

SECURE COMPUTATION IN A PROBABILISTIC POLYNOMIAL-TIME PROCESS CALCULUS

P. Mateus¹, J. Mitchell², and A. Scedrov³

Abstract

In this paper, we present a general definition for secure computation following the general paradigm: a protocol is secure iff it can emulate an ideal protocol.

We start by adopting the probabilistic polynomial-time process calculus, originally presented in [LMMS98], as the natural language to specify protocols. Given the nature of the adversaries together with the secrecy requirements of secure computation protocols, we need to augment the calculus with local output channels and priority terms. Capitalizing on the semantics of the calculus, we extract a Markov process of observations and establish the notion of emulation. After representing the ideal secure computation protocol in the calculus, we present the concept of secure computation via emulation and obtain the associated composition theorem, encompassing both active and passive adversaries. To illustrate the concepts and results in an intuitive and simple manner, we focus the simpler case of oblivious transfer (OT).

Finally, we show that our concept of secure OT is equivalent to the concept formalized by Canetti in [Can00], which is based on interactive Turing machines, and discuss the advantages and simplifications introduced by this work.

Keywords: Secure computation, probabilistic process calculus, computational security, composition theorem.

⁽¹⁾ Center for Logic and Computation, Departamento de Matemática, IST, Lisboa, Portugal.

email: pmat@math.ist.utl.pt

Partially supported by FCT grant SFRH/BPD/5625/2001, and by FCT project Fiblog POCTI/2001/MAT/37239.

⁽²⁾ Department of Computer Science, Stanford University, Stanford, USA.

email: mitchell@cs.stanford.edu

Partially supported by OSD/ONR MURI “Semantic Consistency in Information Exchange” as ONR Grant N00014-97-1-0505, and by OSD/ONR CIP/SW URI “Software Quality and Infrastructure Protection for Diffuse Computing” as ONR Grant N00014-01-1-0795.

⁽³⁾ Department of Mathematics, University of Pennsylvania, Philadelphia, USA.

email: scedrov@saul.cis.upenn.edu

Partially supported by OSD/ONR MURI “Semantic Consistency in Information Exchange” as ONR Grant N00014-97-1-0505, by OSD/ONR CIP/SW URI “Software Quality, Infrastructure Protection for Diffuse Computing” as ONR Grant N00014-01-1-0795, and by NSF Grant CCR-0098096.

SECURE COMPUTATION IN A PROBABILISTIC POLYNOMIAL-TIME PROCESS CALCULUS

1 Introduction

Throughout the last years, designing secure protocols has been one of the most prolific areas in cryptography. The branch which deals with secure computation has not been an exception [AF90, Bea91, GM95, HM00]. The overall goal of this cryptographic task is to evaluate publicly a function f , guaranteeing in the presence of adversary behavior of some part of the system, the secrecy of the arguments of f , which are held by the participants. The fact that adversaries are part of the protocol distinguishes this problem from most of the other cryptographic tasks, like authentication, where adversaries are external to the protocol.

The main objective of this paper is to establish a rigorous notion of secure computation. For the sake of simplicity, we focus a very particular case of secure computation called oblivious transfer (OT). Oblivious transfer protocols are building blocks for several other multi-party secure computation protocols, like the circuit evaluation protocol [AF90]. Therefore, establishing an accurate concept of secure OT is of utmost importance. As discussed in [MR91], any notion of security must be endowed with a composition theorem, which loosely speaking states that composing secure protocols is secure. The composition theorem is specially relevant for OT, since, as discussed before, it is a common component of other protocols.

A formalization of secure computation was already established by Canetti [Can00] using interactive Turing machines. The framework of interactive Turing machines, even if optimal to deal with complexity results, is not abstract enough to specify elegantly cryptographic protocols. Moreover, this framework emerges from relevant problems connecting cryptography and complexity, and therefore, much effort must be put to obtain orthogonal results, such as the envisaged composition theorem. On the other hand, process algebras are endowed with a precise syntax and semantics, which provides a good framework to specify protocols and obtain compositional results.

Process calculi were first used in security analysis by Roscoe [Ros95], Schneider [Sch96], and Abadi and Gordon [AG99]. In the latter, the spi-calculus was established by augmenting the π -calculus [Mil89] with cryptographic primitives. However, since we address the setting where the agents of the protocols are bounded by probabilistic polynomial-time computation (computational security), we consider the probabilistic calculus by Mitchell et al [MRST01].

The above mentioned probabilistic process algebra considers both public and private channels for agent communication. However, for multi-party secure computation it is enough to consider private channels alone (given that adversaries are agents of the protocols). Therefore, we require a new type of channel to generate outputs, which we call local output channels. This augmentation is due to the nature of the adversaries of secure computation protocols, which are part of the protocol, instead of exterior to the protocol. This issue is discussed throughout Section 2, for the OT case.

In Section 3 we present the augmented calculus syntax and its semantics. As discussed before, we only consider private and local output channels. This restriction is broad enough to represent both the OT agents and its adversaries. We follow the common paradigm to define security via emulation, which boils down to the following: a real protocol is secure iff it can emulate an ideal protocol. Hence, a general concept of secure computation is achieved. Finally, a composition theorem for this abstract notion of security is derived.

In Section 4 we present oblivious transfer within the probabilistic process algebra. First, the ideal oblivious transfer protocol is represented in the calculus, and finally, the functionality of oblivious

transfer is set. Next, we present the concept of secure oblivious transfer protocol. Since oblivious transfer is a two-party protocol, the notion of security is split in two, sender and receiver security. This notion is shown to be compositional, and this result is discussed throughout the section. Finally, we show that the notion of secure oblivious transfer is equivalent to the notion of oblivious transfer formalized with interactive Turing machines by Canetti in [Can00] (as a particular case of secure computation). Finally, we generalize the results to secure computation in Section 5.

The contributions of this paper are threefold. First, we obtain a notion of secure computation in the context of process algebras encompassing both active and passive adversaries. Second, we show an abstract composition theorem for the proposed notion secure computation. Finally, we establish an important relationship with the interactive Turing machine framework, by finding an equivalence between our notion of security and the formalization by Canetti.

2 Overview of the problem

In this section we discuss the nature of the adversaries in secure computation protocols and its impact in the calculus and in the notion of security. This section is entirely dedicated to motivate our approach on the definition of secure computation. For illustration purposes, we consider oblivious transfer, a relevant case of secure computation.

Oblivious transfer is a two-party protocol, where one agent is called the sender and the other the receiver. The sender S holds a vector of bits \vec{b} and the receiver R wishes to know the i -th bit of this vector. We can split the goal of the protocol in two parts: R obtains the required i -th bit without knowing further bits of \vec{b} ; S does not get any information about i . If one can trust a third party T , there is a trivial protocol which implements securely OT: the sender sends \vec{b} to T and the receiver sends i to T , then T sends \vec{b}_i to the receiver. This description establishes what is called the ideal protocol I , which is constituted by three parties running in parallel, the sender, the receiver and the trusted party: $I(\vec{b}, i) = S(\vec{b})|R(i)|T$.

If there are no trusted parties (or they are too expensive), then finding protocols that implement OT is not a trivial task (see [Rab81]). The obvious next step is to show that these protocols are secure. A protocol $Q(\vec{b}, i)$ implements oblivious transfer iff $Q(\vec{b}, i)$ can be split in two processes, the sender $Q_S(\vec{b})$ and the receiver $Q_R(i)$, and if after running $Q(\vec{b}, i)$ the receiver knows \vec{b}_i . By secure implementation we mean that the receiver can not trick the sender to know more than \vec{b}_i , and that the sender does not get any information about i . We call the former property *sender security* and the latter *receiver security*. Hence, a oblivious transfer protocol is secure iff it is both sender and receiver secure.

Local output channels are used to model what agents know. For the particular case of OT, the receiver should know b_i after running the protocol, therefore we oblige the receiver to output this value through a local output channel. Note that this output can not be seen by the sender, or else the sender would be able to gather information about i .

