On Session Typing, Probabilistic Polynomial Time, and Cryptographic Experiments

Ugo Dal Lago
University of Bologna, Italy
INRIA Sophia Antipolis, France

Giulia Giusti
University of Bologna, Italy

Abstract

A system of session types is introduced as induced by a Curry Howard correspondence applied to bounded linear logic, suitably extended with probabilistic choice operators and ground types. The resulting system satisfies some expected properties, like subject reduction and progress, but also unexpected ones, like a polynomial bound on the time needed to reduce processes. This makes the system suitable for modelling experiments and proofs from the so-called computational model of cryptography.

2012 ACM Subject Classification Theory of computation → Process calculi; Theory of computation → Program semantics; Security and privacy → Mathematical foundations of cryptography

Keywords and phrases Session Types, Probabilistic Computation, Bounded Linear Logic, Cryptographic Experiments

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2022.37

Related Version *Full Version*: `https://arxiv.org/abs/2207.03360` [20]

Funding The authors are partially supported by the ERC CoG "DIAPASoN", GA 818616.

Acknowledgements The authors would like to thank the anonymous referees for the many insightful comments.

1 Introduction

Session types [30, 23, 31] are a typing discipline capable of regulating the interaction between the parallel components in a concurrent system in such a way as to *prevent* phenomena such as deadlock or livelock, at the same time *enabling* the parties to interact following the rules of common communication protocols. In the twenty-five years since their introduction, session types have been shown to be a flexible tool, being adaptable to heterogeneous linguistic and application scenarios (see, e.g., [45, 10, 32, 16]). A particularly fruitful line of investigation concerns the links between session-type disciplines and Girard's linear logic [26]. This intimate relationship, known since the introduction of session types, found a precise formulation in the work of Caires and Pfenning on a Curry-Howard correspondence between session types and intuitionistic linear logic [11], which has been developed along many directions [47, 48, 44, 18]. In Caires and Pfenning's type system, proofs of intuitionistic linear logic become type derivations for terms of Milner's π-calculus. Noticeably, typable processes satisfy properties (e.g. progress and deadlock freedom) which do not hold for untyped processes.

Process algebras, and in particular algebras in the style of the π-calculus, have been used, among other things, as specification formalisms for cryptographic protocols in the so-called *symbolic* (also known as *formal*) model of cryptography, i.e. in the model, due to Dolev and Yao [25], in which aspects related to computational complexity and probability theory, themselves central to the *computational* model, are abstracted away: strings become symbolic

expressions, adversaries are taken as having arbitrary computing power, and nondeterminism replaces probabilism in regulating the interaction between the involved parties. This includes π-calculus dialects akin to the applied π-calculus [2], or the spi-calculus [3].

Is it possible to model cryptographic protocols by way of process algebras in the so-called computational model *itself*? A widely explored path in this direction consists in the so-called *computational soundness* results for symbolic models, which have been successfully proved in the realm of process algebras [1, 17, 46, 5]. In computationally sound symbolic models, any computational attack can be simulated by a symbolic attack, this way proving that whenever a protocol is secure in the latter, it must be secure in the former, too. If one is interested in calculi precisely and fully capturing the computational model, computational soundness is not enough, i.e., one wants a model capturing all *and only* the computational adversaries. And indeed, there have been some attempts to define process algebras able to faithfully capture the computational model by way of operators for probabilistic choice and constraints on computational complexity [42]. The literature, however, is much sparser than for process algebras in symbolic style. We believe that this is above all due to the fact that the contemporary presence of probabilistic evolution and the intrinsic nondeterminism of process algebras leads to complex formal systems which are hard to reason about.

This paper shows that session typing can be exploited for the sake of designing a simple formal system in which, indeed, complexity constraints and probabilistic choices can be both taken into account, this way allowing for the modelling of cryptographic experiments. At the level of types, we build on the approach by Caires and Pfenning, refining it through the lenses of bounded linear logic, a logical system which captures polynomial time complexity in the sequential setting [27, 29], at the same time allowing for a high degree of intensional expressivity [21]. At the level of processes, we enrich proof terms with first-order function symbols computing probabilistic polytime functions, namely the basic building blocks of any cryptographic protocol. This has two consequences: process evolution becomes genuinely probabilistic, while process terms and types are enriched so as to allow for the exchange of strings, this way turning the calculus to an applied one. From a purely definitional perspective, then, the introduced calculus, called π**DIBLL**, is relatively simple, and does not significantly deviate from the literature, being obtained by mixing well-known ingredients in a novel way. The calculus π**DIBLL** is introduced in Section 3 below.

Despite its simplicity π**DIBLL** is on the one hand capable of expressing some simple cryptographic experiments, and on the other hand satisfies some strong meta-theoretical properties. This includes type soundness, which is expected, and can be spelled out as *subject reduction* and *progress*, but also a polynomial bound on the length of reduction sequences, a form of *reachability* property which is essential for our calculus to be considered a model of cryptographic adversaries. All this is described in Section 4.

As interesting as they are, these properties are not by themselves sufficient for considering π**DIBLL** a proper calculus for computational cryptography. What is missing, in fact, is a way to capture computational indistinguishability, in the sense of the computational model [35, 28]. Actually, this is where the introduced calculus shows its peculiarities with respect to similar calculi from the literature, and in particular with respect to the CCS-style calculus by Mitchell and Scedrov [42]. Indeed, π**DIBLL** typable processes enjoy a confluence property which cannot hold for untyped processes. The latter, in turn, implies that firing internal actions on any typable process results in a *unique* distribution of processes, all of them ready to produce an observable action. This makes relational reasoning handier. We in particular explore observational equivalence in Section 5, then showing how this can be of help in a simple experiment-based security proof in Section 6.

$\mathsf{PrivK}^{eav}_{A,\Pi}(n)$:

$m_0, m_1 \leftarrow A(1^n)$

$k \leftarrow Gen(1^n)$

$b \leftarrow \{0,1\}$

$c \leftarrow Enc(k, m_b)$

$g \leftarrow A(c)$

return $(b = g)$

(a) As Pseudocode.

$PRIVK_\Pi$:

input m_0 from adv;

input m_1 from adv;

`let` $k = gen()$ `in`

`let` $b = flipcoin()$ `in`

`let` $c = enc(k, m_b)$ `in`

output c to adv;

input g from adv;

`let` $r = eq(g, b)$ `in`

output r to exp;

(b) As a Process.

Figure 1 The Indistinguishability Experiment.

Due to space constraints, many details had to be elided, but can be found in the long version of this paper [20], which is available online.

2 A Bird's Eye View on Cryptographic Experiments and Sessions

In this section, we introduce the reader to cryptographic experiments, and we show how they and the parties involved can be conveniently modelled as session-typed processes. We will also hint at how relational reasoning could be useful in supporting proofs of security. We will do all this by way of an example, namely the one of private key encryption schemes and security against passive adversaries. We will try to stay self-contained, and the interested reader can check either this paper's extended version [20] or textbooks [35] for more details or for the necessary cryptographic preliminaries.

