mathcal{T}'')$ which can be obtained from $\mathcal{M}'$ by removing all the transitions in $\mathcal{T}'$ that are not chosen by the scheduler, that is, $(\alpha, a, \mu'') \in \mathcal{T}''$ if and only if $\zeta(\alpha) = (lst(\alpha), a, \mu)$ for some μ and $\mu''(\alpha a P) = \mu(P)$ for all P. Intuitively, $etree(\mathcal{M}, \zeta)$ is produced from $etree(\mathcal{M})$ by resolving all nondeterministic choices using ζ. Note that $etree(M, \zeta)$ is a simple and fully probabilistic automaton.

3 Probabilistic CCS

In this section we present a probabilistic version of Milner's CCS [17], similar to the one considered in [8], that allows for both nondeterministic and probabilistic choice. Following [4] we make a distinction between *observable* and *secret* labels, for the purpose of specifying security systems and protocols.

We consider a countable set Act of labels a, partitioned into a set Sec of *secret labels* s, a set Obs of *observable labels* o, and the silent action τ. For each $o \in Obs$,

$$\text{PROB}\ \frac{}{\sum_i p_i P_i \xrightarrow{\tau} \sum_i p_i \delta(P_i)} \qquad \text{ACT}\ \frac{j \in I}{\lfloor\!+\!\rfloor_I a_i.P_i \xrightarrow{a_j} \delta(P_j)}$$

$$\text{PAR1}\ \frac{P_1 \xrightarrow{a} \mu}{P_1 \,|\, P_2 \xrightarrow{a} \mu \,|\, P_2} \qquad \text{PAR2}\ \frac{P_2 \xrightarrow{a} \mu}{P_1 \,|\, P_2 \xrightarrow{a} P_1 \,|\, \mu} \qquad \text{REP}\ \frac{P \,|\, !\,P \xrightarrow{a} \mu}{!\,P \xrightarrow{a} \mu \,|\, !\,P}$$

$$\text{COM}\ \frac{P_1 \xrightarrow{a} \delta(P_1') \quad P_2 \xrightarrow{\overline{a}} \delta(P_2')}{P_1 \,|\, P_2 \xrightarrow{\tau} \delta(P_1' \,|\, P_2')} \qquad \text{RES}\ \frac{P \xrightarrow{b} \mu \qquad b \neq a, \overline{a}}{(\nu a)P \xrightarrow{b} (\nu a)\mu}$$

Table 1: The semantics of CCS_p.

we assume a complementary label $\overline{o} \in \mathit{Obs}$ with the proviso that $\overline{\overline{a}} = a$. The syntax of CCS_p is:

$P ::=$		*process term*
	$\sum_i p_i P_i$	*probabilistic choice*
	$\lfloor\!+\!\rfloor_i\, s_i.P_i$	*secret choice* ($s_i \in \mathit{Sec}$)
	$\lfloor\!+\!\rfloor_i\, r_i.P_i$	*nondeterministic choice* ($r_i \in \mathit{Obs} \cup \{\tau\}$)
	$P \,|\, P$	*parallel composition*
	$(\nu a)P$	*restriction*
	$!\,P$	*replication*

The term $\sum_i p_i P_i$ represents an *blind probabilistic choice*, in the sense that the choice of the branch is decided randomly (according to the corresponding probability) and there is no visible label associated to the decision. We use the notation $P_1 \oplus_p P_2$ to represent a binary sum with $p_1 = p$ and $p_2 = 1-p$. Similarly, we use $a_1.P_1 \lfloor\!+\!\rfloor\ a_2.P_2$ to represent a binary secret or nondeterministic choice. Finally the term $\mathbf{0}$, representing the terminated process, is syntactic sugar for an empty (secret or nondeterministic) choice.

The operational semantics of a CCS_p term P is a probabilistic automaton whose states $\mathcal{P}$ are the processes reachable from P, and the transition relation is defined according to the rules in the Table 1, where for aesthetic reasons we use $P \xrightarrow{a} \mu$ to represent the transition (P, a, μ). (We will use these two notations interchangeably.) We denote by $\mu \,|\, P$ the measure μ' such that $\mu'(P' \,|\, P) = \mu(P')$ for all processes $P' \in \mathcal{P}$ and $\mu'(P'') = 0$ if P'' is not of the form $P' \,|\, P$. Similarly $(\nu a)\mu = \mu'$ such that $\mu'((\nu a)P) = \mu(P)$. A transition of the form $P \xrightarrow{a} \delta(P')$, i.e. a transition having for target a Dirac measure, corresponds to a transition of a non-probabilistic automaton. From the rule PROB, we know that all transitions to non-Dirac measures are silent.

Following [4] we assume the secret labels to be the *inputs* of the system. Secrets are given in input to the scheduler and determine completely the secret choices. The

scheduler then has to resolve the residual nondeterminism, which is originated by the nondeterministic choice and the parallel operator. From the observer's point of view, only the nondeterministic choices can be observed.

The definition of a scheduler of a CCS$_p$ term is specified more precisely as follows. The notation $X \rightharpoonup Y$ represents the partial functions from X to Y, and $\alpha_{|Sec}$ represents the projection of α on Sec.

Definition 1. *Let P be a process in CCS$_p$ and $\mathcal{M}$ be the probabilistic automaton generated by P. A scheduler is a function $\zeta : Sec^* \to exec^* \rightharpoonup \mathcal{T}$ such that:*

(i) if $\boldsymbol{s} = s_1 s_2 \dots s_n$ and $\alpha_{|Sec} = s_1 s_2 \dots s_m$ with $m \leq n$, and
(ii) there exists a transition $(lst(\alpha), a, \mu)$ such that, if $a \in Sec$ then $a = s_{m+1}$

then $\zeta(\boldsymbol{s})(\alpha)$ is defined, and it is one of such transitions. We will write $\zeta_{\boldsymbol{s}}(\alpha)$ for $\zeta(\boldsymbol{s})(\alpha)$.

We now define the *execution tree* of a CCS$_p$ term, in a way similar to what is done in probabilistic automata. The main difference is that in our case the execution tree depends not only on the scheduler, but also on the secret input.

Definition 2. *Let $\mathcal{M} = (\mathcal{P}, P_{init}, Act, \mathcal{T})$ be the probabilistic automaton generated by a CCS$_p$ process P, where $\mathcal{P}$ is the set of processes reachable from P. Given an input $\boldsymbol{s}$ and a scheduler ζ, the* execution tree *of P, denoted by $etree(P, \boldsymbol{s}, \zeta)$, is a fully probabilistic automaton $\mathcal{M}' = (\mathcal{P}', P_{init}, Act, \mathcal{T}')$ such that $\mathcal{P}' \subseteq exec(\mathcal{M})$, and $(\alpha, a, \mu') \in \mathcal{T}'$ if and only if $\zeta_{\boldsymbol{s}}(\alpha) = (lst(\alpha), a, \mu)$ for some μ, and $\mu'(\alpha a P) = \mu(P)$.*

Process terms as channels We now show how CCS$_p$ terms can be used to specify systems manipulating confidential information.

A system can be seen as an information-theoretic channel [7]. Sequences of secret labels constitute the *secret information* (or *secrets*), given in input to the channel, and sequences of observable labels constitute the *public information* (or *observables*), obtained as output from the channel. We denote secrets and observables by $\mathcal{S}$ and $\mathcal{O}$, and we assume that they have finite cardinality m and n, respectively. We also assume that each sequence in $\mathcal{S} \cup \mathcal{O}$ is finite. Thus, $\mathcal{S} \subseteq_{fin} Sec^*$ and $\mathcal{O} \subseteq_{fin} Obs^*$. This is enough to model the usual situations in security, where each session is supposed to terminate in finite time. Finally, we assume a discrete probability distribution over the inputs, which we will denote by $\boldsymbol{\pi} = (\pi_1, \pi_2, \dots, \pi_m)$, where π_i is the probability of the input $\boldsymbol{s}_i$.