We define security by taking advantage of the ideal process $I(\vec{b}, i) = S(\vec{b})|R(i)|T$. We say that Q is sender secure iff for any real adversary receiver A for $Q_S(\vec{b})$ there exists an ideal adversary receiver B for $S(\vec{b})|T$ (the ideal sender) such that all local outputs $Q_S(\vec{b})|A$ are computationally indistinguishable from the outputs of $S(\vec{b})|T|B$. An ideal adversary receiver B is any process which interacting with $S(\vec{b})|T$ can at most know one projection of \vec{b} , while a real adversary receiver could be any process. Similarly, Q is receiver secure iff for any real adversary sender A for $Q_R(i)$ there exists an ideal adversary sender B for $R(i)|T$ such that all local outputs $Q_R(i)|A$ are computationally indistinguishable from $R(i)|T|B$. In this case, an ideal adversary sender B is any process which

interacting with $R(\vec{b})|T$ can not obtain any information about i , while, once again, a real adversary sender could be any process.

Ideal adversaries are usually bounded by what they are allowed to know, on the other hand, real adversaries are not bounded at all (except by the probabilistic polynomial time computation power). Hence we say that a protocol Q securely implements an ideal protocol I iff for any (unbounded) real adversary A we can find a (bounded) ideal adversary B such that we cannot efficiently distinguish $Q|A$ from $I|B$, where the distinction is based on the local outputs. This definition clearly motivates that observations over a process are obtained from local outputs.

We proceed by presenting the probabilistic process calculus, its semantics and the envisaged notion of emulation. After these steps are made, we establish the notion of secure implementation and discuss its compositional properties.

3 Probabilistic process calculus

In this section, we augment the probabilistic process calculus originally established in [LMMS98, LMMS99, MRST01], envisaging the use of such calculus to characterize secure computation, and in particular oblivious transfer. As traditional for the secure computational setting, we just consider the case where private channels are used in the protocols, which for oblivious transfer is enough, and for general secure computation encompasses most cases (sometimes called private computation protocols). The remaining cases, where public channels are required, can be dealt by asserting that private channels can be emulated by public channels plus encryption, as stated in [CK01].

3.1 Syntax

We assume as given once and for all a countable set of local output channels $U = \{u_1, \dots, u_n, \dots\}$ and a countable set of private channels $V = \{v_1, \dots, v_n, \dots\}$. We take these sets to be disjoint and call $C = U \cup V$ the set of all channels.

Both private and output channels are designated channels which can not be tapped. While for private channel outputs can be issued and inputs received, local output channels can only be used to send outputs. Moreover, mind that the values issued through output channels correspond to local knowledge of the process, and are not meant to be known by all the system, but only by the agent which has issued them. Local output channels model (in the process algebra) the local outputs already present in joint computations of interactive Turing machines, and will be central to establish an equivalence relation between processes, since all the information communicated through private channels is hidden.

Whenever security protocols are considered, it is common to introduce a security parameter $n \in \mathbb{N}$. From now on, the symbol n is reserved to designate such security parameter. One role of this parameter is to bound the bandwidth of the channels, hence, we introduce a *bandwidth map* $w : C \rightarrow \mathbf{q}$, where \mathbf{q} is the set of all polynomials taking positive values. Hence, given a value n for the security parameter, a channel c can send messages with at most $w(c)(n)$ bits.

We take as black-box the set of probabilistic polynomial terms T (endowed with a set of variables Var) established in [MMS98]. The most relevant result on T is that for any probabilistic polynomial (on the security parameter n) time function f there is a term t such that the associated probabilistic Turing machine M_t computes f . Furthermore, given a term t the associated probabilistic Turing machines M_t is always probabilistic polynomial time (on n).

Definition 3.1 The *language of processes* \mathcal{L} is obtained inductively as follows:

1. $0 \in \mathcal{L}$ (termination);
2. $end \in \mathcal{L}$ (priority termination);
3. $eager(Q) \in \mathcal{L}$ where $Q \in \mathcal{L}$ (priority);
4. $\langle t \rangle_c \in \mathcal{L}$ where $t \in T$ and $c \in C$ (output);
5. $(x)_v.Q \in \mathcal{L}$ where $v \in V$, $x \in Var$ and $Q \in \mathcal{L}$ (input);
6. $[t_1 = t_2].Q$ where $t_1, t_2 \in T$ and $Q \in \mathcal{L}$ (match);
7. $Q_1|Q_2$ where $Q_1, Q_2 \in \mathcal{L}$ (parallel composition);
8. $!_q Q$ where $Q \in \mathcal{L}$ and $q \in \mathbf{q}$ (iteration).

Given a process term Q , we denote the set of output channels occurring in Q by $U_Q \subset U$.

After fixing the security parameter n , we evaluate each iteration $!_q R$ as $q(n)$ copies of R in parallel. Hence, given a process term Q we denote by \bar{Q}_n the process without iteration where we replace in Q each subterm $!_q R$ by $q(n)$ copies of R in parallel.

We depart from the usual π -calculus notation by not considering a binding private channel operator ν and by changing the position of the channel labels, for instance we use $(x)_v$ instead of $v(x)$.

3.2 Semantics

The semantics of a process term is a Markov chain over multisets of a special kind of process terms, which we call eligible.

Definition 3.2 The set of eligible processes \mathcal{E} is defined inductively as follows:

- $0 \in \mathcal{E}$;
- $\langle t \rangle_c \in \mathcal{E}$ where $t \in T$ and $c \in C$;
- $(x)_v.Q \in \mathcal{E}$ where $v \in V$, $x \in Var$ and $Q \in \mathcal{L}$;
- $[t_1 = t_2].Q \in \mathcal{E}$ where $t_1, t_2 \in T$ and $Q \in \mathcal{L}$;
- $eager(Q) \in \mathcal{E}$ where $Q \in \mathcal{E}$.

In order to present the operational semantics, we set some notation on finite multisets. A finite multiset \mathcal{M} over a set L is a map $\mathcal{M} : L \rightarrow \mathbb{N}$ such that $\mathcal{M}^{-1}(\mathbb{N}_1)$ is finite. The difference of \mathcal{M} and \mathcal{M}' is the multiset $\mathcal{M} \setminus \mathcal{M}'$ where $(\mathcal{M} \setminus \mathcal{M}')(l) = \max(0, \mathcal{M}(l) - \mathcal{M}'(l))$. The union of two multisets \mathcal{M} and \mathcal{M}' is the multiset $\mathcal{M} \cup \mathcal{M}'$ where $(\mathcal{M} \cup \mathcal{M}')(l) = \mathcal{M}(l) + \mathcal{M}'(l)$. We say that $\mathcal{M} \subset \mathcal{M}'$ iff $\mathcal{M}(l) \leq \mathcal{M}'(l)$ for all $l \in L$. Furthermore, we say that $l \in \mathcal{M}$ iff $\{l\} \subset \mathcal{M}$. Finally, we call $\wp_{fm}(L)$ the set of all finite multisets over L .

Definition 3.3 Given a process $Q \in \mathcal{L}$ without iteration we obtain the *multiset of sequences* of Q , which we denote by \mathcal{M}_Q , as follows:

- $\mathcal{M}_Q = \{\}$ whenever $Q = 0$;
- $\mathcal{M}_Q = \{Q\}$ whenever Q is eligible and different from 0;

- $\mathcal{M}_Q = \bigcup_{Q'' \in \mathcal{M}_{Q'}} \{eager(Q'')\}$ whenever $Q = eager(Q')$;
- $\mathcal{M}_Q = \mathcal{M}_{Q'} \cup \mathcal{M}_{Q''}$ whenever $Q = Q' | Q''$.

Given a multiset of process terms \mathcal{M} we call $eager(\mathcal{M})$ the submultiset of \mathcal{M} which contains all process terms of \mathcal{M} with head $eager(\cdot)$ and $removeager(\mathcal{M}) = \{Q : eager(Q) \in eager(\mathcal{M})\}$, that is, the multiset where we have removed $eager(\cdot)$ from the processes in $eager(\mathcal{M})$.

Instead of presenting of the semantics with probabilistic labeled transition systems, we use the more elegant and well established notation from the stochastic processes community, following the style in [MPP⁺01].