▶ **Definition 1** (Negligible Functions). *A function f from the natural numbers to the non-negative real numbers is negligible iff for every positive polynomial p there is an $N \in \mathbb{N}$ such that for all natural numbers $n > N$ it holds that $f(n) < 1/p(n)$.*

▶ **Definition 2** (Probabilistic Polynomial Time Algorithms). *An algorithm A is called probabilistic polynomial time (PPT in the following) iff it is randomized, and there exists a polynomial p which is an upper limit to the computational complexity of A regardless of the probabilistic choices made by the latter.*

Since the running time of any cryptographic algorithm has to be polynomially bounded w.r.t. the *value* of the security parameter n, the latter is passed in unary (i.e. as 1^n) to the algorithm, so that n is also a lower bound to the *length* of the input.

A private-key encryption scheme is a triple of algorithms $\Pi = (Gen, Enc, Dec)$, the first one responsible for key generation, the latter two being the encryption and decryption algorithms, respectively. When could we say that such a scheme Π is *secure*? Among the many equivalent definitions in the literature, one of the handiest is the one based on indistinguishability, which is based on the experiment PrivK^{eav} reported in Figure 1a exactly in the form it has in [35]. As the reader may easily notice, the experiment is nothing more than a randomized algorithm interacting with both the adversary A and the scheme Π. The interaction between PrivK^{eav} and the adversary A can be put in evidence by switching

to a language for processes, see Figure 1b. The process $PRIVK_\Pi$ communicates with the adversary through the channel *adv* and outputs the result of its execution to the channel *exp*. Apart from the fact that the adversary has been factored out, the process is syntactically very similar to the experiment PrivK^{eav}. Actually, we could have made the interaction between *PRIVK* and Π explicit by turning the latter into a process interacting with the former through a dedicated channel.

The interaction between an adversary *ADV* and $PRIVK_\Pi$ can be modelled through the parallel composition operator, i.e., by studying the behaviour of $ADV \mid PRIVK_\Pi$. As we will soon see, we would like the aforementioned parallel composition to output true on the channel *exp* with probability very close to $\frac{1}{2}$, and this is indeed what cryptography actually prescribes [35]. We should not, however, be too quick to proclaim the problem solved. What, for example, if *ADV* communicates with $PRIVK_\Pi$ in a way different from the one prescribed by the experiment, e.g. by *not* passing two strings to it, thus blocking the interaction? Even worse, what if $PRIVK_\Pi$ becomes the parallel composition $PRIVK \mid \Pi$ and *ADV* cheats on the communication by intercepting the messages exchanged between Π and *PRIVK*? These scenarios are of course very interesting from a security viewpoint, but we are not interested at those here: the only thing *ADV* is allowed to do is to send the two messages and to use its internal computational capabilities to guess the value b the experiment produces.

How to enforce all this at the level of processes? Actually, this is what session types are good for! It would be nice, for example, to be able to type the two processes above as follows:

$$adv : \mathbb{S}[p] \otimes \mathbb{S}[p] \otimes (\mathbb{S}[p] \multimap \mathbb{B}) \vdash PRIVK_\Pi :: exp : \mathbb{B}$$
$$\vdash ADV :: adv : \mathbb{S}[p] \otimes \mathbb{S}[p] \otimes (\mathbb{S}[p] \multimap \mathbb{B})$$

where $\mathbb{B}$ is the type of booleans and $\mathbb{S}[p]$ is the type of strings of length p. Moreover, we would like to somehow force a restriction νadv to be placed next to the parallel composition $ADV \mid PRIVK_\Pi$, so as to prescribe that *ADV* can only communicate with the experiment, and not with the outside world. Finally, we would like *ADV* to range over processes working in polynomial time. All this is indeed taken care by our session type discipline as introduced in Section 3.

But now, would it be possible to not only *express* simple cryptographic situations, but also to *prove* some security properties about them from within the realm of processes? As already mentioned, this amounts to requiring that for every efficient adversary (i.e. for every PPT algorithm) A it holds that $\Pr[\mathsf{PrivK}^{eav}_{A,\Pi}(n)] \leq \frac{1}{2} + \varepsilon(n)$, where ε is negligible. In the world of processes, this becomes the following equation

$$\nu adv.(PRIVK_\Pi \mid ADV) \sim FAIRFLIP_{exp} \tag{1}$$

where $FAIRFLIP_{exp}$ behaves like a fair coin outputting its value on the channel *exp*, and $\sim$ expresses approximate equivalence as induced by negligible functions. Making all this formal is nontrivial for at least three reasons:

- First of all, the statement only holds for *efficient* adversaries. The relation $\sim$, however specified, must then take this constraint into account.
- Secondly, the relation $\sim$ only holds in an approximate sense, and the acceptable degree of approximation crucially depends on n, the so-called security parameter. This is due to negligibly, without which cryptography would be essentially vacuous.
- Finally, the computational security of Π can at the time of writing be proved only based on *assumptions*, e.g. that one-way functions or pseudorandom generators exist. In other words, Equation (1) only holds in a conditional sense, and cryptographic proofs have to be structured accordingly.

The calculus **πDIBLL** successfully addresses all these challenges, as we are going to show in the rest of this paper.

3 Processes and Session Typing

This section is devoted to introducing π**DIBLL**, a variation on π**DILL** [11] in which a polynomial constraint on the replicated processes is enforced following the principles of bounded linear logic [27]. For the sake of properly representing cryptographic protocols in the computational model, π**DIBLL** is also equipped with indexed ground types and a notion of probabilistic choice.

3.1 Preliminaries

Preliminary to the definition of the π**DIBLL** session type system are three concepts, namely polynomials, probability distributions and indexed ground types. Let us start this section introducing polynomials.

▶ **Definition 3** (Polynomials)**.** Polynomial variables *are indicated with metavariables like* n *and* m*, and form a set* $\mathcal{PV}$*.* Polynomials expressions *are built from natural number constants, polynomial variables, addition and multiplication. A polynomial* p *depending on the polynomial variables* $\overline{n} = n_1, \ldots, n_k$ *is sometime indicated as* $p(n_1, \ldots, n_k)$ *and abbreviated as* $p(\overline{n})$*. Such a polynomial is said to be a* $\mathcal{V}$-polynomial *whenever all variables in the sequence* $\overline{n}$ *are in* $\mathcal{V} \subseteq \mathcal{PV}$*. If* $\mathcal{V} \subseteq \mathcal{PV}$*, any map* $\rho : \mathcal{V} \to \mathbb{N}$ *is said to be a* $\mathcal{V}$-substitution*, and the natural number obtained by interpreting any variable* $m \in \mathcal{V}$ *occurring in a* $\mathcal{V}$*-polynomial* p *with* $\rho(m)$ *is indicated just as* $p(\rho)$*. If* $\mathcal{V}$ *is a singleton* $\{n\}$*, the substitution mapping* n *to the natural number* i *is indicated as* ρ_i*, and* $\mathcal{V}$ *is indicated, abusing notation, with* n*.*

Distributions play a crucial role in probability theory and represent the likelihood of observing an element from a given set. In this paper, they will be the key ingredient in giving semantics to types and processes.