Given an input $\boldsymbol{s} \in \mathcal{S}$, a run of the system will produce each $\boldsymbol{o} \in \mathcal{O}$ with a certain probability $p(\boldsymbol{o}|\boldsymbol{s})$ which depends on $\boldsymbol{s}$ and on the randomized operations performed by the system, and also on the scheduler resolving the nondeterminism. The probabilities $p(\boldsymbol{o}|\boldsymbol{s})$, for a given scheduler ζ, constitute a $m \times n$ array M_ζ which is called the *matrix* of the channel, where the rows are indexed by the elements of $\mathcal{S}$ and the columns are indexed by the elements of $\mathcal{O}$.

Definition 3 ([4]). *Given a term P and a scheduler $\zeta : \mathcal{S} \to exec^* \to \mathcal{T}$, the matrix $M_\zeta(P)$ associated to P under ζ is defined as the matrix such that, for each row $\boldsymbol{s} \in \mathcal{S}$ and column $\boldsymbol{o} \in \mathcal{O}$, the element at their intersection, $p_\zeta(\boldsymbol{o}|\boldsymbol{s})$, is the probability of the set of the maximal executions in $etree(P, \boldsymbol{s}, \zeta)$ whose projection in Obs is $\boldsymbol{o}$.*

4 Differential Privacy

Differential Privacy [11] captures the idea that a query on a dataset does not provides too much information about a particular individual, regardless of whether the individual's record is in the dataset or not. In order to achieve this goal, typically some probabilistic noise is added to the answer. The formal definition is the following (where κ denotes the randomized answer, $\Pr$ the probability measure, and ϵ a finite non-negative number):

Definition 4 (Differential Privacy, [11]). *A mechanism κ provides* ϵ-differential privacy *iff for all datasets D_1 and D_2 differing for only one record, and for all $S \subseteq Range(\kappa)$,*

$$\Pr[\kappa(D_1) \in S] \leq e^{\epsilon} \Pr[\kappa(D_2) \in S]$$

Clearly, the smaller the value ϵ is, the higher is the confidentiality protection provided by the mechanism.

We now adapt the notion of differential privacy to our framework. We consider a symmetric adjacency relation $\sim$ between secrets, which extends the dataset-based notion of "differing for only one record" to more general settings.

Example 1 (Anonymity). Anonymity is a particular case of privacy in which the confidential data $\mathcal{S}$ are the agents' identities. If the identities are just names without any particular structure, it is natural to assume that each name is adjacent to any other. Hence $(\mathcal{S}, \sim)$ is a clique, i.e. for all $\boldsymbol{s}_1, \boldsymbol{s}_2 \in \mathcal{S}$ we have $\boldsymbol{s}_1 \sim \boldsymbol{s}_2$.

Example 2 (Geolocation). In the case of geolocation, the confidential data are the coordinates $(latitude, longitude)$ of a point on the earth's surface. If the purpose is to protect the exact location, a good definition of adjacency is: two points are adjacent if their Manhattan distance is 1, i.e. $(x_1, y_1) \sim (x_2, y_2)$ iff $|x_2 - x_1| = 1$ or $|y_1 - y_2| = 1$.

It can be proved that if the set of answers is discrete (which is our case) Definition 4 can be equivalently stated in terms of singleton S's. This leads to the following:

Definition 5. *A process P provides* ϵ-differential privacy *iff for all scheduler ζ, for all secret inputs $\boldsymbol{s}_1, \boldsymbol{s}_2 \in \mathcal{S}$ such that $\boldsymbol{s}_1 \sim \boldsymbol{s}_2$, and for all observable $\boldsymbol{o} \in \mathcal{O}$,*

$$p_\zeta(\boldsymbol{o}|\boldsymbol{s}_1) \leq e^{\epsilon}\, p_\zeta(\boldsymbol{o}|\boldsymbol{s}_2)$$

We use $df_\zeta[\![P]\!]$ to denote the smallest ϵ such that the process term P, under the scheduler ζ, provides ϵ-differential privacy. Furthermore we define

$$df[\![P]\!] = \max_\zeta df_\zeta[\![P]\!]$$

Note that if there are both zero and non-zero probabilities occurring in the same column of the matrix, when the respective secrets are connected by the transitive closure of $\sim$, then the process does not provide differential privacy for any ϵ.

Relation between differential privacy and strong anonymity Strong anonymity for purely probabilistic systems was formalized by Chaum [6] as the property that the observation $\boldsymbol{o}$ does not change the probabilistic knowledge of the culprit's identity $\boldsymbol{s}$, i.e.:

$$p(\boldsymbol{s}|\boldsymbol{o}) = p(\boldsymbol{s}) \tag{1}$$

This notion was extended in [2] to probabilistic and nondeterministic systems, essentially by requiring that (1) hold under any scheduler. The following proposition is an immediate consequence of the characterization of strong anonymity given in [2].

Proposition 1. *Under the assumption that $(\mathcal{S}, \sim)$ is a clique, a process P is strongly anonymous iff $df\ [\![P]\!] = 0$.*

5 Modular Reasoning

In this section we investigate the compositional properties of CCS$_p$ constructs with respect to differential privacy and state our first main result.

We start by introducing the notion of *safe component*.

Definition 6. *Consider a process P, and the observable $o_1, o_2, \ldots, o_k$ such that*

(i) P does not contain any secret label, and
(ii) all the observable labels of P are included in $o_1, o_2, \ldots, o_k$.

Then we say that $(\nu o_1, o_2, \ldots, o_k)P$ is a safe component.

We now introduce the notion of sequential replication, which can be used to represent a sequence of sessions re-executing the same protocol.

Definition 7. *Given a process term P assumed to terminate by performing a specific action $\overline{done}$, the* sequential replication *of P n* times *is defined as*

$$\circlearrowleft^n P = (\nu\, done)(P \,|\, !^n done.P)$$

where

$$!^0 P = \mathbf{0} \quad \textit{and} \quad !^{n+1} P = P \,|\, !^n P$$

We now show that the probabilistic choice, the nondeterministic choice, the restriction operator and a restricted form of parallel composition are *safe*, in the sense that they maintain the degree of differential privacy of the components, while the sequential replication degrades the privacy in proportion to the number of replication times.

Theorem 1. *Consider a set of processes $\{P_i\}_i$, for $i = 1, 2, \ldots$, and assume that for each i, $df\ [\![P_i]\!] = \epsilon_i$. Then:*

(1) $df\ [\![\boxplus_i\, o_i.P_i]\!] \leq \max_i\{\epsilon_i\}$;

(2) $df\ [\![\sum_i\, p_i P_i]\!] \leq \max_i\{\epsilon_i\}$;

(3) $df\ [\![(\nu a)P_i]\!] \leq \epsilon_i$;

(4) Assume that $(\nu o_1, o_2, \ldots, o_k)P_i$ is a safe component, that P_i and P_j can communicate with each other only via the labels of the set $\{o_h, \ldots, o_k\}$, with $1 \leq h \leq k$, and that df $[\![(\nu o_1, \ldots, o_{h-1})P_j)]\!] = \epsilon_j$. Then df $[\![(\nu o_1, o_2, \ldots, o_k)\,(P_i \mid P_j)]\!] \leq \epsilon_j$.

(5) df $[\![\bigcirc^n P_i]\!] \leq n\,\epsilon_i$.

This theorem provides a method for modular reasoning about complex systems: rather than analyzing the system as a whole, the safe constructs allow us to split the system in parts, analyze each part's degree of privacy separately, and combine the results to obtain the degree of privacy of the entire system. Unfortunately the other operators do not preserve the degree of privacy, typically due to the presence of nondeterminism. For instance in the secret choice $\boxplus_i\, s_i.P_i$ the nondeterminism in the P_i's gives the scheduler the possibility to choose different observables for each s_i, and therefore make distinctions among these secrets. The following counterexample formalizes this intuition.