Definition 3.4 A scheduler S for a security parameter n is a Markov chain with state space $\wp_{fm}(\mathcal{E})$ such that $S(\mathcal{M}_1, \mathcal{M}_2)^1 > 0$ iff one of the following (disjoint) conditions hold:

i) (end of priority)

- $eager(end) \in eager(\mathcal{M}_1)$;
- $\mathcal{M}_2 = (\mathcal{M}_1 \setminus eager(\mathcal{M}_1)) \cup (removeager(\mathcal{M}_1) \setminus \{end\})$;
- $S(\mathcal{M}_1, \mathcal{M}_2) = 1$.

This transition corresponds to the end of priority. If $eager(end)$ occurs in $eager(\mathcal{M}_1)$ then the process terms in $eager(\mathcal{M}_1)$ are no longer the first to be reduced. This transition occurs with probability 1.

ii) (homogeneous priority)

- $eager(\mathcal{M}_1) \neq \{\}$;
- $removeager(\mathcal{M}_1)$ is not an absorbing state of S ;
- $\mathcal{M}_2 = (\mathcal{M}_1 \setminus eager(\mathcal{M}_1)) \cup eager(\mathcal{M}_2)$;
- $S(\mathcal{M}_1, \mathcal{M}_2) = S(removeager(\mathcal{M}_1), removeager(\mathcal{M}_2)) > 0$.

This transition is probable whenever there are eager terms in \mathcal{M}_1 which induce a proper transition ($removeager(\mathcal{M}_1)$ is not an absorbing state of S). The transition has the same probability of the transition where the non-eager terms are disregarded.

iii) (heterogeneous priority)

- $eager(\mathcal{M}_1) \neq \{\}$;
- $removeager(\mathcal{M}_1)$ is an absorbing state of S ;
- $removeager(\mathcal{M}_1) \cup \{Q'\}$ is not an absorbing state of S where $Q' \in \mathcal{M}_1$;
- $\mathcal{M}_2 = (\mathcal{M}_1 \setminus (eager(\mathcal{M}_1) \cup \{Q'\})) \cup \{eager(Q) : Q \in \mathcal{M}_3\}$;
- $S(eager(Q') \cup \{Q'\}, \mathcal{M}_3) > 0$.

This transition is probable whenever the eager terms of \mathcal{M}_1 alone can not trigger a proper transition, but with a non eager term $Q' \in \mathcal{M}_1$ are able to induce a transition. The transition has the same probability of the transition where the non-eager terms but Q' are disregarded. Note that heterogeneous priority transitions do not occur if it is possible to make a homogeneous transition, that is, heterogeneous transitions have lower priority than homogeneous.

¹ $S(\mathcal{M}_1, \mathcal{M}_2)$ denotes the probability of going from \mathcal{M}_1 to \mathcal{M}_2 in S .

iv) (communication)

- $eager(\mathcal{M}_1) = \{\}$;
- $\{\langle t \rangle_v, (x)_v.Q\} \subset \mathcal{M}_1$;
- t does not have any free variables, and t evaluates to m' with positive probability and m corresponds to the first $w(v)(n)$ bits of m' ;
- $\mathcal{M}_2 = (\mathcal{M}_1 \setminus \{\langle t \rangle_v, (x)_v.Q\}) \cup \mathcal{M}_{Q_m^x}$ where Q_m^x stands for the process where we substitute all (free) occurrences of x by m in Q .

This transition is probable whenever there are no priority terms to reduce and there is a pair input/output for a private channel v in \mathcal{M}_1 .

v) (local output)

- $eager(\mathcal{M}_1) = \{\}$;
- $\langle t \rangle_u \in \mathcal{M}_1$;
- t does not have any free variables;
- $\mathcal{M}_2 = \mathcal{M}_1 \setminus \{\langle t \rangle_u\}$.

This transition is probable whenever there are no priority terms to reduce and there is an output for a local output channel u in \mathcal{M}_1 .

vi) (match)

- $eager(\mathcal{M}_1) = \emptyset$;
- $[t_1 = t_2].Q \in \mathcal{M}_1$ where t_1, t_2 are closed terms;
- $P(t_1 = t_2) > 0$;
- $\mathcal{M}_2 = (\mathcal{M}_1 \setminus \{[t_1 = t_2].Q\}) \cup \mathcal{M}_Q$.

This transition is probable whenever there are no priority terms to reduce, there is an match term in \mathcal{M}_1 and t_1 evaluates to the same value than t_2 with positive probability.

vii) (mismatch)

- $eager(\mathcal{M}_1) = \emptyset$;
- $[t_1 = t_2].Q \in \mathcal{M}_1$;
- $P(t_1 \neq t_2) > 0$;
- $\mathcal{M}_2 = (\mathcal{M}_1 \setminus \{[t_1 = t_2].Q\})$.

This transition is probable whenever there are no priority terms to reduce, there is an match term in \mathcal{M}_1 and t_1 evaluates to a different value than t_2 with positive probability.

viii) (termination)

- $\mathcal{M}_1 = \mathcal{M}_2$;
- $S(\mathcal{M}_1, \mathcal{M}_2) = 1$;
- all other transitions are not probable for \mathcal{M}_1 .

Whenever all reductions were made, the only enable transition is the loop to the same state, which means that the reduction has terminated and therefore \mathcal{M}_1 is an absorbing state.

Schedulers are expected to have the following good properties:

- **channel and variable independence:** probabilities are independent of the names of the channels and variables, that is
 - $S(\mathcal{M}_1, \mathcal{M}_2) = S(\mathcal{M}_{1y}^x, \mathcal{M}_{2y}^x)$ provided that x occurs free with respect to y in all process terms of \mathcal{M}_1 and \mathcal{M}_2 ;
 - $S(\mathcal{M}_1, \mathcal{M}_2) = S(\mathcal{M}_{1d}^c, \mathcal{M}_{2d}^c)$ where $\omega(d) = \omega(c)$ and d does not occur in all process terms of \mathcal{M}_1 and \mathcal{M}_2 ;
- **environment independence:** probabilities are independent of the processes terms which are not involved in the transition, that is

$$S(\mathcal{M}_1, \mathcal{M}_2 | \mathcal{M} \subseteq \mathcal{M}_1 \cap \mathcal{M}_2) = S(\mathcal{M}_1 \setminus \mathcal{M}, \mathcal{M}_2 \setminus \mathcal{M});$$

- **computational efficiency:** the scheduler is modeled by a probabilistic polynomial time Turing machine.

It is straightforward to check that the scheduler that gives uniform distribution to all possible transitions verifies the above properties. Given a scheduler, the operational semantics for a process term Q is straightforward to obtain.

Definition 3.5 Given a process Q and a scheduler S , the *operational semantics of Q* is the subMarkov chain $S(Q)$ of S consisting of all states reachable from \mathcal{M}_Q such that $S(Q)^0 = \mathcal{M}_Q$ that is, the initial state of $S(Q)$ is \mathcal{M}_Q .

Note that the loops in $S(Q)$ are the absorbing transitions, and hence, all the states are either transient or absorbing. This fact implies that for k sufficiently large we have $P(S(Q)^k = S(Q)^{k+m}) = 1$ for all $m \in \mathbb{N}$. In other words, any random sampling of $S(Q)$ will end in an absorbing state, and therefore $S(Q)$ always terminates.

3.3 Observations

In order to establish the observations of a process term Q , we consider a modulated Markov process $K(Q) = (S(Q), O(Q))$ where $O(Q)$ is the stochastic process of observations of Q . The latter process can be set for the entire scheduler, as follows.

Definition 3.6 Given a scheduler S and a security parameter n , we define the *observation modulated Markov process* $K = (S, O)$ where O is a stochastic process over $(U \times \mathbb{N}) \cup \{\tau\}$ such that:

- $K((\mathcal{M}_1, o_1), (\mathcal{M}_2, o_2)) = S(\mathcal{M}_1, \mathcal{M}_2) \times p$ whenever one of the following conditions hold:
 - $o_2 = (u, m)$ and the channel u outputs t in the transition of \mathcal{M}_1 to \mathcal{M}_2 in S , the probability of t evaluating to m' is p and m corresponds to the first $w(u)(n)$ bits of m' ;
 - $o_2 = \tau, p = 1$ and the transition from \mathcal{M}_1 to \mathcal{M}_2 in S was not an output transition.