▶ **Definition 4** (Probability Distributions)**.** *A* probability distribution *on the finite set* A *is a function* $\mathcal{D} : A \to \mathbb{R}_{[0,1]}$ *such that:* $\sum_{v \in A} \mathcal{D}(v) = 1$*. A probability distribution is often indicated by way of the notation* $\{v_1^{r_1}, \ldots, v_m^{r_m}\}$ *(where* $v_1, \ldots, v_m$ *are distinct elements of* A*), which stands for the distribution* $\mathcal{D}$ *such that* $r_i = \mathcal{D}(v_i)$ *for every* $i \in \{1, \ldots, m\}$*. Given a probability distribution* $\mathcal{D}$ *on* A*, its* support $S(\mathcal{D}) \subseteq A$ *contains precisely those elements of* A *to which* $\mathcal{D}$ *attributes a strictly positive probability. The set of all probability distributions on a set* A *is indicated as* $\mathscr{D}(A)$*.*

In the computational model, the agents involved exchange binary strings. We keep the set of ground types slightly more general, so as to treat booleans as a separate type. As a crucial step towards dealing with polytime constraints, the type of strings is indexed by a polynomial, which captures the length of binary strings inhabiting the type.

▶ **Definition 5** (Ground Types)**.** Ground types *are expressions generated by the grammar* $B ::= \mathbb{B} \mid \mathbb{S}[p]$*, where* p *is a polynomial expression. A* $\mathcal{V}$*-ground type is a ground type* B *such that all polynomial variables occurring in it are taken from* $\mathcal{V}$*, and as such can be given a semantics in the context of a* $\mathcal{V}$*-substitution* ρ*:* $[\![\mathbb{B}]\!]_\rho = \{0, 1\}$ *,* $[\![\mathbb{S}[p]]\!]_\rho = \{0, 1\}^{p(\rho)}$*.*

The precise nature of any ground type $\mathbb{S}[p(\overline{n})]$ is only known when the polynomial variables in $\overline{n}$, which stand for so-called security parameters, are attributed a natural number value. For reasons of generality, we actually allow more than one security parameter, even if cryptographic constructions almost invariably need only one of them.

3.2 Terms

Terms are expressions which are *internally* evaluated by processes, the result of this evaluation having a ground type and being exchanged between the different (sub)processes.

Function Symbols. We work with a set $\mathcal{F}$ of *function symbols*, ranged over by metavariables like f and g. In the context of this paper, it is important that function symbols can be evaluated in probabilistic polynomial time, and this can be achieved by taking function symbols from a language guaranteeing the aforementioned complexity bounds [40, 22]. Each function symbol $f \in \mathcal{F}$ comes equipped with:

- A type $\mathit{typeof}(f)$ having the form $B_1, \ldots, B_m \to C$ where the B_i and C are $\{n\}$-ground types.
- A family $[\![f]\!] = \{[\![f]\!]_i\}_{i \in \mathbb{N}}$ of functions giving semantics to f such that $[\![f]\!]_i$ goes from $[\![B_1]\!]_{\rho_i} \times \ldots \times [\![B_m]\!]_{\rho_i}$ to $\mathscr{D}([\![C]\!]_{\rho_i})$, where $\mathit{typeof}(f)$ is $B_1, \ldots, B_m \to C$.
- We assume each function symbol f to be associated with an $\{n\}$-polynomial $\mathit{comof}(f)$ bounding the complexity of computing f, in the following sense: there must be a PPT algorithm $\mathit{algof}(f)$ which, on input 1^i and a tuple t in $[\![B_1]\!]_{\rho_i} \times \cdots \times [\![B_m]\!]_{\rho_i}$ returns in time at most $\mathit{comof}(f)(\rho_i)$ each value $x \in [\![C]\!]_{\rho_i}$ with probability $[\![f]\!]_i(t)(x)$, where $\mathit{typeof}(f) = B_1, \ldots, B_m \to C$.

Term Syntax and Semantics. Finally, we are able to define terms and values, which are expressions derivable in the following grammars:

$$a, b, c ::= v \mid f_p(v_1, \ldots, v_n) \qquad v, w ::= z \mid \mathsf{true} \mid \mathsf{false} \mid s.$$

Here f is a function symbol, p is a polynomial, true and false are the usual boolean constants, s is any binary string, and z is a *term variable* taken from a set $\mathcal{TV}$ disjoint from $\mathcal{PV}$. Terms are assumed to be well-typed according to an elementary type system that we elide for the sake of simplicity (see [20] for more details). Reduction rules between terms and distributions of values are given only for terms which are closed with respect to both term variables and polynomial variables. Here are the rules:

$$\frac{}{v \hookrightarrow \{v^1\}} \qquad \frac{}{f_i(v_1, \ldots, v_m) \hookrightarrow [\![f]\!]_i(v_1, \ldots, v_m)}$$

3.3 Processes

It is finally time to introduce the process terms of π**DIBLL**, which as already mentioned are a natural generalization of those of π**DILL** [11]. Due to space reasons, we concentrate on the aspects in which π**DIBLL** differs from π**DILL**, at the same time trying to be self-contained. More details can be found in [20].

▶ **Definition 6** (Process Syntax). *Given an infinite set of names, the set of processes, indicated with metavariables like P and Q is defined by the following grammar:*

$$\begin{aligned} P, Q ::= \; & 0 \mid P \mid Q \mid (\nu\ y)\ P \mid x\langle y\rangle.P \mid x(y).P \mid [x \leftarrow v] \mid \mathtt{let}\ x = a\ \mathtt{in}\ P \mid \\ & x.P \mid\ !x(y).P \mid x.\mathtt{inl}; P \mid x.\mathtt{inr}; P \mid x.\mathtt{case}(P, Q) \mid \\ & \mathtt{if}\ v\ \mathtt{then}\ P\ \mathtt{else}\ Q \end{aligned}$$

where x, y are channel names from a set $\mathcal{CV}$ such that $\mathcal{TV} \subseteq \mathcal{CV}$, a is a term and v is a value.

Most of the operators from the grammar above have the same meaning as in π**DILL**, and are standard. Let us briefly describe the novel ones. The process $[x \leftarrow v]$ *outputs* the value v on the channel x. Dually, the process $x.P$ *inputs* a value v through the channel x, substituting it for any free occurrence of x in the process P. The process $\texttt{let } x = a \texttt{ in } P$ serves to *evaluate* the term a, then substituting the outcome for x in P. As such, it is the operator incepting the probabilistic behaviour coming from terms into processes. Finally, the process $\texttt{if } v \texttt{ then } P \texttt{ else } Q$ is as expected a *conditional* construction; observe that the argument is a value, and thus does not need to be evaluated. The grammar of processes we have just introduced is perfectly adequate to represent the process $PRIVK_{\Pi}$ as from Figure 1b. As an example, a process $P \oplus Q$ which evolves as either P or Q depending on the outcome of the flip of fair random coin can be written as the process

$$\texttt{let } x = \mathit{flipcoin}() \texttt{ in if } x \texttt{ then } P \texttt{ else } Q,$$

where *flipcoin* is a function symbol such that $\mathit{typeof}(\mathit{flipcoin})$ is $\epsilon \to \mathbb{B}$ and the underlying algorithm $\mathit{comof}(\mathit{flipcoin})$ is the trivial constant-time procedure one can imagine.