Example 3. Let $\mathcal{S} = \{s_1, s_2\}$ with $s_1 \sim s_2$, $\mathcal{O} = \{o_1, o_2\}$, and consider the process $P = o_1.\mathbf{0} \boxplus o_2.\mathbf{0}$. Clearly P provides 0-differential privacy. Consider now a new process $P' = s_1.P \boxplus s_2.P$, and the scheduler ζ for P' which selects o_1 if the secret is s_1, and o_2 if the secret is s_2. The resulting matrix under ζ does not preserve differential privacy since $p(o_1|s_1) = p(o_2|s_2) = 1$ while $p(o_1|s_2) = p(o_2|s_1) = 0$.

An analogous counterexample can be given for the (unrestricted form of) parallel composition.

6 Computing the degree of privacy automatically

Although the results of previous section cannot be used directly to compute compositionally the degree of privacy in the presence of the secret choice or the (unrestricted) parallel operator, the analyses of the other CCS$_p$'s constructs in the proof of the Theorem 1 suggest a way for computing the channel matrix of a system in a bottom-up fashion, from which we obtain its degree of privacy. In this section we present the algorithm to produce such matrix, and give some toy examples to illustrate the approach.

6.1 The algorithm

Consider a process term $T = \boxplus_i\, s_i.T_i$ starting with a secret choice. We assume that there is no more secret choice inside any of the T_i's, and no occurrence of !. We also assume that every occurrence of the parallel operator is removed by using the *expansion law* [18]. Thus the residual constructs possibly occurring in T_i are the nondeterministic choice, the probabilistic choice, the restriction and the sequential replication.

We start by constructing, for a process P, the set of *all the possible schedulers* for P, denoted by $\Delta(P)$. We denote its size by $|\Delta(P)|$.

- Case $P = \sum_i p_i P_i$: Intuitively, P's scheduler selects for each P_i a scheduler from $\Delta(P_i)$. Hence $\Delta(P) = \{\bigcup_i \zeta_i \mid \zeta_i \in \Delta(P_i)\}$, and $|\Delta(P)| = \prod_i |\Delta(P_i)|$.
- Case $P = \boxplus_i\, r_i.P_i$: Let $\Delta_i(P)$ be the set of all the possible schedulers for P when $r_i.P_i$ is chosen to resolve the nondeterminism. It is easy to see that $\Delta_i(P) = \{\{\zeta(\varepsilon) = r_i\} \cup \{\zeta(r_i\alpha) = \zeta_i(\alpha)\} \mid \zeta_i \in \Delta(P_i)\}$. Thus $\Delta(P) = \bigcup_i \Delta_i(P)$, and $|\Delta(P)| = \sum_i |\Delta(P_i)|$.

- Case $P' = (\nu a)P$: Intuitively, the scheduler $\zeta|_a$ is obtained from ζ by removing the assignments of the executions containing a. Hence $\Delta(P') = \{\zeta|_a \mid \zeta \in \Delta(P)\}$, and $|\Delta(P')| = |\Delta(P)|$.
- Case $P' = \circlearrowleft^n P$: Intuitively, the scheduler $\zeta_1; \zeta_2; \cdots; \zeta_n$ selects for each run of P a scheduler from $\Delta(P)$, and use them in a sequential way. Hence $\Delta(P') = \{\zeta_1; \zeta_2; \cdots; \zeta_n \mid \zeta_i \in \Delta(P)\}$, and $|\Delta(P')| = |\Delta(P)|^n$.

For every term T_i in T, we are able to obtain the scheduler set $\Delta(T_i)$. The corresponding matrix under each scheduler in $\Delta(T_i)$ is computed in a bottom-up way (see appendix).

We now turn to the computation of T's matrix and differential privacy degree. For every secret label s_i, T's scheduler ζ chooses from $\Delta(T_i)$ a scheduler ζ_i. Let $p_\zeta(\boldsymbol{o}|s_i\boldsymbol{s})$ (resp. $p^i_{\zeta_i}(\boldsymbol{o}|\boldsymbol{s})$) be the probability in the matrix $M(T)_\zeta$ (resp. $M_{\zeta_i}(T_i)$). Hence

$$p_\zeta(\boldsymbol{o}|s_i\boldsymbol{s}) = p^i_{\zeta_i}(\boldsymbol{o}|\boldsymbol{s}).$$

By definition of differential privacy we get:

$$df_\zeta[\![T]\!] = \min\{\epsilon \mid \boldsymbol{s} \sim \boldsymbol{s'} \Rightarrow p_\zeta(\boldsymbol{o}|\boldsymbol{s}) \leq e^\epsilon p_\zeta(\boldsymbol{o}|\boldsymbol{s'})\}$$

and

$$df[\![T]\!] = \max_{\zeta \in \Delta(T)} df_\zeta[\![T]\!].$$

Complexity Analysis The time complexity of the above algorithm is determined by the size of the set of all the possible schedulers, that is, the time complexity is $O(|\Delta(T)|)$. However we can make it more efficient in the case in which differential privacy does not hold: Whenever we find a scheduler ζ_i in $\Delta(T_i)$ producing an observable $\boldsymbol{o}$ which is not included in the set of observables generated by a previous scheduler ζ_j in $\Delta(T_j)$ (with $j < i$ and s_i, s_j connected by the transitive closure of $\sim$), then we can halt the algorithm and claim that T does not provide differential privacy for any ϵ. In fact, assigning the scheduler ζ_i (resp. ζ_j) to the secret s_i (resp. s_j) differentiates the two secrets by producing a non-null (resp. null) probability in the column of $\boldsymbol{o}$.

6.2 Some toy examples

Here we give some simple examples to illustrate the above algorithm.

Example 4. (Example 3 revisited) Let $\mathcal{S} = \{s_1, s_2\}, \mathcal{O} = \{o_1, o_2\}$, the process $P = o_1.\mathbf{0} \boxplus o_2.\mathbf{0}$ and the process $P' = s_1.P \boxplus s_2.P$.

For the term $o_1.\mathbf{0}$, we have $\Delta(o_1.\mathbf{0}) = \varnothing$ and $M(o_1.\mathbf{0}) =$

o_1	o_2
1	0

.

For the term $o_2.\mathbf{0}$, we have $\Delta(o_2.\mathbf{0}) = \varnothing$ and $M(o_2.\mathbf{0}) =$

o_1	o_2
0	1

.

For the term P, we have $\Delta(P) = \{\zeta_1, \zeta_2\}$ with ζ_1 (resp. ζ_2) representing the choice of o_1 (resp. o_2). The corresponding matrices are $M_{\zeta_1}(P) = M(o_1.\mathbf{0})$ and $M_{\zeta_2}(P) = M(o_2.\mathbf{0})$.

For the term P', we can define a scheduler ζ' which selects ζ_1 and ζ_2 for the secret s_1 and s_2, respectively. Thus the resulting matrix does not provide differential privacy, as $M_{\zeta'}(P') =$

	o_1	o_2
s_1	1	0
s_2	0	1

.

Example 5. Let $\mathcal{S} = \{s_1, s_2\}, \mathcal{O} = \{o_1, o_2\}$, and consider the following processes: $P_1 = o_1.\mathbf{0} \oplus_{0.3} o_2.\mathbf{0}$, $P_2 = o_1.\mathbf{0} \oplus_{0.5} o_2.\mathbf{0}$, $P_3 = o_1.\mathbf{0} \oplus_{0.8} o_2.\mathbf{0}$, $P = P_1 \boxplus P_2 \boxplus P_3$ and $P' = s_1.P \boxplus s_2.P$.