- $K((\mathcal{M}_1, o_1), (\mathcal{M}_2, o_2)) = 0$ for all other cases.

Observe that K is indeed a Markov process modulated by S , since there exists a function f such that:

$$K((\mathcal{M}_1, o_1), (\mathcal{M}_2, o_2)) = S(\mathcal{M}_1, \mathcal{M}_2)f(\mathcal{M}_1, \mathcal{M}_2).$$

Once again, by restricting K to $S(Q)$ we obtain the required modulated stochastic process of observations $K(Q)$ for the process term Q . For the sake of easing notation we denote the set of all observations by $Ob = (U \times \mathbb{N}) \cup \{\tau\}$.

It is obvious that sampling over the observation process generates a sequence of observations, this induces a family of random quantities, usually called trace.

Definition 3.7 Given an observation process $K = (S, O)$ we obtain a *trace process* $\{T^k\}_{k \in \mathbb{N}}$ where T^k is a random variable over Ob^k such that

$$P(T^k = o_1 \dots o_k) = \sum_{l_1 < \dots < l_k} P\left(\bigwedge_{i=1}^{l_k} O^i = c_i\right)$$

where $c_i = \begin{cases} o_j & \text{pt } i = l_j \\ \tau & \text{otherwise} \end{cases}$.

Hence, for instance, $P(T^2(\overline{Q}_n) = o_1 o_2)$ gives the probability of process \overline{Q}_n generating the output o_1 followed by o_2 .

As expected, the notion of emulation is based on trace processes. As we shall see, one process emulates another if its trace process is computationally indistinguishable to the trace process of the other.

3.4 Emulation

One of the most successful ways for defining secure concurrent cryptographic tasks is via process emulation [AG99, Can00]. This definitional job boils down to the following: a process realizes a cryptographic task iff it emulates an ideal process that is known to realize such task. In this section, we detail the notion of emulation for the previously established probabilistic process calculus, guided by the goal of defining secure computation.

Let I be an ideal protocol that realizes (the honest part of) some secure computation protocol. The set of free variables of I , which we denote by $FreeVar(I) = \{x_1, \dots, x_n\}$, corresponds to the inputs of the honest parties for the secure computation protocol. Since adversaries cannot change the inputs of the honest parties, we can model an adversary as a process running in parallel with I , following the style of [Can00]. Any process Q that implements the functionality specified by I must have the same inputs of I , and therefore $FreeVar(Q) = FreeVar(I)$.

The overall goal is to show that Q realizes without flaws (part of) a secure computation functionality, which is defined by I . The goal is achieved if for any probability distribution over the inputs $X_1 \dots X_k$ and real adversary $A \in \mathcal{A}$, the trace process of $Q_{X_1 \dots X_k}^{x_1 \dots x_k} | A$ is computationally indistinguishable from the trace process of $I_{X_1 \dots X_k}^{x_1 \dots x_k} | B$ for some ideal adversary $B \in \mathcal{B}$ where an ideal adversary is an adversary which cannot corrupt I and a real adversary is any process. This property asserts that given a probability distribution over the inputs of Q and I and a real adversary A we cannot distinguish efficiently the local outputs of $Q_{X_1 \dots X_k}^{x_1 \dots x_k} | A$ with the local outputs of the well behaved process $I_{X_1 \dots X_k}^{x_1 \dots x_k} | B$ for some $B \in \mathcal{B}$, and therefore, we infer that $Q_{X_1 \dots X_k}^{x_1 \dots x_k} | A$ is also well behaved.

Recall that we use local outputs to model what processes know, so if A is able to know efficiently something from $Q_{X_1 \dots X_k}^{x_1 \dots x_k}$ it should not, then A can issue a local output with such knowledge. In this case, we cannot find any ideal adversary B which is able to gather from $I_{X_1 \dots X_k}^{x_1 \dots x_k}$ similar knowledge (by choosing correctly the set of ideal adversaries), and hence, the trace process of $Q_{X_1 \dots X_k}^{x_1 \dots x_k} | A$ is not computationally indistinguishable from $I_{X_1 \dots X_k}^{x_1 \dots x_k} | B$ for all possible ideal processes $B \in \mathcal{B}$.

This discussion leads to the concept of emulation with respect to a set of real adversaries \mathcal{A} and ideal adversaries \mathcal{B} .

Definition 3.8 Let Q and I be process terms such that $FreeVar(Q) = FreeVar(I)$ and \mathcal{A} and \mathcal{B} sets of process terms, then Q emulates R with respect to \mathcal{A} and \mathcal{B} iff

$$\forall X_1 \dots X_k \in RV \forall q \in \mathbf{q} \forall A \in \mathcal{A} \exists B \in \mathcal{B} \forall \vec{u} \in U_{Q|D}^* \exists n_0 \in \mathbb{N} \forall n > n_0 \forall \vec{m} \in \mathbb{N}^{|\vec{u}|}$$

$$|P(T^{|\vec{u}|}(\overline{Q_{X_1 \dots X_k}^{x_1 \dots x_k} | A})_n) = (\vec{u}, \vec{m}) - P(T^{|\vec{u}|}(\overline{I_{X_1 \dots X_k}^{x_1 \dots x_k} | B})_n) = (\vec{u}, \vec{m})| \leq 1/q(n)$$

where $\{x_1 \dots x_k\} = FreeVar(Q)$.

A desirable property of the emulation relation is congruence, in other words the following should hold: if Q emulates I with respect to \mathcal{A} and \mathcal{B} then $C[Q]$ should emulate $C[I]$ with respect to \mathcal{A} and \mathcal{B} , for an arbitrary context $C[\]$. We only consider contexts which do not use iteration neither priority, it is easy to see that we can factorize any context $C[\]$ of this kind in a context having the following structure:

Definition 3.9 The set of *contexts* \mathcal{C} is defined inductively as follows:

- $[\] \in \mathcal{C}$;
- $(x)_v.C[\] \in \mathcal{C}$ provided that $C[\] \in \mathcal{C}$;
- $[x = 0].C[\] \in \mathcal{C}$ provided that $C[\] \in \mathcal{C}$;
- $C[\] | Q \in \mathcal{C}$ provided that $C[\] \in \mathcal{C}$ and $Q \in \mathcal{L}$ and $LocalOut(Q) = \emptyset$;
- $C[\] | (x)_v \langle x \rangle_u$.

Given a context $C[\]$ and a process Q , the notation $C[Q]$ means that we substitute the process Q for the $[\]$ in $C[\]$. Note that an adversary A of Q cannot interfere with private internal communications of Q , moreover, in order to show the congruence result we impose further good properties to the adversaries:

Definition 3.10 Let \mathcal{A} and \mathcal{B} be sets of adversaries and Q and I processes. We say that \mathcal{A} and \mathcal{B} are *structural* for Q and I iff

- $0 \in \mathcal{A}$ and $A | R \in \mathcal{A}$ provided that $A \in \mathcal{A}$ and $R \in \mathcal{L}$;
- \mathcal{A} and \mathcal{B} are α -renaming closed for both channels and variables;
- $B_{eager}^{(t)_u}(\langle t \rangle_v) | (x)_v \langle x \rangle_u \in \mathcal{B}$ provided that $B \in \mathcal{B}$ and v is not a channel of B ;
- $B | eager(\langle t \rangle_v) \in \mathcal{B}$ provided that $B \in \mathcal{B}$, $t \in T$ and v is not a channel of B .

The congruence results follows:

Proposition 3.11 Let Q be a protocol that emulates I with respect to \mathcal{A} and \mathcal{B} , then $C[R]$ emulates $C[I]$ with respect to \mathcal{A} and \mathcal{B} provided that \mathcal{A} and \mathcal{B} are structural for Q and I .