The set of names occurring free in the process P (hereby denoted $\mathit{fn}(P)$) is defined as usual. The same holds for the capture avoiding substitution of a channel x or value v for y in a process P (denoted $P\{x/y\}$ and $P\{v/y\}$, respectively).

3.4 Process Reduction

Process reduction in π**DIBLL** is intrinsically probabilistic, and as such deserves to be described with some care. Structural congruence can be defined in a standard way as a binary relation $\equiv$ on processes, see [20] for a formal definition. The reduction relation between processes is *not* a plain binary relation anymore, and instead puts a process P in correspondence with a *distribution* $\mathscr{D}$ of processes, namely an object in the form $\{P_1^{r_1}, \ldots, P_m^{r_m}\}$, where the P_i are processes and the r_i are positive real numbers summing to 1. We write $P \to \mathscr{D}$ in this case.

Let us now consider reduction rules, starting from the one for the `let` construct, namely the only genuinely probabilistic one:

$$\frac{a \hookrightarrow \{v_i^{r_i}\}_{i \in I}}{\texttt{let } x = a \texttt{ in } P \to \{P\{v_i/x\}^{r_i}\}_{i \in I}}$$

The other new operators can be evaluated in the expected way, giving rise to trivial (i.e. Dirac) distributions:

$$\frac{}{[x \leftarrow v] \mid x.P \to \{P\{v/x\}^1\}}$$

$$\frac{}{\texttt{if true then } P \texttt{ else } Q \to \{P^1\}} \qquad \frac{}{\texttt{if false then } P \texttt{ else } Q \to \{Q^1\}}$$

Reduction axioms and rules from π**DILL** are all available in π**DIBLL**, but must be appropriately tailored to the presence of distributions:

$$\frac{Q \to \{Q_i^{r_i}\}_{i \in I}}{P \mid Q \to \{P \mid Q_i^{r_i}\}_{i \in I}} \qquad \frac{P \to \{Q_i^{r_i}\}_{i \in I}}{(\nu\ y)\ P \to \{(\nu\ y)\ Q_i^{r_i}\}_{i \in I}}$$

$$\frac{P \equiv R \qquad R \to \{Q_i^{r_i}\}_{i \in I} \qquad Q_i \equiv T_i \text{ for every } i \in I}{P \to \{T_i^{r_i}\}_{i \in I}}$$

3.5 Type System

Traditionally, session typing serves the purpose of guaranteeing *safety* properties, like the absence of deadlocks. In this paper, however, they also enforce some bounds on the *complexity* of the reduction process, and as such have to be made more restricted.

Types. First of all, let us introduce the language of types, which is defined as follows:

$$A, B \ ::= \ 1 \ \mid \ A \multimap B \ \mid \ !_p A \ \mid \ A \otimes B \ \mid \ A \oplus B \ \mid \ A \& B \ \mid \ \mathbb{B} \ \mid \ \mathbb{S}[p]$$

Again, most of the type constructions are taken from π**DILL**, but there are some significant and crucial differences:

- The indexed exponential type $!_p A$ takes the place of $!A$ as the type of replicated inputs; the presence of the polynomial p serves to impose an upper bound on the number of replicas, all of them of type A, which can be spawned.
- The ground types $\mathbb{B}$ and $\mathbb{S}[p]$ are available as session types too, and here become the type of channels carrying a single value.

Type Environments. In dual intuitionistic linear logic, and thus in π**DILL**, the typing environment to the left of the turnstyle is split into two parts, namely an unrestricted part Γ and a linear part Δ. Here, we adopt the same notation, but the unrestricted part is modified in such a way that the polynomial limitation is made explicit when giving types to channels. In π**DIBLL**, in other words, Γ contains assignments in the form $x_p : A$ where x is the name of an unrestricted channel, p is a polynomial and A is a type. Type environments, however, have to be further enriched with a *third* portion, denoted by Θ, consisting of assignments in the form $x : B$ where x is a term variable and B is a *ground* type. This reflects the possibility of having occurrences of *term* variables inside *processes*. The unrestricted portion of the type environment has to be manipulated with great care while typing processes, in particular in all binary typing rules. This requires the introduction of a (partial) binary operation $\boxplus$ on unrestricted type environments such that if

$$\Gamma = \{x^1_{p_1} : A_1, \ldots, x^n_{p_n} : A_n\}; \qquad \Xi = \{x^1_{q_1} : A_1, \ldots, x^n_{q_n} : A_n\};$$

then $\Gamma \boxplus \Xi$ is defined as $\{x^1_{p_1+q_1} : A_1, \ldots, x^n_{p_n+q_n} : A_n\}$. With the same hypotheses, we write that $\Gamma \sqsubseteq \Xi$ when $p_i \leq q_i$ for every $i \in \{1, \ldots, n\}$, this way turning $\sqsubseteq$ into a preorder.

Type Judgments. A *type judgment* is an expression in the form

$$\Gamma; \ \Delta; \ \Theta \ \vdash^{\mathcal{V}} \ P \ :: \ z : C$$

where Γ, Δ, Θ are the three aforementioned portions of the type environment, and P is a process offering a session of type C along the channel z. Polynomials can occur in type environments and types, and $\mathcal{V}$ serves to declare all variables which might occur in those polynomials. As in π**DILL**, we assume that all channels and variables declared in Γ, Δ, and Θ are distinct and different from z.

Typing Rules. One way π**DIBLL** deviates from π**DILL** are the typing rules related to the exponential connective !, namely those introducing the connective on the right and on the left, but also the rule capturing contraction, T_{copy}, and the so-called exponential cut rule.

$$\frac{\Gamma;\ \Delta;\ \Theta,\ x : B\ \vdash^{\mathcal{V}}\ Q\ ::\ T}{\Gamma;\ \Delta,\ x : B;\ \Theta\ \vdash^{\mathcal{V}}\ x.Q\ ::\ T}\ [TBL] \qquad \frac{\Theta \vdash^{\mathcal{V}} v : B}{\Gamma;\ \Delta;\ \Theta\ \vdash^{\mathcal{V}}\ [x \leftarrow v]\ ::\ x : B}\ [TBR]$$

$$\frac{\Theta \vdash^{\mathcal{V}} a : A \qquad \Gamma;\ \Delta;\ \Theta, x : A\ \vdash^{\mathcal{V}}\ P\ ::\ T}{\Gamma;\ \Delta;\ \Theta\ \vdash^{\mathcal{V}}\ \texttt{let}\ x = a\ \texttt{in}\ P\ ::\ T}\ [T\texttt{let}]$$

$$\frac{\Theta \vdash^{\mathcal{V}} v : \mathbb{B} \qquad \Gamma;\ \Delta;\ \Theta\ \vdash^{\mathcal{V}}\ P\ ::\ x : A \qquad \Gamma;\ \Delta;\ \Theta\ \vdash^{\mathcal{V}}\ Q\ ::\ x : A}{\Gamma;\ \Delta;\ \Theta\ \vdash^{\mathcal{V}}\ \texttt{if}\ v\ \texttt{then}\ P\ \texttt{else}\ Q\ ::\ x : A}\ [T\texttt{if}]$$

Figure 2 Typing Rules for Ground Types, the `let` construct, and Conditionals.

