Modular Reasoning about Differential Privacy in a Probabilistic Process Calculus

Lili Xu

Modular Reasoning about Differential Privacy in a Probabilistic Process Calculus ⋆

Lili Xu[1,2,3]

[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
xulili@ios.ac.cn

Abstract. The verification of systems for protecting sensitive and confidential information is becoming an increasingly important issue. Differential privacy is a promising notion of privacy originated from the community of statistical databases, and now widely adopted in various models of computation. We consider a probabilistic process calculus as a specification formalism for concurrent systems, and we propose a framework for reasoning about the degree of differential privacy provided by such systems. In particular, we investigate the preservation of the degree of privacy under composition via the various operators. We illustrate our idea by proving an anonymity-preservation property for a variant of the Crowds protocol for which the standard analyses from the literature are inapplicable. Finally, we make some preliminary steps towards automatically computing the degree of privacy of a system in a compositional way.

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 [20] 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. The verification of systems for protecting sensitive and confidential information is becoming an increasingly important issue in the modern world.

Many protocols for protecting confidential information have been proposed in the literature. In order to obfuscate the link between the secret and the public information, several of them use randomized mechanisms. Typical examples are the DCNets [8], Crowds [22], Onion Routing [26], Freenet [9] and many others. Another common denominator is that various entities involved in the system to verify occur as concurrent processes and present typically a nondeterministic behavior. It is therefore natural and standard to apply process calculi, and the adaptation of the process-algebraic framework

⋆ This work has been jointly supported by the National Natural Science Foundation of China (Grant No.60833001), the project ANR-09-BLAN-0169-01 PANDA and the INRIA large scale initiative CAPPRIS.

to specify and reason about security protocols is an active line of research. See e.g. the CCS approach [18, 11, 5], as well as the Applied pi-calculus approach for security [1], anonymity [19] and privacy [6]. In this paper, we consider a probabilistic extension of CCS, which we call CCS$_p$. In addition to the standard parallel composition and nondeterministic choice of CCS, CCS$_p$ provides also a primitive for the probabilistic choice.

Several formalizations of the notion of protection have been proposed in the literature. Among those based on probability theory, there are *strong anonymity* and *conditional anonymity* [8, 15, 4] and *probable innocence* [22]. Different from the previous notions providing only true-or-false properties, the concepts from Information Theory [10] based on *entropy*, notably *mutual information* and *capacity*, express the degree of protection in a quantitative way.

Differential privacy [12, 14, 13] is a promising definition of confidentiality that has emerged recently from the field of statistical databases. It provides strong privacy guarantees, and requires fewer assumptions than the information-theoretical approach. We say that a system is ϵ-differentially private if for every pair of *adjacent* datasets (i.e. datasets which differ in the data of an individual only), the probabilities of obtaining a certain answer differ at most by a factor e^{ϵ}. Differential privacy captures the intuitive requirement that the (public) answer to a query should not be affected too much by the (private) data of each singular individual. In this paper we consider a version of differential privacy using a generic notion of adjacency, which provides the basis for formalizing also other security concepts, like anonymity.

The main contribution of this work is to investigate differential privacy for concurrent systems in the context of a probabilistic process calculus (CCS$_p$). We present a modular approach for reasoning about differential privacy, where the modularity is with respect to the constructs of CCS$_p$. More specifically, we show that the restriction operator, the probabilistic choice, the nondeterministic choice and a restricted form of parallel composition are safe under composition, in the sense that they do not decrease the privacy of a system. Compositionality plays an important role in the construction and analysis of security systems: Rather than analyzing a complex system as a whole, the safe constructs allow us to split the system in parts, analyze the degree of privacy of each part separately, and combine the results to obtain the global degree of privacy.

We illustrate our compositionality results by proving an anonymity-preservation property for an extension of the *Crowds* protocol [22]. Crowds is an anonymity protocol which allows Internet users to perform web transactions without revealing their private identity. This is achieved by using a chain of forwarders, chosen randomly, rather than sending the message directly to the final recipient. In the standard Crowds all members have the same probability of being used as forwarders, which gives the protocol a symmetric structure (cf. equations (13) and (14) in [7]). In practice, however, the protocol can be asymmetric, because a sender may trust some agents (as forwarders) more than others, or may prefer those which are geographically closer, or more stable, in order to achieve a better performance. In this paper, our extension of Crowds consists in allowing each member to have a set of *trusted* users to which they can forward the message. This breaks the symmetry properties of the original protocol, thus making the standard analyses of [22] inapplicable. In contrast, our compositional method gives a very simple proof.

Finally, we make some preliminary steps towards automatically computing the degree of privacy of a CCS$_p$ process in a compositional way. In this paper, we only consider the case of finite processes in which the secret choices are all at the top level. We leave the general case for future work.

Nowadays differential privacy is widely adopted in many models of computation, for instance, it has been investigated in the context of a SQL-like language [17], a linear type system [21], a relational Hoare logic [3], in MapReduce for cloud computing [23], and in probabilistic labeled transition systems for interactive systems [27]. To the best of our knowledge, this paper is the first to investigate differential privacy within the setting of a process algebra.