For the term P_1, we have $\Delta(P_1) = \varnothing$ and $M(P_1) =$

o_1	o_2
0.3	0.7

.

For the term P_2, we have $\Delta(P_2) = \varnothing$ and $M(P_2) =$

o_1	o_2
0.5	0.5

.

For the term P_3, we have $\Delta(P_3) = \varnothing$ and $M(P_3) =$

o_1	o_2
0.8	0.2

.

For the term P, we have $\Delta(P) = \{\zeta_1, \zeta_2, \zeta_3\}$ with ζ_i representing the choice of P_i. The corresponding matrices are: $M_{\zeta_1}(P) = M(P_1)$, $M_{\zeta_2}(P) = M(P_2)$ and $M_{\zeta_3}(P) = M(P_3)$.

For the term P' we can define the scheduler ζ' which selects ζ_1 and ζ_3 for the secret s_1 and s_2, respectively. The corresponding matrix is $M_{\zeta'}(P') =$

	o_1	o_2
s_1	0.3	0.7
s_2	0.8	0.2

, which gives $(\ln 3.5)$-differential privacy.

Example 6. Let $\mathcal{S} = \{s_1, s_2\}, \mathcal{O} = \{o, o_1, o_2\}$, and consider the processes $P_1 = o_1.\mathbf{0} \oplus_{0.3} o_2.\mathbf{0}$, $P_2 = o.\mathbf{0}$, $P = P_1 \mid P_2$, and $P' = s_1.P \boxplus s_2.P$.

First we use the expansion law to rewrite P into $(o_1.o.\mathbf{0} \oplus_{0.3} o_2.o.\mathbf{0}) \boxplus (o.o_1.\mathbf{0} \oplus_{0.3} o.o_2.\mathbf{0})$. Through steps similar to the above examples, we can find a scheduler producing a matrix breaking the differential privacy, for example

	o_1o	o_2o	oo_1	oo_2
s_1	0.3	0.7	0	0
s_2	0	0	0.3	0.7

.

7 A case study: the Crowds protocol

In this section we illustrate the Crowds anonymity protocol proposed by Reiter and Rubin in [21]. We apply our compositional result in Theorem 1 to show that the safety of Crowds is maintained when we add a new honest member. The novelty of our proof is that it is simple and yet it does not require the symmetry assumption that is usually made in order to simplify the analysis.

7.1 The Crowds protocol

Crowds is an anonymity protocol which allows Internet users to perform web transactions without revealing their private identity. More specifically, a crowd is a group of n

participants constituted by m *honest members* and $c\ (= n - m)$ *attackers*. The destination of messages, namely the *server*, may also be malicious and collaborate with other attackers.

The way Crowds works is the following:

- When a member, called the *originator*, wants to send a message to the server, instead of sending it directly to the server, she randomly selects a member (possibly herself) in the crowd and she forwards the message to this member.
- Every member who receives the message, with a certain probability p_f randomly selects another member (possibly herself) in the crowd as the new *forwarder* and relays the message to this new forwarder to repeat the same procedure again, or with probability $1 - p_f$ she delivers the message to the server.

In this way, even the message is caught by an attacker, the attacker cannot be sure whether the previous forwarder is the originator or just a forwarder on behalf of somebody else. Members (including attackers) are assumed to have only access to messages routed through them, so that they only know the identities of their immediate predecessors and successors in the path, and of the destination server.

In addition, we assume that the crowd establishes a *reputation* for each member, and that a member has the legal right to initially send requests only if she is *reputed* (i.e. her reputation level is above a certain threshold). This is not a strong restriction since it is not uncommon that when a new agent wants to join a web conversation, its purpose needs to be examined for a while before it is allowed to be a totally legal member.

The protocol expressed in CCS_p is stated in the Table 2. For simplicity, we introduce a notation for value-passing in CCS_p, following standard lines.

$$\begin{array}{rl} \mathit{Input} & x(i).P = \lfloor\!+\!\rfloor_j\, x_j.P[j/i] \\ \mathit{Output} & \bar{x}\langle i\rangle = \bar{x}_i \end{array}$$

We use $\mathcal{H}$, $\mathcal{H}_r$ and $\mathcal{A}$ to denote the set of honest members, of reputed honest members and of attackers, respectively. $\mathcal{C} = \mathcal{H} \cup \mathcal{A}$ representing the set of all participants in the crowd, and we use $x_1, x_2, \ldots, x_n$ to range over $\mathcal{C}$. The first choice operator $\lfloor\!+\!\rfloor_{x_o \in \mathcal{H}_r}\, act_o$ in the process Origin represents a secret choice, with the originator x_o activating the Crowds mechanism. Once an attacker receives a message from an honest member, it will announce the identity of this member through the observable $\overline{detect}$. For simplicity we assume that it will terminate after reporting the first detection. The reason is that by forwarding the message after the first detection the attackers cannot gain more useful information. Hence at most one honest member can be detected. Precisely, the set of secret labels $\mathcal{S} = \{\, act_o \,|\, x_o \in \mathcal{H}_r \}$ and the set of observable labels $\mathcal{O} = \{\, \overline{detect}\langle i\rangle \,|\, x_i \in \mathcal{H} \,\}$.

The difference between the *symmetric case* and the *non-symmetric case* lies in the point when a new forwarder is chosen. In the first case the current forwarder selects an arbitrary member from the whole crowd as the next forwarder. In the *non-symmetric case*, the current member only selects the next forwarder from the subset of the crowd members which she thinks are trustable. We denote by $\mathcal{T}_i$ the subset of crowd members which the ith honest member trusts. Then the process terms for the non-symmetric crowd are similar with the terms showed in the Table 2, except that the process Origin and the process $honest_i$ are modified as shown in the Table 3.

$$
\begin{aligned}
Origin &= \lfloor + \rfloor_{x_o \in \mathcal{H}_r} act_o.\, \lfloor + \rfloor_{x_j \in \mathcal{C}}\, \bar{x}_j\langle o\rangle \\
honest_i &= x_i.(\lfloor + \rfloor_{x_j \in \mathcal{C}}\, \bar{x}_j\langle i\rangle.\overline{done} \oplus_{p_f} \overline{ser}\langle i\rangle.\overline{done}) \\
Honest_i &= \circlearrowleft\, honest_i \\
Attacker_i &= x_i(j).\overline{detect}\langle j\rangle \\
Server &= ser(j).\overline{detect}\langle j\rangle \\
crowd_n &= Server \mid Origin \mid \prod_{x_i \in \mathcal{H}} Honest_i \mid \prod_{x_j \in \mathcal{A}} Attacker_j \\
Crowd_n &= (\nu ser_{1,\ldots,n})(\nu x_1, x_2, \ldots, x_n) crowd_n
\end{aligned}
$$

Table 2: The CCS$_p$ code for Crowds with n members in the symmetric case.

$$
\begin{aligned}
Origin &= \lfloor + \rfloor_{x_o \in \mathcal{H}_r} act_o.\, \lfloor + \rfloor_{x_j \in \mathcal{T}_o}\, \bar{x}_j\langle o\rangle \\
honest_i &= x_i.(\lfloor + \rfloor_{x_j \in \mathcal{T}_i}\, \bar{x}_j\langle i\rangle.\overline{done} \oplus_{p_f} \overline{ser}\langle i\rangle.\overline{done})
\end{aligned}
$$

Table 3: The new definitions for the non-symmetric case.

Based on the symmetry property provided by the standard symmetric case, it has been proved in [21] that Crowds protocol provides probable innocence under certain conditions. In [5] the following equivalent definition of probable innocence is proposed, where $p(o|s)$ denotes the probability of the event that the member o is the one first detected by attackers when the originator is s.

Definition 8 ([5]). *Crowds satisfies probable innocence if for all users o, i, j,*

$$(m-1)p(o|i) \geq p(o|j)$$

where m is the number of honest members.