For example, the T_{copy} rule, which allows to type processes which perform output actions on an unrestricted channel $u_p \in \Gamma$, must now keep track of the increase in p:

$$\frac{\Gamma,\ u_p : A;\ \Delta,\ y : A;\ \Theta\ \vdash^{\mathcal{V}}\ P\ ::\ T}{\Gamma,\ u_{p+1} : A;\ \Delta;\ \Theta\ \vdash^{\mathcal{V}}\ (\nu\ y)\ u\langle y\rangle.P\ ::\ T}\ [T_{copy}]$$

This, however, is not enough to enforce the polynomial bounds we aim at from within the type system. Whenever typing a parallel composition $P \mid Q$, it is necessary to appropriately account for the use of any unrestricted channel by *both* P and Q, thus splitting the unrestricted part of the underlying type environment. More concretely, any typing rule dealing with parallel composition has to make appropriate use of the $\boxplus$ operator, applying to the two unrestricted environments at hand, the result of this sum thus weakened through the partial order $\sqsubseteq$. For example, the T_{cut} rule is modified as follows

$$\frac{\Gamma_1 \boxplus \Gamma_2 \sqsubseteq \Gamma \qquad \Gamma_1;\ \Delta_1;\ \Theta\ \vdash^{\mathcal{V}}\ P\ ::\ x : A \qquad \Gamma_2;\ \Delta_2,\ x : A;\ \Theta\ \vdash^{\mathcal{V}}\ Q\ ::\ T}{\Gamma;\ \Delta_1, \Delta_2;\ \Theta\ \vdash^{\mathcal{V}}\ (\nu\ x)\ (P \mid Q)\ ::\ T}\ [T_{cut}]$$

All typing rules involving a parallel composition between two processes have to be amended similarly. The remaining typing rules, and in particular those for the additive connectives $\oplus$ and $\&$, are exactly as in π**DILL**. We conclude by giving the typing rules for the new process operators, which are in Figure 2, and are all standard. It is worth observing that in these rules we implicitly refer to the type system for terms and values in the rules TBR, $T\texttt{if}$, and $T\texttt{let}$.

4 Safety and Reachability

In this section we prove some properties about the transition system induced by the reduction relation $\rightarrow$, as introduced in Section 3.4. Before delving into the details, a couple of remarks are in order. Although the relation $\rightarrow$ is defined for arbitrary processes, we will be concerned with the reduction of typable *closed* processes, namely those processes which can be typed under *empty* Θ and $\mathcal{V}$. In fact, reducing processes in which term variables occur free does not make sense when reduction is supposed to model computation (as opposed to equational reasoning), like here. When Θ or $\mathcal{V}$ are empty, we simply omit them from the underlying typing judgment. Reduction being probabilistic, it is convenient to introduce some other reduction relations, all of them derived from $\rightarrow$:

▶ **Definition 7** (Auxiliary Reduction Relations). *We first of all define a relation $\mapsto$ on plain processes by stipulating that $P \mapsto R$ iff $P \rightarrow \mathscr{D}$ and $R \in S(\mathscr{D})$. We also need another reduction relation $\Rightarrow$ as the monadic lifting of $\rightarrow$, thus a relation on process distributions:*

$$\frac{R_i \rightarrow \mathscr{E}_i \text{ for every } i \in I}{\{R_i^{r_i}\}_{i \in I} \Rightarrow \sum_{i \in I} r_i \cdot \mathscr{E}_i}$$

Finally, it is convenient to put any process P in relation with the distribution of irreducible *processes to which P evaluates:*

$$\frac{P \text{ is irreducible}}{P \Rrightarrow \{P^1\}} \qquad \frac{P \to \{R_i^{r_i}\}_{i \in I} \qquad R_i \Rrightarrow \mathscr{E}_i \text{ for every } i \in I}{P \Rrightarrow \sum_{i \in I} r_i \cdot \mathscr{E}_i}$$

The relation $\mapsto$ is perfectly sufficient to capture the qualitative aspects of the other reduction relations, e.g., if $P \to \mathscr{D}$ then for every $R \in S(\mathscr{D})$ it holds that $P \mapsto R$. Indeed, in the rest of this section we will be concerned with $\mapsto$, only.

4.1 Subject Reduction

The property of subject reduction is the minimal requisite one asks to a type system, and says that types are preserved along reduction. In π**DIBLL**, as in π**DILL**, this property holds:

▶ **Theorem 8** (Subject Reduction). *If $\Gamma;\ \Delta \vdash P :: z : C$ and $P \mapsto R$, then it holds that $\Gamma;\ \Delta \vdash R :: z : C$.*

Following [11], subject reduction can be proved by carefully inspecting how P can be reduced to R, which can happen as a result of either communication, the evaluation of a term, or the firing of a conditional construction. Many cases have to be analysed, some of them not being present in π**DILL**. A novel aspect is a lemma about the typing of processes obtained by substituting term variables with values. More details can be found in [20], where a progress property is also given, very much in the style of that from [11].

4.2 Polytime Soundness

As already mentioned in Section 1, subject reduction is not the only property one is interested in proving about reduction in π**DIBLL**. In fact, the latter has been designed to guarantee polynomial bounds on reduction time, as prescribed by the computational model of cryptography. But what do we mean by that, exactly? What is the underlying parameter on which the polynomial depends? In cryptography, computation time must be polynomially bounded on the value of the so-called *security parameter* which, as we hinted at already, is modelled by an element of $\mathcal{V}$. As a consequence, what we are actually referring to are bounds *parametric* on the value of the polynomial variables which P mentions in its type judgments, i.e. the $\mathcal{V}$ in

$$\Gamma;\ \Delta \vdash^{\mathcal{V}} P :: z : C \tag{2}$$

Doing so, we have to keep in mind that process reduction is only defined on *closed* processes. We can thus proceed in three steps:

- We can first of all assign a $\mathcal{V}$-polynomial $\mathbf{W}(\pi)$ to every type derivation π with conclusion mentioning $\mathcal{V}$. This is done by induction on the structure of π.
- We then prove that for closed type derivations, $\mathbf{W}(\cdot)$ strictly decreases along process reduction, at the same time taking the cost of each reduction step into account. In other words, if π is closed and types P where $P \mapsto Q$, then a type derivation ξ for Q can be found such that $\mathbf{W}(\pi) \geq \mathbf{W}(\xi) + k$, where k is the cost of the reduction leading P to Q. (In most cases k is set to be 1, the only exception being the evaluation of a `let` operator, which might involve the evaluation of costly functions.)

- Finally, the previous two points must be proved to interact well, and this is done by showing that for every type derivation π with conclusion in the form (2) and for every $\mathcal{V}$-substitution ρ, there is a type derivation $\pi\rho$ with conclusion

  $$\Gamma\rho;\ \Delta\rho\ \vdash\ P\rho\ ::\ z : C\rho$$

  such that, crucially, $\mathbf{W}(\pi\rho) = \mathbf{W}(\pi)\rho$. In other words, the weight functor on type derivations commutes well with substitutions.