Summary of Contributions Our contributions are three-fold. We present a modular approach for reasoning about differential privacy for protocols expressed in a probabilistic process algebra (CCS$_p$). We apply our compositionality method to prove an anonymity-preservation property for an extended version of Crowds, i.e. with member-wise trusted forwarders. We show an algorithm for computing the degree of privacy of a finite process.

Plan of the paper In Sections 2 - 4 we review the preliminary notions of CCS$_p$ and differential privacy. In Section 5 we investigate the compositionality of differential privacy with respect to CCS$_p$ constructs. In Section 6, we apply our compositionality result to Crowds with trust. In Section 7, we present the algorithm for computing the degree of privacy. Finally, Section 8 and 9 discuss further related work and conclude. Detailed proofs of the theorem in Section 5 are in the Appendix.

Acknowledgments The author gratefully acknowledges the contributions of Catuscia Palamidessi and Kostas Chatzikokolakis. Discussions with them helped the author to substantially improve the paper.

2 Preliminaries

2.1 Probabilistic automata

We recall the formalism of *probabilistic automata* [25], to be used as the operational semantics for the probabilistic CCS.

We denote the set of all discrete probability distributions over a set X by $Disc(X)$. For $x \in X$, we denote by $\delta(x)$ (the *Dirac measure* on x) the probability distribution that assigns probability 1 to $\{x\}$.

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 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 of alternating states and labels $P_{init}a_0P_1a_1P_2a_2P_3\cdots$, such that for each i, there is a

transition $(P_i, a_i, \mu_i) \in \mathcal{T}$ with $\mu_i(P_{i+1}) > 0$. We will use $exec^*(\mathcal{M})$ to represent the set of all the finite executions of $\mathcal{M}$, $exec(\mathcal{M})$ to represent the set of all the executions of $\mathcal{M}$, and $lst(\alpha)$ to denote the last state of a finite execution $\alpha \in exec^*(\mathcal{M})$.

The *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 $P = lst(\alpha)$. (We omit $\mathcal{M}$ when it is clear from the context.) The idea is that a scheduler resolves the nondeterminism by selecting a transition among the ones available in $\mathcal{T}$, based on the history of the execution.

The *execution tree* of $\mathcal{M}$ *relative to a scheduler* ζ, denoted by $etree(\mathcal{M}, \zeta)$, is a fully probabilistic automaton $\mathcal{M}'' = (\mathcal{P}'', P''_{init}, Act, \mathcal{T}'')$, 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 [18], that allows for both nondeterministic and probabilistic choice. Following [5] we make a distinction between *observable* and *secret* labels, for applications to security systems and protocols.

We consider a countable set Act of labels a, partitioned into a set Sec of *secrets* s, a set Obs of *observables* o, and the silent action τ. For each $o \in Obs$, we assume a complementary label $\overline{o} \in Obs$ with the convention that $\overline{\overline{o}} = o$. The syntax of CCS$_p$ is:

$P ::=$		*process term*
	$\bigoplus_i p_i P_i$	*probabilistic choice*
	$\boxplus_i s_i.P_i$	*secret choice* $(s_i \in Sec)$
	$\boxplus_i r_i.P_i$	*nondeterministic choice* $(r_i \in Obs \cup \{\tau\})$
	$P \mid P$	*parallel composition*
	$(\nu a)P$	*restriction*
	$!\,P$	*replication*

The term $\bigoplus_i p_i P_i$, in which the p_is are positive rationals that sum up to one, represents a *blind probabilistic choice*, in the sense that the choice of the branch is decided randomly (according to the corresponding probabilities) 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 $a_1.P_1 \boxplus a_2.P_2$ 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 whose transition relation is defined

$$\text{PROB}\ \frac{}{\bigoplus_i p_i\, P_i \xrightarrow{\tau} \sum_i p_i\, \delta(P_i)} \qquad \text{ACT}\ \frac{j \in I}{\biguplus_I a_i.P_i \xrightarrow{a_j} \delta(P_j)}$$

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

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

Table 1: The semantics of CCS$_p$.

according to the rules in the Table 1, where we use $P \xrightarrow{a} \mu$ to represent the transition (P, a, μ). We denote by $\mu \mid P$ the measure μ' such that $\mu'(P' \mid P) = \mu(P')$ for all processes $P' \in \mathcal{P}$ and $\mu'(P'') = 0$ if P'' is not of the form $P' \mid P$. Similarly $(\nu a)\mu = \mu'$ such that $\mu'((\nu a)P) = \mu(P)$. A transition of the form $P \xrightarrow{a} \delta(P')$, 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 [5] 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 $\zeta : Sec^* \to exec^*(\mathcal{M}) \rightharpoonup \mathcal{T}$ such that if:*

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

then $\zeta(\boldsymbol{s})(\alpha)$ is defined as 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 the 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. 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:*

(i) $\mathcal{P}' \subseteq exec(\mathcal{M})$,
(ii) $(\alpha, a, \mu') \in \mathcal{T}'$ *iff* $\zeta_{\boldsymbol{s}}(\alpha) = (lst(\alpha), a, \mu)$ *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 [10]. 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 usually enough to model the typical security systems, where each session is supposed to terminate in finite time.

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 $\boldsymbol{s}$, 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 $\mathcal{O}$. (See some examples of channel matrix in Section 7.2.)