A corollary of Proposition 3.11 is the so called Composition Theorem. This result was first discussed informally for the secure computation setting in [MR91], and states the following: if R is a secure implementation of the ideal protocol I , Q and J are two protocols which use the ideal protocol I as a component, and finally, $Q[I]$ is a secure implementation of $J[I]$, then $Q[R]$ should be a secure implementation of $J[I]$. This result is captured as follows:

Corollary 3.12 Let R be a protocol that emulates I with respect to \mathcal{A} and \mathcal{B} and $Q[I]$ a protocol that emulates $J[I]$ with respect to \mathcal{A} and \mathcal{B} . Then $Q[R]$ emulates $J[I]$ whenever \mathcal{A} and \mathcal{B} are structural with respect to R and I , $\mathcal{B} \subseteq \mathcal{A}$ and $Q[] \in \mathcal{C}$.

Proof: By Proposition 3.11 we obtain that $Q[R]$ implements securely $Q[I]$, then by composition this equality with the hypothesis that $Q[I]$ implements securely $J[I]$ we get the desired result. \square

4 Oblivious transfer

Recall that oblivious transfer is a two party secure computation protocol where one agent is called the *sender* and the other the *receiver*. The sender holds a vector of bits \vec{b} and the receiver wishes to know the i -th bit of this vector. We can split the goal of the protocol in two parts: the receiver obtains the required i -th bit without knowing further bits of \vec{b} ; the sender never gets to know which bit was required from the receiver.

Ideally, this can be set in a context where there is a trusted and neutral party, $T = (\vec{x})_v \cdot (y)_{v'} \cdot \langle \vec{x}_y \rangle_{v''}$ that waits from the sender the vector of bits \vec{b} and from the receiver to value i and then sends to the receiver \vec{b}_i . The sender only sends to the trusted party \vec{b} , that is $S(\vec{b}) = \langle \vec{b} \rangle_v$. Finally, the receiver just sends i to the trusted party, waits for the value \vec{b}_i and then issues through a local output channel u this value, $R(i) = \langle i \rangle_{v'} | (z)_{v''} \cdot \langle z \rangle_u$ (the local output u models what the receiver knows in the end of the protocol). The ideal process corresponds to placing all this three processes in parallel, that is, $I(\vec{b}, i) = T | S(\vec{b}) | R(i)$. In practice, trusted parties can not be found, or are too expensive, and therefore a real OT protocol $Q(\vec{b}, i)$ contains only two agents, the sender $Q_S(\vec{b})$ and the receiver $Q_R(i)$. We now focus our attention in defining when is Q a secure OT protocol.

4.1 Secure oblivious transfer

For the sake of simplicity we start by assuming that only the receiver is an adversary. This means that the receiver is an evil party, and hence, it wants to know more than just the i -th bit from vector \vec{b} which is held by the sender. Having this in mind, consider again the ideal setting with a trusted party, but, since in this case we do not know the receiver behaviour, we must delete it from the ideal process, that is $I_S(\vec{b}) = T | S(\vec{b})$.

Ideally, an adversary receiver can only obtain a bit from \vec{b} , that is.

Definition 4.1 An *ideal receiver adversary* is a process B such that:

- $u \in U_B$;
- $\exists_{n_0} \forall_{n > n_0} \exists_i^1 \exists_{h: \mathbb{N} \rightarrow \mathbb{N} \in \text{PPT}}$ such that

$$P(H_{(I_S|B)_n}^u(\vec{b}) = x) = P(g(b_i) = x)$$

where $H_{(I_S|B)_n}^u : \mathbb{N}^k \rightarrow \mathbb{N}$ is the random function such that

$$P(H_{(I_S|B)_n}^u(\vec{b}) = x) = \sum_{s \in (\text{Ob} \setminus \{u\} \times \mathbb{N})^*} P(T^{|\vec{b}|+1}(\overline{(I_S(\vec{b})|B)_n}) = s(u, x)).$$

Note that $H_{(I_S|B)_n}^u(\vec{b})$ is the random variable which gives the output of $\overline{(I_S(\vec{b})|B)_n}$ via u . This condition imposes that the output via u must be a function of only one projection of \vec{b} .

We denote the *set of all ideal receiver adversaries* by \mathcal{I} .

Note that we impose the ideal adversary to output information through channel u , and that furthermore, this information can only be obtained (efficiently) through one bit of \vec{b} .

On the other hand, real adversaries, who will attack a real protocol, have no restriction whatsoever to the amount of information they might obtain from interacting with a real sender, and so, we can only assume they will locally output the information they got through the channel u .

Definition 4.2 An *real receiver adversary* is a process C such that $u \in U_C$. We call the set of all real receiver adversaries \mathcal{R} .

As expected, we say that an oblivious transfer protocol is sender secure, if the sender interacting with a real adversary simulates the ideal setting (that is the ideal process interacting with an ideal adversary). In detail we have:

Definition 4.3 Let $Q(\vec{b}, i) = \langle Q_S(\vec{b}), Q_R(i) \rangle$ be an oblivious transfer protocol, we say that $Q(\vec{b}, i)$ is sender secure iff Q_S emulates I_S for all data \vec{b} with respect to \mathcal{R} and \mathcal{I} .

It is straightforward to set the dual notion of *receiver secure* OT protocol.

4.2 Composition Theorem

We now focus our attention to compositionality. Once again, the fact that adversaries are part of the protocol, instead of exterior to the protocol, make compositionality harder to present. First we must understand what does it mean to a protocol to call, for instance, an OT protocol, in a modular way.

Assume that $Q(\vec{b}, i) = (Q_S(\vec{b}), Q_R(i))$ is a protocol that implements securely the OT functionality. Consider now an ideal two party protocol $J(x, y) = (J_1(x), J_2(y))$ which realizes some other cryptographic task and another protocol $U(x, y)$ which implements J securely for some set of adversaries. Assume that J requires an OT from agent 1 to agent 2, and therefore uses the ideal protocol I mentioned above. Hence, the first agent of J contains an ideal sender, in other words, is of the form $J_1(x)[S(f(x))]$, the second agent contains an ideal receiver, so is of the form $J_2(y)[R(g(y))]$, and, moreover, J must contain the trusted party T . We can think of J as a protocol running in parallel all three elements, that is $J(x, y) = J_1(x)[S(f(x))]|J_2(y)[R(g(y))]|T$. Now, assume that $U(x, y)$ also uses the ideal OT protocol and therefore, making a similar analysis, is of the form $U(x, y) = U_1(x)[S(f(x))]|U_2(y)[R(g(y))]|T$. What the composition theorem states is that if we replace the ideal sender and the ideal receiver in U by the sender and the receiver of Q (and get rid of the trusted party), then the new protocol $U'(x, y)$ still implements securely $J(x, y)$.

There are still several issues underspecified in all that has been said on compositionality. For instance, how does J_2 call the ideal receiver R with the value $g(y)$ and receives its output? In other words, how is modular call modeled in the process algebra?

Consider the ideal receiver $R(i)$ as an example, this process contains the variable i which carries the value of the input of the receiver. One can pass the value 3, for instance, to i in the following $(i)_v R(i) | \langle 3 \rangle_v$. Finally, one can receive the output of the receiver (which is issued by the output channel u), by augmenting the receiver in order to send the local output through a private channel v' , which then can be used to receive the value in some context. For instance consider $R' = R_{\langle t \rangle_u | \langle t \rangle_{v'}}$ and obtain this value for Q' as follows $R'(i) | (x)_v' Q'$. It is easy to check that R and R' have the same behaviour with respect to real adversaries, provided that these adversaries do not read through channel v' . Observe that both the ideal receiver $R(i)$ and the real receiver $Q_R(i)$ must use the variable i to carry the input and the channel u to output, and therefore we can replace one by the other in $J_2(x) | (i)_c R(i) | \langle t(x) \rangle$.

Given the previous discussion, the composition theorem can be stated as follows.

Theorem 4.4 Let $J(x, y) = J_1(x) | J_2(y)$ be an ideal protocol, $U(x, y) = U_1(x) | U_2(y)$ be a protocol and $Q(\vec{b}, i) = Q_S(\vec{b}) | Q_R(i)$ be a OT protocol such that:

- J and U have an ideal OT as a component, that is $J(x, y) = J_1(x) | S(f(x)) | J_2(y) | R'(g(y)) | T$ and $U(x, y) = U_1(x) | S(f(x)) | U_2(y) | R'(g(y)) | T$;
- U_1 emulates J_1 with respect to \mathcal{R} and \mathcal{I} and $U_1[] \in \mathcal{C}$, that is, $U_1[]$ is a context.;
- Q_S emulates $S | T$ with respect to \mathcal{R} and \mathcal{I} ;

Then $\tilde{U}_1 = U_1[Q_S(f(-))]$ emulates J_1 with respect to \mathcal{R} and \mathcal{I} .