It is easy to see that the definition can be written in the form of differential privacy.

Remark 1 (Relation with probable innocence). A symmetric crowd satisfies probable innocence if and only if it provides $\ln(m-1)$-differential privacy.

7.2 The non-symmetric case

Without the symmetry property, the directly analysis of the degree of differential privacy provided by Crowds becomes a tough task, especially when there are a large number of members participating in the crowd and the sets of trustable members according to each individual user are arbitrary. The modular reasoning method proposed in the previous section enables us to easily obtain an anonymity preservation result about the non-symmetric case.

Consider the scenario in which there exists a crowd with n members (shown in Table 2). Assume that from a previous analysis we already know that this crowd provides a

$$crowd_{n+1} = crowd_n \mid Honest_{n+1}$$

$$Crowd_{n+1} = (\nu ser_{1,\ldots,n+1})(\nu x_1, x_2, \ldots, x_{n+1}) crowd_{n+1}$$

Table 4: CCS$_p$ code expressing the addition of the $n+1$th member.

certain level of differential privacy. Assume that a new honest member is allowed to join the crowd but it does not enjoy the reputation to be an originator right away. Old members in the crowd can decide by themselves to trust this member or not, which means there is no restriction of how this new member is added, thus resulting in a non-symmetric crowd. Applying the result in Section 5 we can ensure that the privacy of the crowd will not be diminished by this addition.

Theorem 2. *The privacy of a crowd is preserved by the addition of a non-reputed honest member. More precisely, df* $[\![Crowd_{n+1}]\!] \leq$ *df* $[\![Crowd_n]\!]$ *.*

Proof. The addition of a new honest member to a crowd with n participants is presented in the Table 4. It is mainly a parallel composition of the original crowd's process term $crowd_n$ and the new member's process term $Honest_{n+1}$. In the original $crowd_n$ although there is no entity of $Honest_{n+1}$, we assume that the identity x_{n+1} is already available in the set $\mathcal{C}$ as a free label, to be selected as the forwarder.

This new member is known not to be an originator (because she is not reputed). Therefore her presence will not induce an addition of the secret label act_{n+1}. In the sense that the process $Honest_{n+1}$ does not contain any secret labels. Clearly

$$(\nu ser_{1,\ldots,n+1})(\nu x_1, x_2, \ldots, x_{n+1}) Honest_{n+1}$$

is a safe component. Note that the observables in $Honest_{n+1}$ through which she communicates with $crowd_n$ are $\{x_{n+1}\} \cup \{\overline{ser}\langle n+1\rangle\} \cup \{\bar{x}_j\langle n+1\rangle | x_j \in \mathcal{T}_{n+1}\}$. They are free in $Crowd_n$ while are restricted in $Crowd_{n+1}$. By Theorem 1(4), $Crowd_{n+1}$ provides at least as much privacy as $Crowd_n$.

8 Related work

- *Compositionality properties of probabilistic process calculi.* Compositional methods for probabilistic process calculi have been intensively investigated in the field of formal verification of concurrency [15, 8, 9, 4]. The most related works, which study compositional methods specific for security protocols are [9] and [4]. In [9] Deng et al. used the notion of relative entropy to measure privacy and proved that, based on this measuring method, all the constructs in the CCS$_p$ except parallel composition are non-expansive. In [4] Braun et al. proved that the safety measured by Bayes risk is maintained under the composition of CCS$_p$ constructs with a restricted form of parallel composition and without the secret choice. The compositionally results in our paper are closely related to those of [4], although we use a different measure of protection (differential privacy).

- *Compositionality of Differential Privacy.* The way in which the composition of queries affects the degree of differential privacy has been investigated in [16]. In that paper, the author has proved that the degree of privacy adds up when queries are applied to the same dataset, and is preserved when the queries are applied to disjoint datasets. Our result about the sequential composition in CCS_p is reminiscent of the first of the above results, while for the other operators the correspondence is not so clear.

- *Other extensions on the Crowds protocol.* There are several anonymity results for Crowds, both theoretical [21, 5, 10, 26] and experimental [3]. All of these assume some form of symmetry of the Crowds protocol . To the best of our knowledge, this paper is the first to investigate the Crowds protocol without any dependence on the symmetry condition.

  We also consider a more realistic scenario in which users have the right to choose to communicate only with the users they consider reliable, which is is the most common case in web transactions in the real world. The studies aiming at extending the Crowds protocol to more realistic scenarios include [14] and [23]. In [14] Hamadou et al. have taken into account that attackers usually gather additional information correlated to the anonymous agents before attacking the protocol. In [23] Sassone et al. extended Crowds by allowing the possibility for members to turn corrupt, and by considering a trust estimation over participants which is expressed by a known probability distribution over participants. They gave policies of how to achieve a satisfactory level of safety under this scenario. Note that this kind of trust estimation is shared by all members which implies that the level of a member's reliability is the same from all members' points of views and therefore can still be thought of as the symmetry condition.

9 Conclusion and Future Work

In this paper, we have defined the differential privacy degree of systems expressed in a probabilistic process calculus, and we have investigated how the privacy is affected under the compositions of CCS_p's constructs. We have proved that the probabilistic choice, the nondeterministic choice, the restriction and a restricted form of parallel composition are safe. Furthermore, we have proposed an algorithm for computing the channel matrix and the precise degree of differential privacy for a finite process in which the secret choices are only at the top-level. Then we have applied our modular reasoning methods to study the Crowds anonymity protocol and we have given a simple proof of anonymity without assuming the symmetry condition.

In future work, we intend to investigate the applicability of our approach to the problem of preserving privacy in geolocation-related applications. More specifically, we intend to use (a possibly extended version of) our probabilistic process calculus to express systems of concurrent agents moving in space and time, and interacting with each other in ways that depend on the adjacency relation. We believe that our compositional method will provide a way to synthesize differentially private mechanisms in a (semi-)automatic way.

References

1. Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella Béguelin. Probabilistic relational reasoning for differential privacy. In *Proc. of POPL*. ACM, 2012.
2. Mohit Bhargava and Catuscia Palamidessi. Probabilistic anonymity. In *Proc. of CONCUR*, volume 3653 of *LNCS*, pages 171–185. Springer, 2005.
3. Lokesh Bhoobalan, K. and Piyush Harsh. An Experimental Study and Analysis of Crowds based Anonymity. In *The 2011 Int. Conf. on Internet Computing*, 2011.
4. Christelle Braun, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Compositional methods for information-hiding. In *Proc. of FOSSACS*, volume 4962 of *LNCS*, pages 443–457. Springer, 2008.
5. Konstantinos Chatzikokolakis and Catuscia Palamidessi. Probable innocence revisited. *Theor. Comp. Sci.*, 367(1-2):123–138, 2006.
6. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.
7. Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. J. Wiley & Sons, Inc., 1991.
8. Yuxin Deng, Catuscia Palamidessi, and Jun Pang. Compositional reasoning for probabilistic finite-state behaviors. In *Processes, Terms and Cycles: Steps on the Road to Infinity*, volume 3838 of *LNCS*, pages 309–337. Springer, 2005.
9. Yuxin Deng, Jun Pang, and Peng Wu. Measuring anonymity with relative entropy. In *Proc. of the of the 4th Int. Worshop on Formal Aspects in Security and Trust*, volume 4691 of *LNCS*, pages 65–79. Springer, 2006.
10. Claudia D ̂ıaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *Proc. of the workshop on Privacy Enhancing Technologies (PET) 2002*, volume 2482 of *LNCS*, pages 54–68. Springer, 2002.
11. Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd Int. Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proc., Part II*, volume 4052 of *LNCS*, pages 1–12. Springer, 2006.
12. Cynthia Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–96, 2011.
13. Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proc. of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 371–380. ACM, 2009.
14. Sardaouna Hamadou, Catuscia Palamidessi, Vladimiro Sassone, and Ehab ElSalamouny. Probable innocence in the presence of independent knowledge. In *Postproceedings of the 6th Int. Workshop on Formal Aspects in Security and Trust*, volume 5983 of *LNCS*, pages 141–156. Springer, 2010.
15. Kim G. Larsen and Arne Skou. Compositional verification of probabilistic processes. In *Proc. of CONCUR*, volume 630 of *LNCS*, pages 456–471. Springer, 1992.
16. Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 19–30. ACM, 2009.
17. R. Milner. *Communication and Concurrency*. Series in Comp. Sci. Prentice Hall, 1989.
18. Robin Milner. *Communicating and mobile systems: the π-calculus*. CUP, 1999.
19. Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *Proc. of S&P*, pages 173–187. IEEE, 2009.
20. Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger: a calculus for differential privacy. In *Proceeding of the 15th ACM SIGPLAN international conference on Functional programming (ICFP)*, pages 157–168. ACM, 2010.

21. Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Trans. on Information and System Security*, 1(1):66–92, 1998.
22. Indrajit Roy, Srinath T. V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. Airavat: Security and privacy for mapreduce. In *Proc. of the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 297–312. USENIX Association, 2010.
23. Vladimiro Sassone, Ehab ElSalamouny, and Sardaouna Hamadou. Trust in crowds: Probabilistic behaviour in anonymity protocols. In *Proc. of the Fifth Int. Symposium on Trustworthly Global Computing*, volume 6084 of *LNCS*, pages 88–102. Springer, 2010.
24. Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, 1995. Tech. Rep. MIT/LCS/TR-676.
25. Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. In *Proc. of CONCUR*, volume 836 of *LNCS*, pages 481–496. Springer, 1994.
26. Hongfei Sui, Jianxin Wang, Jianer Chen, and Songqiao Chen. The cost of becoming anonymous: on the participant payload in crowds. *Information Processing Letters*, 90(2):81–86, 2004.

10 Appendix

10.1 Proof of Theorem 1

1. Let $P = \lfloor + \rfloor_i \, o_i.P_i$. Consider an arbitrary scheduler ζ of the process P, according to the transition rule ACT in the Table 1, ζ resolves the nondeterminism in the first step by arbitrarily choosing a label $o_j, j \in \{1, 2, \ldots, h\}$. Let ζ_j be the projection of ζ by removing the first state and the first nondeterministic observation o_j from the execution fragments in the domain . Observe that, the scheduler ζ_j is compatible with process term P_j, that is, ζ_j is one of P_j's schedulers. For every conditional probability $p(o_j \boldsymbol{o}|\boldsymbol{s})$ in $M_\zeta(P)$, there exists a corresponding element $p^j(\boldsymbol{o}|\boldsymbol{s})$ in $M_{\zeta_j}(P_j)$, such that
$$p(o_j \boldsymbol{o}|\boldsymbol{s}) = p^j(\boldsymbol{o}|\boldsymbol{s}). \tag{2}$$
From the differential privacy that P_j gives, we derive
$$df_\zeta[\![P]\!] = df_{\zeta_j}[\![P_j]\!] = \epsilon_j \leq \max_i\{\epsilon_i\}$$
which concludes the proof in this case.
2. Let $P = \sum_i p_i P_i$. Consider an an arbitrary scheduler ζ for P. After one subprocess is randomly chosen according to its probability, ζ must be compatible with this subprocess, resolving all the nondeterminism in it. It holds that the conditional probability $p(\boldsymbol{o}|\boldsymbol{s})$ in the resulting matrix $M_\zeta(P)$ and the corresponding one $p^i(\boldsymbol{o}|\boldsymbol{s})$ in each matrix $M_\zeta(P_i)$ satisfy the following relation.
$$p(\boldsymbol{o}|\boldsymbol{s}) = \sum_i p_i \cdot p^i(\boldsymbol{o}|\boldsymbol{s}) \tag{3}$$
It is easy to see that
$$\begin{aligned}
\frac{p(\boldsymbol{o}|\boldsymbol{s})}{p(\boldsymbol{o}|\boldsymbol{s}')} &= \frac{\sum_i p_i \cdot p^i(\boldsymbol{o}|\boldsymbol{s})}{\sum_i p_i \cdot p^i(\boldsymbol{o}|\boldsymbol{s}')} \\
&\leq \frac{\sum_i p_i \cdot e^{\epsilon_i} p^i(\boldsymbol{o}|\boldsymbol{s}')}{\sum_i p_i \cdot p^i(\boldsymbol{o}|\boldsymbol{s}')} \quad \text{(since } M_\zeta(P_i) \text{ gives } \epsilon_i\text{-d.p.)} \\
&\leq e^{\max_i\{\epsilon_i\}} \frac{\sum_i p_i \cdot p^i(\boldsymbol{o}|\boldsymbol{s}')}{\sum_i p_i \cdot p^i(\boldsymbol{o}|\boldsymbol{s}')} \\
&= e^{\max_i\{\epsilon_i\}}
\end{aligned}$$
which concludes the proof in this case.
3. Let $P = (\nu a)P_1$. Let $p(\boldsymbol{o}|\boldsymbol{s})$ denote the conditional probability in the matrix $M_\zeta(P_1)$. According to the operational semantic rule RES of restriction construct in the Table 1, the process P is not able to perform the label a. Its execution tree $etree(P)$ can be obtained from $etree(P_1)$ by removing all the transitions with the label a and the whole subtrees following those $\xrightarrow{a}$ transitions. Two cases need to be considered.

- $a \in Obs$. Consider an arbitrary scheduler ζ' of P, let $(\mathcal{S}, \mathcal{O}', M_{\zeta'}(P))$ represent the new channel produced by $etree(P, \zeta')$. Observe that there exists a scheduler ζ of the process P_1, such that, for all the conditional probability $p'(\boldsymbol{o}'|\boldsymbol{s})$ in the matrix $M_{\zeta'}(P)$, the following equation holds.

$$p'(\boldsymbol{o}'|\boldsymbol{s}) = \sum_{f(\boldsymbol{o})=\boldsymbol{o}'} p(\boldsymbol{o}|\boldsymbol{s}) \tag{4}$$

where the function $f(\boldsymbol{o})$ with $\boldsymbol{o} = o_1, o_2, \ldots, o_k$ is

$$f(\boldsymbol{o}) = \begin{cases} \boldsymbol{o} & \text{if the label } a \text{ does not appear in the sequence } \boldsymbol{o} \\ o_1, o_2, \ldots, o_i & \text{if the label } a \text{ first occurs in the sequence } \boldsymbol{o} \text{ at the position } i+1 \text{ with } i < k. \end{cases} \tag{5}$$

It's easy to get that for all secret input $\boldsymbol{s}_1, \boldsymbol{s}_2 \in \mathcal{S}$, and for all observable $\boldsymbol{o}' \in \mathcal{O}'$,

$$\frac{p'(\boldsymbol{o}'|\boldsymbol{s}_1)}{p'(\boldsymbol{o}'|\boldsymbol{s}_2)} = \frac{\sum_{f(\boldsymbol{o})=\boldsymbol{o}'} p(\boldsymbol{o}|\boldsymbol{s}_1)}{\sum_{f(\boldsymbol{o})=\boldsymbol{o}'} p(\boldsymbol{o}|\boldsymbol{s}_2)} \leq \frac{\sum_{f(\boldsymbol{o})=\boldsymbol{o}'} e^{\epsilon_1} p(\boldsymbol{o}|\boldsymbol{s}_2)}{\sum_{f(\boldsymbol{o})=\boldsymbol{o}'} p(\boldsymbol{o}|\boldsymbol{s}_2)} = e^{\epsilon_1}$$

which shows that $M_{\zeta'}(P)$ enjoys ϵ_1-differential privacy when the restricted label a is observable.