Definition 3 ([5]). *Given a term* P *and a scheduler* $\zeta : \mathcal{S} \to exec^* \rightharpoonup \mathcal{T}$*, 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 [12] captures the idea that a query on a dataset does not provide 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, [12]). *A mechanism* κ *provides* ϵ-differential privacy *iff for all datasets* D_1 *and* D_2 *differing in 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. The confidential data can be complex information like sequences of keys or tuples of individual data (see Example 2). Consider complex secrets composed of n elements , for instance

$(\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n)$, and denote the adjacency relation between them by $\approx$. Without loss of generality, we assume that,

If $(\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n) \approx (\boldsymbol{s}'_1, \boldsymbol{s}'_2, \ldots, \boldsymbol{s}'_n)$, then $\boldsymbol{s}_1 \sim \boldsymbol{s}'_1, \boldsymbol{s}_2 \sim \boldsymbol{s}'_2, \ldots, \boldsymbol{s}_n \sim \boldsymbol{s}'_n$.

Example 1 (Anonymity). In the case of anonymity, the confidential data $\mathcal{S}$ are the agents' identities. Since 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 schedulers 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 $dp_\zeta[\![P]\!]$ to denote the smallest ϵ such that the process term P, under the scheduler ζ, provides ϵ-differential privacy. Furthermore we define

$$dp[\![P]\!] = \sup_\zeta dp_\zeta[\![P]\!]$$

Note that if there are both zero and non-zero probabilities occurring in the same column of the channel matrix, when the respective secrets are connected by $\sim$, 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 [8] as the property that the observation $\boldsymbol{o}$ does not change the probabilistic knowledge of the culprit's identity $\boldsymbol{s}$, i.e. $p(\boldsymbol{s}|\boldsymbol{o}) = p(\boldsymbol{s})$. This notion was extended in [4] to probabilistic and nondeterministic systems, essentially by requiring that the equation holds under any scheduler. Next proposition is an immediate consequence of the characterization given in [4].

Proposition 1. *An anonymity system P is strongly anonymous iff $dp[\![P]\!] = 0$.*

Relation between differential privacy and probable innocence Probable innocence was defined in [22] as the property that, to the eyes of an observer, each user is more likely to be innocent rather than culpable (of having initiated the message). In [7] it was shown that this is equivalent to requiring $(m-1)p(o|x_i) \geq p(o|x_j)$ for all o, i and j, where m is the number of anonymous users, and $p(o|x_i)$ denotes the probability of detecting o given that the initiator is i. It is straightforward to see the following:

Proposition 2. *An anonymity system P provides probable innocence iff $dp[\![P]\!] \leq \ln(m-1)$.*

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 notions of *safe component* and *sequential replication*. The latter can be used to represent a sequence of sessions re-executing the same protocol. .

Definition 6. *Consider a process P, and observables $o_1, o_2, \ldots, o_k$, we say that $(\nu o_1, o_2, \ldots, o_k)P$ is* a safe component *if*

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

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*

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

where

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

We now show that the nondeterministic choice, the probabilistic choice, the restriction operator and a restricted form of parallel composition are *safe*, in the sense that combining components in a safe construct does not compromise the privacy of the system, 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, $dp(\llbracket P_i \rrbracket) = \epsilon_i$. Then:*

(1) $dp(\llbracket \boxplus_i o_i.P_i \rrbracket) \leq \max_i\{\epsilon_i\}$;

(2) $dp(\llbracket \bigoplus_i p_i P_i \rrbracket) \leq \max_i\{\epsilon_i\}$;

(3) $dp(\llbracket (\nu o) P_i \rrbracket) \leq \epsilon_i$;

(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 $dp(\llbracket (\nu o_1, \ldots, o_{h-1})P_j) \rrbracket) = \epsilon_j$. Then $dp(\llbracket (\nu o_1, o_2, \ldots, o_k)\,(P_i \mid P_j) \rrbracket) \leq \epsilon_j$.

(5) $dp(\llbracket \circlearrowleft^n P_i \rrbracket) \leq n\,\epsilon_i$.

Properties (1) and (2) point out that the degree of privacy of a system, consisting of some subsystems in a nondeterministic or probabilistic choice, is determined by the subsystem with the lowest degree of privacy. Properties (3) and (4) intuitively say that, turning an observable label to be unobservable, and paralleling with a safe component, maintain the level of privacy. Property (5) means that the degree of privacy of a process degrades through multiple runs, since more information may be exposed.

Unfortunately the secret choice and the unrestricted form of parallel composition do not preserve the privacy, essentially due to the presence of nondeterminism. This is illustrated by the following counterexamples taken from [5]. (In Examples 3 - 5, we use the original definition of the adjacency relation, that is, the difference in only one label.)

Example 3 (For the secret choice). Let $Sec = \{s_1, s_2\}$ and assume that $\mathcal{S}$ does not contain the empty sequence. Consider the process $P = o_1.\mathbf{0} \boxplus o_2.\mathbf{0}$. Clearly, P provides 0-differential privacy, because for every sequence $\boldsymbol{s} \in \mathcal{S}$ we have $p(o_1|\boldsymbol{s}) = p(o_2|\boldsymbol{s})$. 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 $p(o_1|s_1\boldsymbol{s}) = p(o_2|s_2\boldsymbol{s}) = 1$ while $p(o_1|s_2\boldsymbol{s}) = p(o_2|s_1\boldsymbol{s}) = 0$.