Altogether, this allows us to reach the following:

▶ **Theorem 9** (Polytime Soundness). *For every derivation π typing P, there is a polynomial p_π such that for every substitution ρ, if $P\rho \mapsto^* Q$, then the overall computational cost of the aforementioned reduction is bounded by $p_\pi(\rho)$.*

We can thus claim that, e.g., every process ADV such that

$$\vdash^n ADV :: c : \mathbb{S}[p] \otimes \mathbb{S}[p] \otimes (\mathbb{S}[p] \multimap \mathbb{B})$$

can actually be evaluated in probabilistic polynomial time, since out of it one can type the processes R_0, R_1, R_{fun} computing the three components in ADV's type. Moreover, since $\mathcal{F}$ can be made large enough to be complete for PPT (see, e.g., [22]), one can also claim that all probabilistic (first-order) polytime behaviours can be captured from within π**DIBLL**.

5 Typable Processes and Their Probabilistic Behaviour

The properties we proved in the last section, although remarkable, are agnostic to the probabilistic nature of π**DIBLL**. It is now time to investigate the genuinely quantitative aspects of the calculus.

5.1 Confluence

It might seem weird that confluence can be proved for a calculus in which internal probabilistic choice is available. In fact, there are *two* forms of nondeterministic evolution π**DIBLL** processes can give rise to, the first one coming from the presence of terms which can evolve probabilistically, the second one instead due, as in π**DILL**, to the presence of parallel composition, itself offering the possibility of concurrent interaction. Confluence is meant to address the latter, not the former. More specifically, we would like to prove that whenever a typable process P evolves towards two distinct distributions $\mathscr{D}$ and $\mathscr{E}$, the latter can be somehow unified. This indeed turns out to be the case:

▶ **Theorem 10** (Confluence). *If $\Gamma;\ \Delta\ \vdash\ P\ ::\ T$ and $\mathscr{D} \leftarrow P \rightarrow \mathscr{E}$ then either $\mathscr{D} = \mathscr{E}$ or there exists $\mathscr{F}$ such that $\mathscr{D} \Rightarrow \mathscr{F} \Leftarrow \mathscr{E}$.*

In other words, while probabilistic evolution coming from terms is unavoidable, the choice of how to reduce a typable process does not matter, in the spirit of what happens in sequential languages like the λ-calculus. Notice, however, that confluence holds in a very strong sense here, i.e., Theorem 10 has the flavour of the so-called diamond property. The proof of it, which can be found in [20], exploits a characterization of reduction semantics by way of labelled semantics, the latter rendering the proof simpler.

Among the corollaries of confluence, one can prove that the way a typable process is reduced is irrelevant as far as the resulting distribution is concerned:

▶ **Corollary 11** (Strategy Irrelevance). *If $\Gamma;\ \Delta\ \vdash\ P\ ::\ T$ and $\mathscr{D} \Leftarrow P \Rrightarrow \mathscr{E}$, then $\mathscr{D} = \mathscr{E}$.*

5.2 Relational Reasoning

Strategy irrelevance has a positive impact on the definition of techniques for relational reasoning on typed processes. In this section, we are concerned precisely with introducing a form of observational equivalence, which turns out to have the shape one expects it to have in the realm of sequential languages. This is a simplification compared to similar notions from the literature on process algebras for the computational model of cryptography [42].

When defining observational equivalence, we want to dub two processes as being equivalent when they behave *the same* in *any environment*. We thus have to formalize environments and observable behaviours. The former, as expected, is captured through the notion of a *context*, which here takes the form of a term in which a single occurrence of the hole $[\cdot]$ is allowed to occur *in linear position* (i.e. outside the scope of the any replication).

$$\begin{aligned}\mathscr{C}[\cdot] ::= {} & [\cdot] \mid \mathscr{C}[\cdot] \mid Q \mid P \mid \mathscr{C}[\cdot] \mid (\nu\ y)\ \mathscr{C}[\cdot] \mid x\langle y\rangle.\mathscr{C}[\cdot] \mid x(y).\mathscr{C}[\cdot] \mid \\ & x.\mathscr{C}[\cdot] \mid x.\mathtt{inl};\mathscr{C}[\cdot] \mid x.\mathtt{inr};\mathscr{C}[\cdot] \mid x.\mathtt{case}(\mathscr{C}[\cdot], Q) \mid x.\mathtt{case}(P, \mathscr{C}[\cdot]) \mid \\ & \mathtt{let}\ x = a\ \mathtt{in}\ \mathscr{C}[\cdot] \mid \mathtt{if}\ v\ \mathtt{then}\ \mathscr{C}[\cdot]\ \mathtt{else}\ Q \mid \mathtt{if}\ v\ \mathtt{then}\ P\ \mathtt{else}\ \mathscr{C}[\cdot]\end{aligned}$$

On top of contexts, it is routine to give a type system deriving judgments in the form

$$\Gamma;\ \Delta;\ \Theta\ \vdash^{\mathcal{V}}\ \mathscr{C}[(\Xi;\ \Psi;\ \Phi\ \vdash^{\mathcal{V}}\ \cdot\ ::\ T)]\ ::\ U$$

guaranteeing that whenever a process P is such that $\Xi;\ \Psi;\ \Phi\ \vdash^{\mathcal{V}}\ P\ ::\ T$, it holds that $\Gamma;\ \Delta;\ \Theta\ \vdash^{\mathcal{V}}\ \mathscr{C}[P]\ ::\ U$.

Talking about observations, what we are interested in observing here is the *probability* of certain very simple events. For example, in Section 2 we hint at the fact that the experiment PrivK^{eav} can be naturally modelled as a process offering a channel carrying just a boolean value. This will very much inform our definition. Suppose that $\vdash^{\mathcal{V}}\ P\ ::\ x : \mathbb{B}$. After having instantiated P through a substitution $\rho : \mathcal{V} \to \mathbb{N}$, some internal reduction steps turn $P\rho$ into a (unique) distribution $\mathscr{D}$ of irreducible processes, and the only thing that can happen at that time is that a boolean value b is produced in output along the channel x. We write $P\ \downarrow_x^\rho\ b$ for this probabilistic event, itself well defined thanks to Theorem 9, Corollary 11 (which witness the existence and unicity of $\mathscr{D}$, respectively), and Progress (see [20] for more details).

Finally, we are ready to give the definition of observational equivalence.

▶ **Definition 12** (Observational Equivalence)**.** *Let P and Q two processes such that $\Gamma;\ \Delta;\ \Theta\ \vdash^{\mathcal{V}}\ P, Q\ ::\ T$. We say that P and Q are observationally equivalent iff for every context $\mathscr{C}[\cdot]$ such that $\vdash^{\mathcal{V}}\ \mathscr{C}[(\Gamma;\ \Delta;\ \Theta\ \vdash^{\mathcal{V}}\ \cdot\ ::\ T)]\ ::\ x : \mathbb{B}$, there is a negligible function $\varepsilon : (\mathcal{V} \to \mathbb{N}) \to \mathbb{R}_{[0,1]}$ such that for every ρ it holds that $\left|\Pr[\mathscr{C}[P]\ \downarrow_x^\rho\ v] - \Pr[\mathscr{C}[Q]\ \downarrow_x^\rho\ v]\right| \leq \varepsilon(\rho)$. In that case we write $\Gamma;\ \Delta;\ \Theta\ \vdash^{\mathcal{V}}\ P \cong Q\ ::\ T$, or just $P \cong Q$ if this does not cause any ambiguity.*