Proof: Straightforward by Corollary 3.12 and by observing that \mathcal{R} and \mathcal{I} are structural for Q_S and $S | T$. \square

We state the above property by saying that *sender security is compositional with respect to \mathcal{C}* .

4.3 Related notions of security

In this section we compare the notion of secure oblivious transfer established in Definition 4.3 with the formalization due to Canetti in [Can00]. The result follows from the auxiliar representation result between process algebra terms and (polynomial) interactive Turing machines which we start by presenting. Please refer to [Gol01] to recall the notion of interactive Turing machine.

Proposition 4.5 Let M and N be two interactive Turing machines with common input n (both probabilistic and polynomial on n) and with auxiliar input x and y respectively, then there are two process terms $Q(x)$ and $R(y)$ with security parameter n and output channels u_1 and u_2 such that $P(\langle M(x), N(y) \rangle(n) = \langle m_1, m_2 \rangle) =$

$$P(T_{Q(x)|R(y)} \in \{ \langle u_1, m_1 \rangle . \langle u_2, m_2 \rangle, \langle u_2, m_2 \rangle . \langle u_1, m_1 \rangle \}).$$

Moreover the converse also holds.

This result allow us to compare the previously established notion of security with the notion proposed by Canetti for non-adaptive adversaries in the computational setting [Can00] (we specialize the definition for the simpler case of a sender secure OT protocol and do not distinguish the receiver from the adversary).

Proposition 4.6 A protocol is sender secure by Canneti’s notion (Definition A.1) iff it is sender secure according to Definition 4.3. In other words, let M_S , and N_R be interactive Turing machines that implement an OT protocol (as a sender and receiver) and are sender secure in the computational setting according to Definition A.1 then the associated OT protocol specified with process terms is sender secure according to Definition 4.3. Moreover, the converse also holds.

Proof: Straightforward application of Proposition 4.5 and by noticing that the ideal and real adversaries coincide. \square

5 Secure computation

In this section we generalize the concepts and results from oblivious transfer to secure computation in general.

Recall that the overall objective of secure computation, originally proposed in [Yao82], is to compute a function (eventually random) publicly, maintaining its arguments secret. To fully characterize this task, consider k agents A_1, \dots, A_k who wish to evaluate the random function $f : \mathbb{N}^k \rightarrow \mathbb{N}^k$. Moreover, each agent A_i holds a secret argument d_i . A secure computation protocol should provide a value of $f_i(\vec{d})$ to A_i and guarantee that both this value and d_i remain secret (except to A_i).

Note that oblivious transfer is a particular case of secure computation, where there are two agents, the receiver R and the sender S . The secret data for the sender is \vec{b} and for the receiver is i , moreover, $f_S(\vec{b}, i) = b_i$ and $f_R(\vec{b}, i)$ is irrelevant.

Once again, it is trivial to find a secure computation protocol if there is an outside trusted agent T . In this case, all agents A_i just need to send their secret data d_i to T and then, wait for T to give them back the value $f_i(\vec{d})$. This is precisely what happens when we vote, we expect it to be secret and that the state, which is to be trusted, provide us the final result. In the process algebra, we could model this trusted agent by $T = (x_1)_{v_1} \dots (x_n)_{v_1} (\langle f_1(\vec{x}) \rangle_{v'_1} | \dots | \langle f_k(\vec{x}) \rangle_{v'_k})$ and the other agents by $Q_i(d_i) = \langle d_i \rangle_{v_i} | (r)_{v'_i} \langle r \rangle_{b_i}$.

One hurdle consists in finding protocols which provide the above mentioned functionality without a trusted party. Examples of these protocols can be found over the literature. It is out of the scope of this paper to present these protocols and hence, in the sequel we will just denote by $Q(\vec{d})$ such a protocol.

The next immediate hurdle corresponds to show that $Q(\vec{d})$ is secure, and this is the problem addressed in the next section.

5.1 Secure computation analysis

The overall idea in secure computation analysis is to consider the adversary as a set of agents who are corrupted by an evil hacker, or are just colluding against the other agents. Following this line of reasoning, consider a fixed number of agents $h \leq k$ which are not corrupted. Since we do not know exactly what the remaining $(h - k)$ are going to do, even in the ideal setting of a trusted party, we obtain the following ideal process

$$I(\vec{d}) = T | Q_1(d_1) | \dots | Q_h(d_h)$$

Moreover, we split the real protocol $Q(\vec{a} \cdot \vec{d})$ in two $A(\vec{a})$ and $H(\vec{d})$ where $A(\vec{a})$ corresponds to the part of the protocol supposedly to be run by the corrupted agents and $H(\vec{d})$ to the part which corresponds to the honest players.

Now we say that $Q(\vec{a}, \vec{d})$ is secure for $H(\vec{d})$ iff $H(\vec{d})$ can simulate $I(\vec{d})$. It is usual to consider two types of attack: passive attack (where the adversaries try to get some information from the secret data or the outcome of f for other agents); active attack (where the adversaries try to change the value of f given to the parties). The sender secure oblivious transfer presented in Section 4.1 correspond to security against passive attack. Once again, the Canneti's concept of secure computation presented in [Can00] merges with ours, the details are omitted for the sake of space.

5.2 Passive adversaries

For passive adversaries the following is expected to happen: the ideal adversary set of parties may only obtain, by running the protocol, their input data and their correspondent projections of f , in other words:

Definition 5.1 An *ideal passive adversary* is a process B such that:

- $u \in U_B$;
- $\exists_{n_0} \forall_{n > n_0} \exists_{a \in \mathbb{N}^{k-h}} \exists_{g: \mathbb{N}^{2(k-h)} \rightarrow \mathbb{N} \in \text{PPT}}$ such that

$$P(H_{(I|B)_n}^u(\vec{d}) = x) = P(g(a_1, \dots, a_{k-h}, f_1(\vec{a} \cdot \vec{d}), \dots, f_{k-h}(\vec{a} \cdot \vec{d})) = x)$$

where $H_{(I|B)_n}^u : \mathbb{N}^h \rightarrow \mathbb{N}$ is the random function such that

$$P(H_{(I|B)_n}^u(\vec{d}) = x) = \sum_{s \in (\text{Ob} \setminus \{u\} \times \mathbb{N})^*} P(T^{|s|+1}(\overline{(I(\vec{d})|B)_n}) = s(u, x)).$$

We call the set of all ideal receiver adversaries \mathcal{I} .

Note that we impose the ideal adversary to output information through channel u , and that furthermore, this information can only be obtained (efficiently) through the data which the adversary is allowed to retrieve.

Once again, real adversaries have no restriction whatsoever to the amount of information they might obtain from interacting with a real sender, except that they will locally output the information they got through channel u .

Definition 5.2 An *real passive adversary* is a process A such that $u \in U_A$. We call the set of all real passive adversaries \mathcal{R} .

As expected, we say that a secure computation protocol is secure for passive adversaries iff the honest parties interacting with a real passive adversary simulate the ideal setting (that is the ideal process interacting with an ideal passive adversary). In detail we have:

Definition 5.3 Let $Q(\vec{a} \cdot \vec{d}) = \langle A(\vec{a}), H(\vec{d}) \rangle$ be a secure computation protocol, we say that $Q(\vec{d})$ is *secure for passive attacks of A* iff H emulates I with respect to \mathcal{R} and \mathcal{I} .

The next best thing happens:

Proposition 5.4 The notion of security for passive attacks is compositional with respect to \mathcal{C} .

Proof: Straightforward by Proposition 3.11 and by observing that \mathcal{R} and \mathcal{I} is structural.

In the sequel, we focus our attention to active adversaries. As we shall see, active security is easier to establish than passive.