Example 4 (For the need of condition (i) in Def. 6). Let Sec and $\mathcal{S}$ be as in Example 3. Define $P_1 = s_1.\mathbf{0} \boxplus s_2.\mathbf{0}$ and $P_2 = o_1.\mathbf{0} \boxplus o_2.\mathbf{0}$. Clearly, P_2 provides 0-differential privacy. Consider now the parallel term $P_1 \mid P_2$ and define a scheduler that first executes a secret label s in P_1 and then, if s is s_1, it selects o_1, while if s is s_2, it selects o_2. The rest proceeds like in Example 3.

Example 5 (For the need of condition (ii) in Def. 6). Let Sec and $\mathcal{S}$ be as in Example 3. Define $P_1 = o.\mathbf{0}$ and $P_2 = s_1.(o_1.\mathbf{0} \oplus_{.5} o_2.\mathbf{0}) \boxplus s_2.(o_1.\mathbf{0} \oplus_{.5} o_2.\mathbf{0})$. It is easy to see that P_2 provides 0-differential privacy. Consider the term $P_1 \mid P_2$ and define a scheduler that first executes a secret label s in P_2 and then, if s is s_1, it selects first P_1 and then the continuation of P_2, while if s is s_2, it selects first the continuation of P_2 and then P_1. Hence, under this scheduler, for every sequence $\boldsymbol{s} \in \mathcal{S}$, $p(oo_1|s_1\boldsymbol{s}) = p(oo_2|s_1\boldsymbol{s}) = 0.5$ and also $p(o_1o|s_2\boldsymbol{s}) = p(o_2o|s_2\boldsymbol{s}) = 0.5$ while $p(oo_1|s_2\boldsymbol{s}) = p(oo_2|s_2\boldsymbol{s}) = 0$ and $p(o_1o|s_1\boldsymbol{s}) = p(o_2o|s_1\boldsymbol{s}) = 0$. Therefore s_1 and s_2 are disclosed.

Intuitively, the existence of free observables (i.e. o of P_1 in this example) may create different interleavings, which can be used by the scheduler to mark different secrets.

6 A case study: the Crowds protocol

In this section, we apply the compositionality result (4) in Theorem 1 to prove the anonymity-preservation property of an extended Crowds protocol with member-wise trusted forwarders. The novelty of our proof is that it is simple. Moreover, it does not require the symmetry property that is usually made in the literature in order to simplify the analysis.

Crowds is an anonymity protocol which allows Internet users to send messages without revealing their identity. More specifically, a crowd is a group of n participants constituted by m *honest members* and $c\ (= n - m)$ *corrupted members* (the *attackers*). The destination of messages is named the *server*. The protocol works as follows:

- When a member, called the *initiator*, 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, either
  - with probability $1 - p_f$, delivers the message to the server, or
  - with 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.

In this way, even if the message is caught by an attacker, the attacker cannot be sure whether the previous forwarder is the initiator 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.

6.1 The Crowds with member-wise trusted forwarders

The above standard Crowds protocol implicitly requires the *symmetry conditions* (see [7]), in the sense that a new forwarder is chosen among all the members in the crowd with uniform probability. In this context, it has been proved in [22] that Crowds can satisfy probable innocence under certain conditions.

In this paper, we consider a *member-wise trusted forwarders* scenario, in which the member currently holding the message selects a forwarder only among the members which she thinks are trustable. We also consider the case of *reputed initiators*. The idea is that a new member has the right to initiate requests only when she has acquired a certain level of *reputation*. This is motivated by some kinds of social networks in which, when a new agent wants to join a web conversation, her behavior needs to be examined for a while until she becomes a totally legal member.

The standard Crowds 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 Crowds. We use $x_1, x_2, \ldots, x_n$ to range over $\mathcal{C}$, and $\bigoplus^{\mathcal{U}}$ to represent an uniform distribution. For simplicity we assume that once an attacker receives a message from an honest member, it will terminate after reporting the detection through the observable $\overline{detect}$. The reason is that by forwarding the message after the first detection, attackers can not gain more useful information. Hence at most one honest member can be detected. Precisely, the set of secret labels is $\{\, act_i \mid x_i \in \mathcal{H}_r \}$ and the set of observable labels is $\{\, \overline{detect}\langle i \rangle \mid x_i \in \mathcal{H} \,\} \cup \{\, \overline{S} \,\}$. We denote by $\mathcal{T}_i$ the subset of Crowds members which the ith honest member trusts. Then the process terms for Crowds with member-wise trusted forwarders are similar to the terms showed in Table 2, except that the process $\mathit{Initiator}$ and the process $honest_i$ are modified as shown in Table 3.