Proving pairs of processes to be observationally equivalent is notoriously hard, due to the presence of a universal quantification over all contexts. However, the fact π**DIBLL** enjoys confluence facilitates the task. In particular, we can prove certain pairs of processes to be observationally equivalent, and this will turn out to be very useful in the next section. The first equation we can prove is the commutation of input prefixes and the `let` construct:

$$x(y).\mathtt{let}\ z = t\ \mathtt{in}\ P \cong \mathtt{let}\ z = t\ \mathtt{in}\ x(y).P \tag{3}$$

As innocuous as it seems, this equation is not validated by, e.g., probabilistic variations on bisimilarity, being (essentially) the classic counterexample to the coincidence of the latter and trace equivalence. In fact, Equation 3 indeed holds here, see [20] for the details. An equation scheme which can be easily proved to be sound for observational equivalence is the one induced by so-called *Kleene-equivalence*: if P and Q are such that there exists a distribution $\mathscr{D}$ such that both $P\rho \Rrightarrow \mathscr{D}$ and $Q\rho \Rrightarrow \mathscr{D}$ (for every ρ), then we can safely conclude that $P \cong Q$. Finally, an equation in which the approximate nature of observational equivalence comes into play is the following one:

$$x.[y \leftarrow x] \cong x.\texttt{let } z = \mathit{rand} \texttt{ in let } b = \mathit{eq}(x, z) \texttt{ in if } b \texttt{ then } [y \leftarrow x] \texttt{ else } [y \leftarrow z]$$

The two involved processes, both offering a session on y having type $\mathbb{S}[n]$ by way of a session with the same type on x, behave the same, except on *one* string z chosen at random.

6 A Simple Cryptographic Proof

In this section, we put relational reasoning at work on the simple example we introduced in Section 2. More specifically, we will show that the notion of observational equivalence from Section 5.2 is sufficient to prove Equation 1, where $\sim$ is taken to be the non-contextual version of observational equivalence. We will do that for an encryption scheme Π_g such that *Enc* is based on a pseudorandom generator g, i.e. *Enc* returns on input a message m and a key k the ciphertext $\mathit{xor}(m, g(k))$. When can such a function g be said to be pseudorandom? This happens when the output of g is indistinguishable from a truly random sequence of the same length. This, in turn, can be spelled out as the equation

$$\mathit{OUTPR}_g \cong \mathit{OUTR} \tag{4}$$

where OUTR is a process outputting a random string of polynomial length of a channel *out*, while OUTPR_g is a process outputting a *pseudo*random such string produced according to g.

We now want to prove, given (4), that (1) holds, the latter now taking the following form:

$$\nu \mathit{adv}.(\mathit{PRIVK}_\Pi \mid \mathit{ADV}) \sim \mathit{FAIRFLIP}_{\mathit{exp}}.$$

Following the textbook proof of this result (see, e.g., [35]), we can structure the proof as a construction, out of ADV, of a distinguisher D_{ADV} having type $\mathit{out} : \mathbb{S}[p] \vdash^n D_{\mathit{ADV}} :: \mathit{exp} : \mathbb{B}$ such that the following two equations hold:

$$\nu \mathit{out}.(\mathit{OUTPR}_G \mid D_{\mathit{ADV}}) \sim \nu \mathit{adv}.(\mathit{PRIVK}_\Pi \mid \mathit{ADV}) \tag{5}$$

$$\nu \mathit{out}.(\mathit{OUTR} \mid D_{\mathit{ADV}}) \sim \mathit{FAIRFLIP} \tag{6}$$

Actually, the construction of D_{ADV} is very simple, being it the process

$$\nu \mathit{adv}.(\mathit{PRIVKEYK}_{\mathit{OTP}} \mid \mathit{ADV}),$$

where OTP is the so-called one-time pad encryption scheme, and $\mathit{PRIVKEYK}_{\mathit{OTP}}$ is the process obtained from $\mathit{PRIVK}_{\mathit{OTP}}$ by delegating the computation of the key to a subprocess:

$\mathit{PRIVKEYK}_{\mathit{OTP}}$:

- input m_0 from adv;
- input m_1 from adv;
- let $b = \mathit{flipcoin}()$ in
- input k from out;
- let $c = \mathit{xor}(k, m_b)$ in
- output c to adv;
- input g from adv;
- let $r = \mathit{eq}(g, b)$ in
- output r to exp;

By construction, and using some of the equations we mentioned in Section 5, one can prove that $PRIVK_{\Pi_g} \cong \nu out.(OUTPR_g \mid PRIVKEYK_{OTP})$, from which, by congruence of $\cong$, one derives Equation (5):

$$\begin{aligned}\nu out.(OUTPR_G \mid D_{ADV}) &\equiv \nu out.(OUTPR_G \mid (\nu adv.(PRIVKEYK_\Pi \mid ADV))) \\ &\equiv \nu adv.(\nu out.(OUTPR_G \mid PRIVKEYK_\Pi) \mid ADV) \\ &\sim \nu adv.(PRIVK_\Pi \mid ADV).\end{aligned}$$

Since $PRIVK_{OTP} \cong \nu out.(OUTR \mid PRIVKEYK_{OTP})$, one can similarly derive that

$$\nu out.(OUTR_g \mid D_{ADV}) \sim \nu adv.(PRIVK_{OTP} \mid ADV).$$

It is well known, however, that the OTP encryption scheme is perfectly secure, which yields Equation (6).

7 Conclusion

Contributions. In this paper, we show how the discipline of session types can be useful in modelling and reasoning about cryptographic experiments. The use of sessions, in particular, allows to resolve the intrinsic nondeterminism of process algebras without the need for a scheduler, thus simplifying the definitional apparatus. The keystone to that is a confluence result, from with it follows that the underlying reduction strategy (i.e. the scheduler) does not matter: the distribution of irreducible processes one obtains by reducing a typable process is unique. The other major technical results about the introduced system of session types are a polynomial bound on the time necessary to reduce any typable process, together with a notion of observational equivalence through which it is possible to faithfully capture computational indistinguishability, a key notion in modern cryptography.

Future Work. This work, exploratory in nature, leaves many interesting problems open. Currently, the authors are investigating the applicability of π**DIBLL** to more complex experiments than that considered in Section 6. In particular, the ability to build higher-order sessions enables the modelling of adversaries which have access to an oracle, but also of experiments involving such adversaries. As an example, an *active* adversary $ACTADV$ to an encryption scheme would have type

$$\vdash^n ACTADV :: c :!_q(\mathbb{S}[p] \multimap \mathbb{S}[p]) \multimap \mathbb{S}[p] \otimes \mathbb{S}[p] \otimes (\mathbb{S}[p] \multimap \mathbb{B}).$$

This reflects the availability of an oracle, modelled as a server for the encryption function, which can be accessed only a polynomial amount of times. Being able to capture *all* those adversaries within our calculus seems feasible, but requires extending the grammar of processes with an iterator combinator. On the side of relational reasoning, notions of equivalence are being studied which are sound with respect to observational equivalence, that is, included in it, at the same time being handier and avoiding any universal quantification on contexts. The use of logical relations or bisimulation, already known in π**DILL** [11] can possibly be adapted to π**DIBLL**, but does not allow to faithfully capture linearity, falsifying equations (like (3)) which are crucial in concrete proofs. As a consequence, we are considering forms of trace equivalence and distribution-based bisimilarity [19], since the latter are known to be fully abstract with respect to (linear) observational equivalence, even in presence of effects.