5.3 Active adversaries

For the case of active adversaries, we do not care about the information the adversary may gather in the end of the protocol, but instead, we only impose that the adversary is not able to interfere with the outcome of f .

Definition 5.5 Let $Q(\vec{a}, \vec{d}) = \langle A(\vec{a}), H(\vec{d}) \rangle$ be a secure computation protocol, we say that Q is secure for active attacks of A iff H emulates I with respect to \mathcal{L} and \mathcal{L} .

Hence, we consider both the set of ideal adversaries and real adversaries to be the set of all processes. Clearly, if a real adversary is able to change the outputs of u_i in $H(\vec{d})$ for some $i \in 1 \dots h$ then the protocol is not secure, because there is no adversary that can achieve this for $I(\vec{d})$. Compositionality also works for this case.

Proposition 5.6 The notion of security for passive attacks is compositional with respect to \mathcal{C} .

Proof: Straightforward by Proposition 3.11 and by observing that \mathcal{L} is structural.

6 Conclusions

The main results obtained in this paper are threefold. First, we have settled a notion of secure computation for both passive and active adversaries in the context of a probabilistic polynomial-time process calculus, giving special attention to relevant case of oblivious transfer. Second, we provided an abstract composition theorem. Finally, we showed that the notion of security is equivalent to the notion formalized by Canetti [Can00], based on interactive Turing machines.

We stress that there are several advantages of using a calculus instead of interactive Turing machines, in particular to address problems related with secure protocols. First, the process algebra is a much more natural and clear language to specify protocols than interactive Turing machines, in other words, the process algebra syntax is much closer to the actual language used to implement protocols.

Another advantage consists on the fact that compositional issues are dealt in a much more clear way using process algebras. This property is due to the fact that congruence is a standard concept for an algebra. Observe that to show the composition theorem, it is enough to prove a congruence property for the emulation relation. The candidate for the congruence relation is obvious: if A emulates B then $C[A]$ emulates $C[B]$ for any context C .

Finally, we note that the framework on process algebras seems to be very promising result wise, having in mind applications in security. Moreover, probabilistic polynomial-time process algebras are the adequate setting to grasp the concepts related to computational security, since both the parties and the adversaries are, in practice, bounded by probabilistic polynomial-time algorithms. Therefore, this work should be counted as one more contribution to the more general effort of searching an abstract notion of security for all possible cryptographic tasks.

The next step starting from this work consists in adding public channels, and henceforward consider protocols with public communication. The work on secure channels by [CK01], wherein private channels are emulated by public channels endowed with encryption, tailors the line of research that shall be pursued on this topic. Namely, we expect in the near future to find a secure channel compiler that maps a process term containing just private channels to an equivalent process containing just public channels. Another on going work consists in actually applying the definition to a particular protocol like the circuit-evaluation protocol [AF90]. In the long run, we expect to axiomatize the calculus along with the emulation relation (or a related relation) in such a way, that we could infer security syntactically from the specification.

References

- [AF90] M. Abadi and J. Feigenbaum. Secure circuit evaluation. *Journal of Cryptology*, 2(1):1–12, 1990.
- [AG99] M. Abadi and A. Gordon. A calculus for cryptographic protocols: the spi-calculus. *Information and Computation*, 143:1–70, 1999.
- [Bea91] D. Beaver. Secure multi-party protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(1):75–122, 1991.
- [Can00] R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
- [CK01] R. Canneti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT’01*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer-Verlag, 2001.
- [GM95] R. Gennaro and S. Micali. Verifiable secret sharing as secure computation. In *Eurocrypt’95*, volume 921 of *Lecture Notes in Computer Science*, pages 168–182. Springer-Verlag, 1995.
- [Gol01] O. Goldreich. *Foundation of Cryptography (Fragments of a Book)*. Weizmann Inst. of Science, 2001. (Available at <http://philby.ucsd.edu/cryptolib.html>).
- [HM00] M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation. *Journal of Cryptology*, 13(1):31–69, 2000.
- [LMMS98] P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. Probabilistic polynomial-time framework for protocol analysis. In M. Reiter, editor, *5-th ACM Conferece on Computer and Communication Security*, pages 112–121. ACM Press, 1998.
- [LMMS99] P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. Probabilistic polynomial-time equivalence and security analysis. In *Formal Methods in the Development of Computing Systems*, volume 1708 of *Lecture Notes in Computer Science*, pages 776–793. Spriger-Verlag, 1999.
- [Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
- [MMS98] J. Mitchell, M. Mitchell, and A. Scedrov. A linguistic characterization of bounded oracle computation and probabilistic polynomial time. In *39-th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 725–733. IEEE Computer Society Press, 1998.
- [MPP⁺01] P. Mateus, A. Pacheco, J. Pinto, A. Sernadas, and C. Sernadas. Probabilistic situation calculus. *Annals of Mathematics and Artificial Intelligence*, 32(1/4):393–431, 2001.
- [MR91] S. Micali and P. Rogaway. Secure computation. In *CRYPTO’91*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404. Springer-Verlag, 1991.
- [MRST01] J. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for analysis of cryptographic protocols. *Electronic Notes in Theoretical Computer Science*, 45, 2001.

- [Rab81] M. Rabin. How to exchange secrets by oblivious transfer. Tech. memo TR-81, Aiken Computation Laboratory, Harvard U., 1981.
- [Ros95] A. W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. In *8-th IEEE Computer Security Foundations Workshop (CSFW)*. IEEE Computer Society Press, 1995.
- [Sch96] S. Schneider. Security properties and CSP. In *IEEE Symp. Security and Privacy*, 1996.
- [Yao82] A. Yao. Protocols for secure computation. In M. Reiter, editor, *23rd Annual Symp. on Foundations of Computer Science (FOCS)*, pages 80–91. IEEE Press, 1982.

A Related concepts

Definition A.1 (Canetti) Let $Q = (S, R)$ be an oblivious transfer protocol. We say that Q is sender secure in the computational setting iff for any PPT receiver limited real adversary A there exists a PPT ideal receiver adversary I such that:

$$\text{IDEAL}_{OT,I} \stackrel{c}{\approx} \text{EXEC}_{S,A}.$$

where:

- S, R, A, I are all PPT (over the security parameter n) interactive Turing machines;
- $\stackrel{c}{\approx}$ stands for computationally indistinguishable.
- $\text{EXEC}_{S,A}$ is the probabilistic ensemble

$$\{\text{EXEC}_{S,A}(n, \vec{b}, i)\}_{n \in \mathbb{N}, (\vec{b}, i) \in \{0,1\}^*}$$

where $\text{EXEC}_{S,A}(n, \vec{b}, i)$ denotes the distribution over the outputs of A (since this is an execution of an OT protocol we do not require the sender to generate an output) after the following computation:

1. The sender starts with the security parameter n and input \vec{b} (we consider that the machine is probabilistic in the transitions which is equivalent to having a random input). The adversary starts with the security parameter n and input i (possibly modified).
 2. Initialize the round number $r = 0$;
 3. As long as the sender does not halt, repeat:
 - (a) the sender generates the message m_r to the adversary;
 - (b) the adversary learns m_r and generates m'_r ;
 - (c) the sender receives m'_r ;
 - (d) $r = r + 1$;
 4. The adversary generates an output.
- $\text{IDEAL}_{OT,I}$ is the probabilistic ensemble

$$\{\text{IDEAL}_{OT,I}(n, \vec{b}, i)\}_{n \in \mathbb{N}, (\vec{b}, i) \in \{0,1\}^*}$$

where $\text{IDEAL}_I(n, \vec{b}, i)$ denotes the distribution over the outputs of I after the following computation:

1. The sender starts with the security parameter n and input \vec{b} (we consider that the machine is probabilistic in the transitions which is equivalent to having a random input). The adversary starts with the security parameter n and input i (possibly modified);
2. the sender sends \vec{b} to the trusted party and the adversary also sends a value j (eventually different from i) to the trusted party. The trusted party sends b_j to the adversary;
3. The adversary generates and output that is a (probabilistic polynomial time on n) function of i and b_j .