An anonymity-preservation property Consider the scenario in which there exists a Crowds with n members (shown in Table 2). Assume that a new honest agent is allowed to join the crowd but she does not enjoy the reputation to be an initiator right away. Old members in Crowds can decide by themselves to trust the agent or not, which means that there is no restriction of how the new agent is added, thus resulting in a Crowds with member-wise trusted forwarders. Applying the compositionality theory in Section 5, we show that the privacy level of Crowds with $n + 1$ members can be estimated by the value of privacy of a simplified Crowds, obtained by considering the non-reputed

$$
\begin{aligned}
Initiator &= \bigoplus^{\mathcal{U}}_{x_i \in \mathcal{H}_r} p_i.act_i.(\bigoplus^{\mathcal{U}}_{x_j \in \mathcal{C}} p_j.\bar{x}_j\langle i\rangle) \\
honest_i &= x_i.((\bigoplus^{\mathcal{U}}_{x_j \in \mathcal{C}} p_j.\bar{x}_j\langle i\rangle.\overline{done}) \oplus_{p_f} \overline{ser}.\overline{done}) \\
Honest_i &= \circlearrowleft honest_i \\
Attacker_i &= x_i(j).\overline{detect}\langle j\rangle \\
Server &= ser.\overline{S} \\
crowd_n &= Server \,|\, Initiator | \prod_{x_i \in \mathcal{H}} Honest_i | \prod_{x_j \in \mathcal{A}} Attacker_j \\
Crowd_n &= (\nu ser)(\nu x_1, x_2, \ldots, x_n) crowd_n
\end{aligned}
$$

Table 2: The CCS$_p$ code for standard Crowds (i.e. with the symmetry conditions).

$$
\begin{aligned}
Initiator &= \bigoplus^{\mathcal{U}}_{x_i \in \mathcal{H}_r} p_i.act_i.(\bigoplus^{\mathcal{U}}_{x_j \in \mathcal{T}_i} p_j.\bar{x}_j\langle i\rangle) \\
honest_i &= x_i.((\bigoplus^{\mathcal{U}}_{x_j \in \mathcal{T}_i} p_j.\bar{x}_j\langle i\rangle.\overline{done}) \oplus_{p_f} \overline{ser}.\overline{done})
\end{aligned}
$$

Table 3: The new definitions for Crowds with member-wise trusted forwarders.

agent as an attacker and therefore ignoring her following trust links to successors. The fact is supported by the following theorem.

Theorem 2. $dp[\![Crowd_{n+1}]\!] \leq dp[\![Crowd_n]\!]$.

Proof. The addition of a new honest agent to a Crowds with n participants is presented in Table 4. Basically, it is a parallel composition of the process term $crowd_n$ of old crowd, and the process term $Honest_{n+1}$ of the new agent. In $crowd_n$, 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 a forwarder.

Consider the term $Crowd_{n+1}$. Remove the term $Honest_{n+1}$ and the corresponding restriction operators. Note that the free labels through which old Crowds communicates with the new agent are $\{ser\} \cup \{x_j | x_j \in \mathcal{T}_{n+1}\} \cup \{\bar{x}_{n+1}\langle i\rangle | x_i \in \mathcal{S}_{n+1}\}$, where $\mathcal{S}_{n+1}$ is the subset of Crowds members who trust the new agent. The label $\bar{x}_{n+1}\langle i\rangle$ behaves like an attacker, because it can reveal the identity of member (e.g. x_i) who is sending the request. Since this new agent is known not to be an initiator (because she is not reputed). 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,

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

is a safe component. By Theorem 1(4), $Crowd_{n+1}$ provides at least as much privacy as $Crowd_n$.

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

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

Table 4: Specification of the addition of the $n+1$th agent.

7 Computing the degree of privacy

In this section, we study the possibility of computing the degree of privacy in CCS$_p$ in a fine-grained way. We make a first step by considering finite processes in which the secret choices are only at the top-level. We produce the channel matrix in a bottom-up fashion, from which we obtain its degree of privacy, and we give some toy examples to illustrate the approach.

7.1 The algorithm

Consider a process term $T = \lfloor\!+\!\rfloor_i\, s_i.T_i$ starting with a secret choice. For simplicity, in this paper, we assume that there is no secret choice inside any of T_i, and no occurrence of ! , (while considering ! is a real difficult problem, the restriction about the secret choice can be lifted by using more complicated definitions of how to combine the schedulers of the components. We however leave them for future work). We also assume that every occurrence of the parallel operator is removed by using the *expansion law* [18]. We construct the set of *all possible schedulers* for a process P, denoted by $\Delta(P)$. We denote its size by $|\Delta(P)|$. Thus the residual constructs possibly occurring in T_i are the following cases:

- Case $P = \bigoplus_i p_i P_i$: Intuitively, P's scheduler selects for each P_i a scheduler from $\Delta(P_i)$. Hence, $|\Delta(P)| = O(\prod_i |\Delta(P_i)|)$.
- Case $P = \lfloor\!+\!\rfloor_i\, r_i.P_i$: P's scheduler first resolves the top nondeterministic choice and then proceeds the continuation of the process corresponding to the selected label. Thus, $|\Delta(P)| = O(\sum_i |\Delta(P_i)|)$.
- Case $P' = (\nu o)P$: Intuitively, a scheduler of P' is obtained from a scheduler in $\Delta(P)$ by removing the assignments of the executions containing o. Hence, $|\Delta(P')| = O(|\Delta(P)|)$.
- Case $P' = \circlearrowleft^n P$: Intuitively, a scheduler of P' selects for each run of P a scheduler from $\Delta(P)$, and use them in a sequential way. Hence, $|\Delta(P')| = O(|\Delta(P)|^n)$.