- $a \in Sec$. Consider an arbitrary scheduler ζ' of P, let $(\mathcal{S}', \mathcal{O}, M_{\zeta'}(P))$ represent the new channel produced by $etree(P, \zeta')$. Observe that there exists a scheduler ζ of the process P_1, such that, for all the conditional probability $p'(\boldsymbol{o}|\boldsymbol{s}')$ in the matrix $M_{\zeta'}(P)$, the following equation holds.

$$p'(\boldsymbol{o}|\boldsymbol{s}') = \sum_{f(\boldsymbol{s})=\boldsymbol{s}'} p(\boldsymbol{o}|\boldsymbol{s}) \cdot \frac{\pi_{\boldsymbol{s}}}{\sum_{f(\boldsymbol{s}^*)=\boldsymbol{s}'} \pi_{\boldsymbol{s}^*}} \tag{6}$$

where the function $f(\boldsymbol{s})$ with $\boldsymbol{s} = s_1, s_2, \ldots, s_k$ is

$$f(\boldsymbol{s}) = \begin{cases} \boldsymbol{s} & \text{if the label } a \text{ does not appear in the sequence } \boldsymbol{s} \\ s_1, s_2, \ldots, s_i & \text{if the label } a \text{ first occurs in the sequence } \boldsymbol{s} \text{ at the position } i+1 \text{ with } i < k. \end{cases} \tag{7}$$

Remember that $\pi_{\boldsymbol{s}}$ is the probability of the input $\boldsymbol{s}$. We use $p(\boldsymbol{o}|\min_{\boldsymbol{s}'})$ to denote the minimal probability in the set $\{\, p(\boldsymbol{o}|\boldsymbol{s}) \mid f(\boldsymbol{s}) = \boldsymbol{s}' \}$. For all secret input

$\boldsymbol{s}'_1, \boldsymbol{s}'_2 \in \mathcal{S}'$, and for all observable $\boldsymbol{o} \in \mathcal{O}$, we obtain

$$\begin{aligned}
\frac{p'(\boldsymbol{o}|\boldsymbol{s}'_1)}{p'(\boldsymbol{o}|\boldsymbol{s}'_2)} &= \frac{\sum_{f(\boldsymbol{s}_1)=\boldsymbol{s}'_1} p(\boldsymbol{o}|\boldsymbol{s}_1) \cdot \frac{\pi_{\boldsymbol{s}_1}}{\sum_{f(\boldsymbol{s}^*_1)=\boldsymbol{s}'_1} \pi_{\boldsymbol{s}^*_1}}}{\sum_{f(\boldsymbol{s}_2)=\boldsymbol{s}'_2} p(\boldsymbol{o}|\boldsymbol{s}_2) \cdot \frac{\pi_{\boldsymbol{s}_2}}{\sum_{f(\boldsymbol{s}^*_2)=\boldsymbol{s}'_2} \pi_{\boldsymbol{s}^*_2}}} \\
&\leq \frac{\sum_{f(\boldsymbol{s}_1)=\boldsymbol{s}'_1} p(\boldsymbol{o}|\boldsymbol{s}_1) \cdot \frac{\pi_{\boldsymbol{s}_1}}{\sum_{f(\boldsymbol{s}^*_1)=\boldsymbol{s}'_1} \pi_{\boldsymbol{s}^*_1}}}{\sum_{f(\boldsymbol{s}_2)=\boldsymbol{s}'_2} p(\boldsymbol{o}|\min_{\boldsymbol{s}'_2}) \cdot \frac{\pi_{\boldsymbol{s}_2}}{\sum_{f(\boldsymbol{s}^*_2)=\boldsymbol{s}'_2} \pi_{\boldsymbol{s}^*_2}}} \\
&\leq \frac{\sum_{f(\boldsymbol{s}_1)=\boldsymbol{s}'_1} e^{\epsilon_1} \cdot p(\boldsymbol{o}|\min_{\boldsymbol{s}'_2}) \cdot \frac{\pi_{\boldsymbol{s}_1}}{\sum_{f(\boldsymbol{s}^*_1)=\boldsymbol{s}'_1} \pi_{\boldsymbol{s}^*_1}}}{\sum_{f(\boldsymbol{s}_2)=\boldsymbol{s}'_2} p(\boldsymbol{o}|\min_{\boldsymbol{s}'_2}) \cdot \frac{\pi_{\boldsymbol{s}_2}}{\sum_{f(\boldsymbol{s}^*_2)=\boldsymbol{s}'_2} \pi_{\boldsymbol{s}^*_2}}} \quad \text{(since } M_\zeta(P_1) \text{ gives } \epsilon_1\text{-d.p.)} \\
&\leq \frac{e^{\epsilon_1} \cdot p(\boldsymbol{o}|\min_{\boldsymbol{s}'_2}) \cdot \sum_{f(\boldsymbol{s}_1)=\boldsymbol{s}'_1} \frac{\pi_{\boldsymbol{s}_1}}{\sum_{f(\boldsymbol{s}^*_1)=\boldsymbol{s}'_1} \pi_{\boldsymbol{s}^*_1}}}{p(\boldsymbol{o}|\min_{\boldsymbol{s}'_2}) \cdot \sum_{f(\boldsymbol{s}_2)=\boldsymbol{s}'_2} \frac{\pi_{\boldsymbol{s}_2}}{\sum_{f(\boldsymbol{s}^*_2)=\boldsymbol{s}'_2} \pi_{\boldsymbol{s}^*_2}}} \\
&= e^{\epsilon_1}
\end{aligned}$$

which shows that $M_{\zeta'}(P)$ enjoys ϵ_1-differential privacy when the restricted label a is secret and concludes the proof of the case of the restriction construct.

4. Let $P = (\nu o_1, o_2, \ldots, o_k)\,(P_1 \mid P_2)$. The proof proceeds by reducing the execution tree of P relative to an arbitrary scheduler ζ to a new one $etree'(P, \zeta)$. This new tree enjoys a level of differential privacy which is less safer than the one of the original $etree(P, \zeta)$, while it is isomorphic to the execution tree of $(\nu o_1, \ldots, o_{i-1})P_2$ relative to a certain scheduler ζ_2. We derive,

$$df\,[\![etree(P, \zeta)]\!] \leq df\,[\![etree'(P, \zeta)]\!] = df\,[\![etree((\nu o_1, \ldots, o_{i-1})P_2, \zeta_2)]\!] = \epsilon_2$$

which proves that the process P enjoys ϵ_2-differential privacy.

The reduction from $etree(P, \zeta)$ to $etree'(P, \zeta)$ is described in the following. First we give the definitions of P_1*'s positions* and P_2*'s positions*. Consider a generic state in $etree(P, \zeta)$, and let $(\nu o_1, o_2, \ldots, o_k)\,(P'_1 \mid P'_2)$ be the new term in that state. Assume α to be the execution history up to that node. From the assumption that $(\nu o_1, o_2, \ldots, o_k)P_1$ is a safe component, there are three possible kinds of transitions performable from each state according to the operational semantics.

(a-step) $(\nu o_1, o_2, \ldots, o_k)\,(P'_1 \mid P'_2) \xrightarrow{a} (\nu o_1, o_2, \ldots, o_k)\,(\mu \mid P'_2)$ due to a transition $P'_1 \xrightarrow{a} \mu$. In this case, a must be τ, because P_1 does not contain secret labels and all its observable labels are included in $\{o_1, o_2, \ldots, o_k\}$. Assume that $\mu = \sum_i p_i\ \delta(P'_{1i})$. Then we have $(\nu o_1, o_2, \ldots, o_k)\,(\mu \mid P'_2) = \sum_i p_i\ \delta((\nu o_1, o_2, \ldots, o_k)\,(P'_{1i} \mid P'_2))$.