B Proofs

Proof of Proposition 3.10 By structural induction on $C[]$

- Base: trivial
- $(y)_v.C[]$: First consider that $y \notin \text{FreeVar}(Q)$, then $(y)_v.Q_{\vec{X}}^{\vec{x}}|A$ is computationally indistinguishable from

$$Q_{\vec{X}}^{\vec{x}}|eager((y)_v.end|A).$$

By hypothesis there is B' such that $I_{\vec{X}}^{\vec{x}}|B'$ is computationally indistinguishable from

$$Q_{\vec{X}}^{\vec{x}}|eager((y)_v.end|A),$$

which is also indistinguishable from

$$(x)_v I_{\vec{X}}^{\vec{x}}|eager(\langle 0 \rangle_v)|B' = (x)_v I_{\vec{X}}^{\vec{x}}|B''.$$

If $y \in \text{FreeVar}(Q)$ then let Y be the (polynomial-time) distribution of the output of v in $(y)_v.Q_{\vec{X}}^{\vec{x}}|A$. By hypothesis there is B such that $Q_{\vec{X},Y}^{\vec{x},y}|eager(A_{end}^{(t)_v})$ is computationally indistinguishable from $I_{\vec{X},Y}^{\vec{x},y}|B$, which is computationally indistinguishable from

$$(y)_v I_{\vec{X}}^{\vec{x}}|B|eager(\langle Y \rangle_v).$$

- $[x = 0].C[]$: Let A be an adversary for $[x = 0]Q$. Since $x \in \text{OpenVar}([x = 0]Q)$, then just choose as ideal adversary B such that $I_{\vec{X}}^{\vec{x}}|B$ is computationally indistinguishable from $Q_{\vec{X}}^{\vec{x}}|A$.
- $C[]|R$, where Q has no local outputs: let $A \in \mathcal{A}$, since \mathcal{A} is structural, there exists $B \in \mathcal{B}$ such that $Q_{\vec{X}}^{\vec{x}}|R_{\vec{X}}^{\vec{x}}|A$ is computationally indistinguishable from $I_{\vec{X}}^{\vec{x}}|B$, we can re-label B and I such that no private channels in common exist with $R_{\vec{X}}^{\vec{x}}$ and therefore $I_{\vec{X}}^{\vec{x}}|B|R_{\vec{X}}^{\vec{x}}$ is also computationally indistinguishable from $I_{\vec{X}}^{\vec{x}}|B$ and B is the required real adversary.
- $C[]|(x)_v \langle x \rangle_u$: let $A \in \mathcal{A}$, since \mathcal{A} is structural, there exists $B \in \mathcal{B}$ such that $Q_{\vec{X}}^{\vec{x}}|(x)_v \langle x \rangle_u|A$ is computationally indistinguishable from $I_{\vec{X}}^{\vec{x}}|B$, just replace in B every occurrence of $\langle t \rangle_u$ by $eager(\langle t \rangle_v)$, where v does not occur in $B|I$ and obtain $B' \in \mathcal{B}$. Then $I_{\vec{X}}^{\vec{x}}|B$ is computationally indistinguishable from $I_{\vec{X}}^{\vec{x}}|B'|(x)_v \langle x \rangle_u$.

□

Lemma B.1 Let M be an interactive Turing machine with input n which is probabilistic and polynomial on n for any interaction with another (probabilistic and polynomial on n) interactive Turing machine, then the following family of functions is probabilistic polynomial (on n) time:

- $\{f_s^n : \mathbb{N} \times N_n \rightarrow 2\}_{n \in \mathbb{N}}$ where:
 - $N_n = \bigcup_{i=0}^{q(n)} \{0, \dots, q(n)\}^i$ (we call ε the element of $\{0, \dots, q(n)\}^0$);
 - $f_s(x, \hat{r})$ take the value 1 whenever $M(n, x)$ switches after receiving \hat{r} from the read-only communication tape² and 0 otherwise;
- $\{f_c^n : \mathbb{N} \times N_n \rightarrow \mathbb{N}\}_{n \in \mathbb{N}}$ where $f_c^n(x, \vec{r})$ gives the value of the write-only communication tape of $M(n, x)$ after receiving \hat{r} from the read-only communication tape;
- $\{f_o^n : \mathbb{N} \times N_n \rightarrow \mathbb{N}\}_{n \in \mathbb{N}}$ where $f_o^n(x, \vec{r})$ gives the value of the output tape of $M(n, x)$ after receiving \hat{r} from the read-only communication tape.

Proof: To obtain a probabilistic Turing machine that calculates $f_s^n : \mathbb{N} \times N_n \rightarrow 2$ just consider M as a simple probabilistic Turing machine (assume that it has associated the bit 0 for identification) and enrich it so that whenever it switches, the value \hat{r}_i is placed in the read-only communication tape, iterating i from 1 to, at most, $|\hat{r}|$ or until M halts. Finally, consider as result the value in the switch tape. Since both the size of \hat{r}_i and $|\hat{r}|$ are bounded by a polynomial on n , the resulting Turing machine can still be made polynomial on n . The other Turing machines are obtained similarly. \square

Proof of Proposition 4.5: First observe that there exists a polynomial $q(\cdot)$ such that the number of messages exchanged by $M(n, x)$ and $N(n, y)$ is less or equal to $q(n)$. Assume, without lack of generality, that M starts the computation. Consider $Q(x) = |_{|q(n)|} Q'(x) | \langle \varepsilon \rangle_{l_1} | \langle 0 \rangle_{h_2}$ and $R(y) = |_{|q(n)|} R'(y) | \langle \varepsilon \rangle_{l_2}$ where Q' and R' are as follows:

- $Q'(x) = (\hat{x}_l)_{l_1} \cdot (x_h)_{h_2} \cdot ($
 $[x_h = 0] \cdot (x_v)_{v_2} \cdot ($
 $[f_s^n(x, \hat{x}_l \cdot x_v) = 0] \cdot (\langle f_c^n(x, \hat{x}_l \cdot x_v) \rangle_{v_1} | \langle \hat{x}_l \cdot x_v \rangle_{l_1} | \langle 0 \rangle_{h_1}) |$
 $[f_s^n(x, \hat{x}_l \cdot x_v) = 1] \cdot (\langle f_o^n(x, \hat{x}_l \cdot x_v) \rangle_{u_1} | \langle 1 \rangle_{h_1})$
 $) |$
 $[x_h = 1] \cdot \langle f_o^n(x, \hat{x}_l \cdot x_v) \rangle_{u_1}$
 $)$
- $R'(y) = (\hat{y}_l)_{l_2} \cdot (y_h)_{h_1} \cdot ($
 $[y_h = 0] \cdot (y_v)_{v_1} \cdot ($
 $[g_s^n(y, \hat{y}_l \cdot y_v) = 0] \cdot (\langle g_c^n(y, \hat{y}_l \cdot y_v) \rangle_{v_2} | \langle \hat{y}_l \cdot y_v \rangle_{l_2} | \langle 0 \rangle_{h_2}) |$
 $[g_s^n(y, \hat{y}_l \cdot y_v) = 1] \cdot (\langle g_o^n(y, \hat{y}_l \cdot y_v) \rangle_{u_2} | \langle 1 \rangle_{h_2})$
 $) |$
 $[y_h = 1] \cdot \langle g_o^n(y, \hat{y}_l \cdot y_v) \rangle_{u_2}$
 $)$

Note that Q' and R' correspond to a message exchange round between M and N . The channels l 's are used to store the previous values of the read-only communication tape for each machine. Furthermore, the channels h 's are used to know if the machine emulated by the other process has halted or not and

²In order words, the interaction between M and another machine has produced \hat{r} in the read-only communication tape, that is, the first interaction the tape has the value \hat{r}_1 , and so on. Note that M does not require to halt after receiving \hat{r} .

finally the channels v 's are used to send to the other machine the value in the write only communication tape.

The family of functions f and g are the probabilistic polynomial terms given by Lemma B.1 for M and N respectively. Note that both Q and R have only one transition enable in each step, hence the scheduler does not play any role. It is now straightforward to check that the probabilities match.

The converse follows from the fact that the semantics of a process for a computational efficient scheduler can be computed in probabilistic polynomial time. \square