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 construction of the matrix of T. For every secret label s_i, T's scheduler ζ chooses a scheduler ζ_i from the set $\Delta(T_i)$. 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 the definition of differential privacy we get:

$$dp_\zeta[\![T]\!] = \min\{\epsilon \mid \boldsymbol{s} \sim \boldsymbol{s}' \Rightarrow \forall \boldsymbol{o} \in \mathcal{O}.\ p_\zeta(\boldsymbol{o}|\boldsymbol{s}) \leq e^\epsilon p_\zeta(\boldsymbol{o}|\boldsymbol{s}')\}$$

and

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

Complexity Analysis The time complexity of the above algorithm is determined by the size of the set of all 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 $s_i \sim s_j$), 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 $\boldsymbol{o}$.

7.2 Some toy examples

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

Example 6. Let $Sec = \{s_1, s_2\}$, $Obs = \{o_1, o_2\}$, 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 $M(P_1) = \begin{array}{|c|c|}\hline o_1 & o_2 \\ \hline 0.3 & 0.7 \\ \hline \end{array}$.

For the term P_2, we have $\Delta(P_2) = \{\varnothing\}$ and $M(P_2) = \begin{array}{|c|c|}\hline o_1 & o_2 \\ \hline 0.5 & 0.5 \\ \hline \end{array}$.

For the term P_3, we have $\Delta(P_3) = \{\varnothing\}$ and $M(P_3) = \begin{array}{|c|c|}\hline o_1 & o_2 \\ \hline 0.8 & 0.2 \\ \hline \end{array}$.

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 $M_{\zeta'}(P') = \begin{array}{|c|c|c|}\hline & o_1 & o_2 \\ \hline s_1 & 0.3 & 0.7 \\ \hline s_2 & 0.8 & 0.2 \\ \hline \end{array}$, which gives $(\ln 3.5)$-differential privacy.

Example 7. Let $Sec = \{s_1, s_2\}$, $Obs = \{o, o_1, o_2\}$, 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 example, we can find a scheduler producing a matrix breaking the differential privacy, for example $\begin{array}{|c|c|c|c|c|}\hline & o_1o & o_2o & oo_1 & oo_2 \\ \hline s_1 & 0.3 & 0.7 & 0 & 0 \\ \hline s_2 & 0 & 0 & 0.3 & 0.7 \\ \hline \end{array}$.

Example 8 (A discussion about the recursion operator). Furthermore, for a process including the replication operator while not producing infinite observables, we show a computation of the degree of privacy in this case. To facilitate the analysis, here we use a *fixpoint construct* [18], i.e. $\mu X.P$ where the *agent variable* X may occur in P. Note that $!\,P$ can be obtained by the definition $\mu X.(P \mid X)$.

Let Sec and Obs be as in Example 6. Consider the processes $P = \mu X.((0.4o_1 \oplus 0.2o_2 \oplus 0.4X) \boxplus (o_1 \oplus_{0.3} o_2) \boxplus (o_1 \oplus_{0.5} o_2))$ and $P' = s_1.P \boxplus s_2.P$. We have $\Delta(P') = \Delta(P) = \{\zeta_1, \zeta_2, \zeta_3\}$, where the respective matrices are $M_{\zeta_1}(P) =$

o_1	o_2	X
0.4	0.2	0.4

, $M_{\zeta_2}(P) =$

o_1	o_2
0.3	0.7

and $M_{\zeta_3}(P) =$

o_1	o_2
0.5	0.5

. It can be proved that using the scheduler ζ_1 finitely many times generates the highest (resp. lowest) probability of observing o_1 (resp. o_2), which is 0.67 (resp. 0.33). Using the scheduler ζ_2 generates the lowest (resp. highest) probability of observing o_1 (resp. o_2), which is 0.3 (resp. 0.7). Thus P' enjoys $(\ln 2.22)$-differential privacy, determined by the matrix

	o_1	o_2
s_1	0.67	0.33
s_2	0.3	0.7

.

8 Related work

- *Compositionality properties of probabilistic process calculi for security protocols.* In [11] Deng et al. used the notion of relative entropy to measure privacy. In [5] 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 compositionality results in our paper are closely related to those of [5], although we use a different measure of protection (differential privacy).
- *Compositionality of Differential Privacy.* As stated in the introduction, there is a vast body of work on formal methods for differential privacy. Compositional methods, as one of the most important features, have been intensively investigated in the field of statistical databases [17] and programs [3, 21]. These works investigate the so-called sequential and parallel compositions of queries (programs), which, in their context, mean a sequence of queries (programs) applied to the same dataset and to disjoint parts of the dataset, respectively. Under this setting, they have proved that the sequential composition decreases the privacy, and the parallel composition maintains the privacy. Our result about the sequential replication and the parallel composition in CCS$_p$ are reminiscent of the above results. But the context is different. In particular, the parallel composition concerned in this paper is different from the above one, in that the parallel operator here represents interactions of concurrent agents. Our restrictions on the parallel composition are incomparable with those of [17, 3, 21] (disjointness of databases).
- *Other extensions on the Crowds protocol.* In [16] Hamadou et al. have taken into account that attackers usually gather additional information correlated to the anonymous agents before attacking the protocol. In [24] 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. Note that, this kind of trust estimation is shared among all members, which implies that the level of a member's reliability is the same from all members' points of view and therefore can still be thought of as the symmetry condition. We consider a more realistic scenario in which users have the right to choose to communicate only with the users they consider reliable, which is the most common case in web transactions in the real world.