(b-step) $(\nu o_1, o_2, \ldots, o_k)\,(P'_1 \mid P'_2) \xrightarrow{a} (\nu o_1, o_2, \ldots, o_k)\,(P'_1 \mid \mu)$ due to a transition $P'_2 \xrightarrow{a} \mu$, with a not included in $\{o_1, o_2, \ldots, o_k\}$.

(c-step) $(\nu o_1, o_2, \ldots, o_k)\,(P'_1 \mid P'_2) \xrightarrow{\tau} (\nu o_1, o_2, \ldots, o_k)\,\delta(P''_1 \mid P''_2)$ due to the transitions $P'_1 \xrightarrow{a} \delta(P''_1)$ and $P'_2 \xrightarrow{\bar{a}} \delta(P''_2)$. As assumed in the condition a must be an observable in $\{o_i, o_{i+1}, \ldots, o_k\}$.

We define P_1*'s positions* (resp. P_2*'s positions*) as the set of states in $etree(P,\zeta)$ where the transition of type (a) (resp. the transition of type (b) or (c)) is chosen by ζ. The tree $etree'(P,\zeta)$ is obtained by replacing each P_1's position with its subtree $etree(\alpha\tau(\nu o_1, o_2, \ldots, o_k)\,(P'_{1m} \mid P'_2), \zeta)$ which gives the maximal value of differential privacy among all its subtrees, that is,

$$df\,[\![etree(\alpha\tau(\nu o_1, o_2, \ldots, o_k)\,(P'_{1m} \mid P'_2), \zeta)]\!] = \max_i\{\,df\,[\![etree(\alpha\tau(\nu o_1, o_2, \ldots, o_k)\,(P'_{1i} \mid P'_2), \zeta)]\!]\}$$

By simple induction on the depth of the tree, we obtain that $df\,[\![etree(P,\zeta)]\!]$ is an increasing function of its subtrees. Then by the previous result about the probabilistic choice, we have

$$df\,[\![etree(P,\zeta)]\!] \leq df\,[\![etree'(P,\zeta)]\!]$$

Note that after the process P_1's impact on the safety is resolved, all the states left in $etree'(P,\zeta)$ are P_2's positions. It is easy to find a corresponding scheduler ζ_2 for the execution tree of $(\nu o_1, \ldots, o_{i-1})P_2$ such that

- for every b-step in $etree'(P,\zeta)$, ζ_2 chooses the same transition in $etree((\nu o_1, \ldots, o_{i-1})P_2, \zeta_2)$, i.e. $P'_2 \xrightarrow{a} \mu$ with $a \notin \{o_1, o_2, \ldots, o_k\}$,
- for every c-step in $etree'(P,\zeta)$, ζ_2 chooses the same transition in $etree((\nu o_1, \ldots, o_{i-1})P_2, \zeta_2)$, i.e. $P'_2 \xrightarrow{\bar{a}} \delta(P''_2)$ with $a \in \{o_i, o_{i+1}, \ldots, o_k\}$.

Observe now that $etree'(P,\zeta)$ is isomorphic to $etree((\nu o_1, \ldots, o_{i-1})P_2, \zeta_2)$, which concludes the proof in this case.

5. Rerun the process P_1 n times. Let $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n$ and $\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_n$ be the n times replications' secret inputs and observables, respectively. Because of the assumption that the new process starts after the current one terminates, we have

$$\begin{aligned}
& p(\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_n | \boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n) \\
=\ & p(\boldsymbol{o}_1 | \boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n) p(\boldsymbol{o}_2 | \boldsymbol{o}_1, \boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n) \\
& \cdots p(\boldsymbol{o}_n | \boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_{n-1}, \boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n) \\
=\ & p(\boldsymbol{o}_1 | \boldsymbol{s}_1) p(\boldsymbol{o}_2 | \boldsymbol{s}_2) \cdots p(\boldsymbol{o}_n | \boldsymbol{s}_n) \qquad \text{(sequential replication)} \qquad (8) \\
\leq\ & e^{\epsilon_1} p(\boldsymbol{o}_1 | \boldsymbol{s}'_1) e^{\epsilon_1} p(\boldsymbol{o}_2 | \boldsymbol{s}'_2) \cdots e^{\epsilon_1} p(\boldsymbol{o}_n | \boldsymbol{s}'_n) \quad \text{(since } M_\zeta(P_1) \text{ gives } \epsilon_1\text{-d.p.)} \\
=\ & e^{n\epsilon_1} p(\boldsymbol{o}_1 | \boldsymbol{s}'_1) p(\boldsymbol{o}_2 | \boldsymbol{s}'_2) \cdots p(\boldsymbol{o}_n | \boldsymbol{s}'_n) \\
=\ & e^{n\epsilon_1} p(\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_n | \boldsymbol{s}'_1, \boldsymbol{s}'_2, \ldots, \boldsymbol{s}'_n)
\end{aligned}$$

which concludes the proof.

10.2 Computing the channel matrix

We now show the combination of matrices of subprocesses under four operators, that are the nondeterministic choice, the probabilistic choice,the restriction and the sequential replication. Let $p_\zeta(\boldsymbol{o}|\boldsymbol{s})$ (resp. $p^i_\zeta(\boldsymbol{o}|\boldsymbol{s})$) be the probability in the matrix $M_\zeta(P)$ (resp. $M_\zeta(P_i)$) .

- Case $P = \sum_i p_i P_i$: If the scheduler $\zeta = \bigcup_i \zeta_i$ where $\zeta_i \in \Delta(P_i)$, then

$$p_\zeta(\boldsymbol{o}|\boldsymbol{s}) = \sum_i p_i \cdot p^i_{\zeta_i}(\boldsymbol{o}|\boldsymbol{s}).$$

- Case $P = \biguplus_i r_i.P_i$: If the scheduler $\zeta = \{\zeta(\varepsilon) = r_i\} \cup \{\zeta(r_i\alpha) = \zeta_i(\alpha)\}$ where $\zeta_i \in \Delta(P_i)$, then

$$p_\zeta(r_i\boldsymbol{o}|\boldsymbol{s}) = p^i_{\zeta_i}(\boldsymbol{o}|\boldsymbol{s}).$$

- Case $P' = (\nu a)P$: If the scheduler $\zeta' = \zeta|_a$ where $\zeta \in \Delta(P)$, then

$$p'_{\zeta'}(\boldsymbol{o}'|\boldsymbol{s}) = \sum_{f(\boldsymbol{o})=\boldsymbol{o}'} p_\zeta(\boldsymbol{o}|\boldsymbol{s})$$

  if $a \in \mathcal{O}$, where the function $f(\boldsymbol{o})$ is shown in the formula (5) ;

$$p'_{\zeta'}(\boldsymbol{o}|\boldsymbol{s}') = \sum_{f(\boldsymbol{s})=\boldsymbol{s}'} p_\zeta(\boldsymbol{o}|\boldsymbol{s}) \cdot \frac{\pi_{\boldsymbol{s}}}{\sum_{f(\boldsymbol{s}^*)=\boldsymbol{s}'} \pi_{\boldsymbol{s}^*}}$$

  if $a \in \mathcal{S}$, where the function $f(\boldsymbol{s})$ is shown in the formula (7) .
- Case $P' = \overset{\circlearrowleft}{}^n P$: Let $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n$ and $\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_n$ be the n times replications' secret inputs and observables, respectively. If the scheduler $\zeta' = \zeta_1; \zeta_2; \cdots; \zeta_n$ where $\zeta_i \in \Delta(P)$ for all i, then

$$p'_{\zeta'}(\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_n|\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n) = p_{\zeta_1}(\boldsymbol{o}_1|\boldsymbol{s}_1)p_{\zeta_2}(\boldsymbol{o}_2|\boldsymbol{s}_2)\cdots p_{\zeta_n}(\boldsymbol{o}_n|\boldsymbol{s}_n).$$