9 Conclusion and Future Work

In this paper, we have defined the degree of differential privacy for concurrent systems expressed in a probabilistic process calculus, and investigated how the privacy is affected under composition of the CCS_p constructs. We have applied our approach to give a simple proof for the anonymity-preservation of an extended Crowds protocol with member-wise trusted forwarders. Finally, we have presented an algorithm for computing the degree of differential privacy for a finite process.

Fine-grained methods for computing the channel matrix have been studied in [2]. In future work, we plan to optimize our current algorithm, extend it to more general processes, more precisely, develop an approach that can deduce the global property of differential privacy from local information, w.r.t. the adjacency relation. Another interesting problem is 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. Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. and Comp.*, 148(1):1–70, 1999.
2. Miguel E. Andrés, Catuscia Palamidessi, Peter van Rossum, and Geoffrey Smith. Computing the leakage of information-hiding systems. In *Proc. of TACAS*, volume 6015 of *LNCS*, pages 373–389. Springer, 2010.
3. Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Z. Béguelin. Probabilistic relational reasoning for differential privacy. In *Proc. of POPL*. ACM, 2012.
4. Mohit Bhargava and Catuscia Palamidessi. Probabilistic anonymity. In *Proc. of CONCUR*, volume 3653 of *LNCS*, pages 171–185. Springer, 2005.
5. 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.
6. Mayla Brusò, Konstantinos Chatzikokolakis, and Jerry den Hartog. Formal verification of privacy for RFID systems. In *CSF*, pages 75–88, 2010.
7. Konstantinos Chatzikokolakis and Catuscia Palamidessi. Probable innocence revisited. *Theor. Comp. Sci.*, 367(1-2):123–138, 2006.

8. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.
9. Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proc. of DIAU*, volume 2009 of *LNCS*, pages 44–66. Springer, 2000.
10. Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. J. Wiley & Sons, Inc., 1991.
11. Yuxin Deng, Jun Pang, and Peng Wu. Measuring anonymity with relative entropy. In *Proc. of the 4th Int. Worshop on Formal Aspects in Security and Trust*, volume 4691 of *LNCS*, pages 65–79. Springer, 2006.
12. Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Proceedings, Part II*, volume 4052 of *LNCS*, pages 1–12. Springer, 2006.
13. Cynthia Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–96, 2011.
14. 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.
15. Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *J. of Comp. Security*, 13(3):483–512, 2005.
16. 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.
17. 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.
18. Robin Milner. *Communication and Concurrency*. Series in Comp. Sci. Prentice Hall, 1989.
19. Aybek Mukhamedov and Mark D. Ryan. Identity escrow protocol and anonymity analysis in the applied pi-calculus. *ACM Trans. Inf. Syst. Secur.*, 13(4):41, 2010.
20. Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *Proc. of S&P*, pages 173–187. IEEE, 2009.
21. 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.
22. Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Trans. on Information and System Security*, 1(1):66–92, 1998.
23. 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.
24. 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.
25. Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, 1995. Tech. Rep. MIT/LCS/TR-676.
26. Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. In *Proc. of S&P*, pages 44–54, 1997.
27. Michael C. Tschantz, Dilsun Kaynar, and Anupam Datta. Formal verification of differential privacy for interactive systems (extended abstract). *Electron. Notes Theor. Comput. Sci.*, 276:61–79, September 2011.

10 Appendix

10.1 Proof of Theorem 1

1. Let $P = \underline{+}_i\, o_i.P_i$. Consider an arbitrary scheduler ζ of P, according to the transition rule ACT in Table 1, ζ resolves the top nondeterministic choice by arbitrarily choosing a label $o_j, j \in \{1, 2, \ldots, h\}$. Let ζ_j be the projection of ζ obtained by removing the first state and the following observation o_j from the execution fragments. 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

$$p(o_j \boldsymbol{o}|\boldsymbol{s}) = p^j(\boldsymbol{o}|\boldsymbol{s}).$$

   From the differential privacy that P_j gives, we derive

$$dp_\zeta[\![P]\!] = dp_{\zeta_j}[\![P_j]\!] \leq \epsilon_j \leq \max_i\{\epsilon_i\}$$

   which concludes the proof in this case.

2. Let $P = \bigoplus_i p_i P_i$. Consider 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.

$$p(\boldsymbol{o}|\boldsymbol{s}) = \sum_i p_i \cdot p^i(\boldsymbol{o}|\boldsymbol{s})$$

   It is easy to see that for all secret inputs $\boldsymbol{s}$, $\boldsymbol{s}'$ such that $\boldsymbol{s} \sim \boldsymbol{s}'$,

$$\begin{aligned}
\frac{p(\boldsymbol{o}|\boldsymbol{s})}{p(\boldsymbol{o}|\boldsymbol{s}')} &= \frac{\sum_i p_i \cdot p^i(\boldsymbol{o}|\boldsymbol{s})}{\sum_i p_i \cdot p^i(\boldsymbol{o}|\boldsymbol{s}')} \\
&\leq \frac{\sum_i p_i \cdot e^{\epsilon_i} p^i(\boldsymbol{o}|\boldsymbol{s}')}{\sum_i p_i \cdot p^i(\boldsymbol{o}|\boldsymbol{s}')} \quad \text{(since } M_\zeta(P_i) \text{ gives } \epsilon_i\text{-d.p.)} \\
&\leq e^{\max_i\{\epsilon_i\}} \frac{\sum_i p_i \cdot p^i(\boldsymbol{o}|\boldsymbol{s}')}{\sum_i p_i \cdot p^i(\boldsymbol{o}|\boldsymbol{s}')} \\
&= e^{\max_i\{\epsilon_i\}}
\end{aligned}$$

   which concludes the proof in this case.

3. Let $P = (\nu o)P_1$. According to the rule RES of restriction construct in Table 1, the process P is not able to perform the label o. Its execution tree $etree(P)$ can be obtained from $etree(P_1)$ by removing all transitions with the label o and the whole subtrees following those $\xrightarrow{o}$ transitions. Let $p(\boldsymbol{o}|\boldsymbol{s})$ denote the conditional probability in the matrix $M_\zeta(P_1)$. Let $\mathcal{O}'$ denote the set of observables of P. Consider an

arbitrary scheduler ζ' of P. Observe that there exists a scheduler ζ of the process P_1, such that, for every conditional probability $p'(\boldsymbol{o}'|\boldsymbol{s})$ in $M_{\zeta'}(P)$, the following equation holds.

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

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

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

It's easy to get that 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 $\boldsymbol{o}' \in \mathcal{O}'$,

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

which shows that $M_{\zeta'}(P)$ enjoys ϵ_1-differential privacy.

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 safe than the one of the original $etree(P, \zeta)$, while it is isomorphic to the execution tree of $(\nu o_1, \ldots, o_{h-1})P_2$ relative to a certain scheduler ζ_2. We derive,

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

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

The reduction from $etree(P, \zeta)$ to $etree'(P, \zeta)$ is described as follows. First we give the definitions of *P_1's positions* and *P_2's positions*. Consider an arbitrary state α in $etree(P, \zeta)$, and let $(\nu o_1, o_2, \ldots, o_k)\,(P_1' \mid P_2')$ be the generic process term of $lst(\alpha)$. 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 the 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 $\{o_h, o_{h+1}, \ldots, o_k\}$.

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,

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

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

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

Note that after the process P_1's impact on the safety is resolved, all 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 $(\nu o_1, \ldots, o_{h-1})P_2$ such that

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

Observe now that $etree'(P,\zeta)$ is isomorphic to $etree((\nu o_1, \ldots, o_{h-1})P_2, \zeta_2)$, 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, for all inputs sequences $(\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n)$ and $(\boldsymbol{s}'_1, \boldsymbol{s}'_2, \ldots, \boldsymbol{s}'_n)$ such that $(\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n) \approx (\boldsymbol{s}'_1, \boldsymbol{s}'_2, \ldots, \boldsymbol{s}'_n)$, we have

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

which concludes the proof.

10.2 Computing the channel matrix

We now show the detailed combination of matrices of subprocesses under 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 $P = \bigoplus_i p_i P_i$: Intuitively, P's scheduler selects for each P_i a scheduler from $\Delta(P_i)$. Hence $\Delta(P) = \{\bigcup_i \zeta_i \,|\, \zeta_i \in \Delta(P_i)\}$, and $|\Delta(P)| = \prod_i |\Delta(P_i)|$.
  For the scheduler $\zeta = \bigcup_i \zeta_i$, where $\zeta_i \in \Delta(P_i)$,

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

- Case $P = \biguplus_i r_i.P_i$: 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 $\Delta_i(P) = \{\, \{\zeta \mid \zeta(\varepsilon) = r_i \wedge \zeta(r_i\alpha) = \zeta_i(\alpha)\,\} \mid \zeta_i \in \Delta(P_i)\,\}$. Thus $\Delta(P) = \bigcup_i \Delta_i(P)$, and $|\Delta(P)| = \sum_i |\Delta(P_i)|$.
  For the scheduler $\zeta = \{\zeta(\varepsilon) = r_i \wedge \zeta(r_i\alpha) = \zeta_i(\alpha)\}$, where $\zeta_i \in \Delta(P_i)$,

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

- Case $P' = (\nu o)P$: Intuitively, the scheduler $\zeta|_o$ is obtained from ζ by removing the assignments of the executions containing o. Hence $\Delta(P') = \{\, \zeta|_o \,|\, \zeta \in \Delta(P)\}$, and $|\Delta(P')| = |\Delta(P)|$.
  For the scheduler $\zeta' = \zeta|_o$, where $\zeta \in \Delta(P)$,

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

  where the function $f(\boldsymbol{o})$ is shown in the formula (1).
- 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 $\Delta(P') = \{\zeta_1;\zeta_2;\cdots;\zeta_n \,|\, \zeta_i \in \Delta(P)\}$, and $|\Delta(P')| = |\Delta(P)|^n$.
  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. For the scheduler $\zeta' = \zeta_1;\zeta_2;\cdots;\zeta_n$, where $\zeta_i \in \Delta(P)$ for all i,

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