Related Work. We are certainly not the first to propose a formal calculus in which to model cryptographic constructions and proofs according to the computational model. The so-called Universal Composability model (UC in the following), introduced by Canetti more than twenty years ago [12, 13], has been the subject of many investigations aimed at determining if it is possible to either simplify it or to capture it by way of a calculus or process algebra (e.g. [14, 38, 37, 6]). In all the aforementioned works, a tension is evident between the need to be expressive, so as to capture UC proofs, and the need to keep the model simple enough, masking the details of probability and complexity as much as possible. π**DIBLL** is too restrictive to capture UC in its generality, but on the other hand it is very simple and handy. As for the approaches based on process algebras, it is once again worth mentioning the series of works due to Mitchell et al. and based, like ours, on a system of types derived from bounded linear logic [39, 41, 42]. As already mentioned, the main difference with this work is the absence of a system of behavioural types such as session types, which forces the framework to be complex, relying on a further quantification on probabilistic schedulers, which is not needed here. Another very interesting line of work is the one about imperative calculi, like the one on which tools like `EasyCrypt` are based [8, 7]. Recently, there have been attempts at incepting some form of probabilistic behaviour into session types, either by allowing for probabilistic internal choice in multiparty sessions [4], or by enriching the type system itself, by making it quantitative in nature [34]. The system π**DIBLL** is certainly more similar to the former, in that randomization does not affect the type structure but only the process structure. This design choice is motivated by our target applications, namely cryptographic experiments, in which randomization affects *which* strings protocols and adversaries produce, rather than their high-level behaviour. Indeed, our calculus is closer in spirit to some previous work on cryptographic constructions in λ-calculi [43] and logical systems [33], although the process algebraic aspects are absent there. Finally, session types have also been used as an handy tool guaranteeing security properties like information flow or access control (see, e.g., [15, 9]), which are however different from those we are interested at here. The literature on session types and their relations with linear logic is vast, and deeply influenced the way π**DIBLL** is defined. As an example, the way we handle ground types is very inspired by the one adopted by Toninho, Caires and Pfenning [47]. Moreover, indexed modalities in the style of bounded linear logic have been already considered, e.g., by Kokke et al [36] as a way to tame the inherent nondeterminism arising from a more permissive typing rule for parallel composition. Das and Pfenning [24], instead, label ground types with a sequence of arithmetic expressions obtaining a notion of refinement which is similar, although much more expressive, than the one we use here. In all these works, however, the underlying objective is different from ours, which consists in enforcing complexity bounds in the context of cryptographic experiments.

References

1 Martín Abadi, Ricardo Corin, and Cédric Fournet. Computational secrecy by typing for the pi calculus. In *Proc. of APLAS 2006*, volume 4279 of *LNCS*, pages 253–269. Springer, 2006. doi:10.1007/11924661_16.

2 Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proc. of POPL 2001*, pages 104–115. ACM, 2001. doi:10.1145/360204.360213.

3 Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inf. Comput.*, 148(1):1–70, 1999. doi:10.1006/inco.1998.2740.

4 Bogdan Aman and Gabriel Ciobanu. Probabilities in session types. In *Proc. of FROM 2019*, volume 303 of *EPTCS*, pages 92–106, 2019. doi:10.4204/EPTCS.303.7.

5 Michael Backes, Esfandiar Mohammadi, and Tim Ruffing. Computational soundness results for proverif - bridging the gap from trace properties to uniformity. In *Proc. of POST 2014*, volume 8414 of *LNCS*, pages 42–62. Springer, 2014. doi:10.1007/978-3-642-54792-8_3.

6 Manuel Barbosa, Gilles Barthe, Benjamin Grégoire, Adrien Koutsos, and Pierre-Yves Strub. Mechanized proofs of adversarial complexity and application to universal composability. In *Proc. of CCS 2021*, pages 2541–2563. ACM, 2021. doi:10.1145/3460120.3484548.

7 Gilles Barthe, Thomas Espitau, Benjamin Grégoire, Justin Hsu, Léo Stefanesco, and Pierre-Yves Strub. Relational reasoning via probabilistic coupling. In *Proc. of LPAR 2015*, volume 9450 of *LNCS*, pages 387–401. Springer, 2015. doi:10.1007/978-3-662-48899-7_27.

8 Gilles Barthe, Cédric Fournet, Benjamin Grégoire, Pierre-Yves Strub, Nikhil Swamy, and Santiago Zanella Béguelin. Probabilistic relational verification for cryptographic implementations. In *Proc. of POPL 2014*, pages 193–206. ACM, 2014. doi:10.1145/2535838.2535847.

9 Massimo Bartoletti, Ilaria Castellani, Pierre-Malo Deniélou, Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, Jovanka Pantovic, Jorge A. Pérez, Peter Thiemann, Bernardo Toninho, and Hugo Torres Vieira. Combining behavioural types with security analysis. *J. Log. Algebraic Methods Program.*, 84(6):763–780, 2015. doi:10.1016/j.jlamp.2015.09.003.

10 Laura Bocchi, Tzu-Chun Chen, Romain Demangeon, Kohei Honda, and Nobuko Yoshida. Monitoring networks through multiparty session types. *Theor. Comput. Sci.*, 669:33–58, 2017. doi:10.1016/j.tcs.2017.02.009.

11 Luís Caires and Frank Pfenning. Session types as intuitionistic linear propositions. In *Proc. of CONCUR 2010*, pages 222–236. Springer, 2010. doi:10.1007/978-3-642-15375-4_16.

12 Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. of FOCS 2001*, pages 136–145. IEEE, 2001. doi:10.1109/SFCS.2001.959888.

13 Ran Canetti. Universally composable security. *J. ACM*, 67(5):28:1–28:94, 2020. doi:10.1145/3402457.

14 Ran Canetti, Alley Stoughton, and Mayank Varia. Easyuc: Using easycrypt to mechanize proofs of universally composable security. In *Proc. of CSF 2019*, pages 167–183. IEEE, 2019. doi:10.1109/CSF.2019.00019.

15 Sara Capecchi, Ilaria Castellani, Mariangiola Dezani-Ciancaglini, and Tamara Rezk. Session types for access and information flow control. In *Proc. of CONCUR 2010*, volume 6269 of *LNCS*, pages 237–252. Springer, 2010. doi:10.1007/978-3-642-15375-4_17.

16 David Castro-Perez, Raymond Hu, Sung-Shik Jongmans, Nicholas Ng, and Nobuko Yoshida. Distributed programming using role-parametric session types in go: statically-typed endpoint apis for dynamically-instantiated communication structures. *Proc. of POPL*, 3:29:1–29:30, 2019. doi:10.1145/3290342.

17 Hubert Comon-Lundh, Masami Hagiya, Yusuke Kawamoto, and Hideki Sakurada. Computational soundness of indistinguishability properties without computable parsing. In *Proc. of ISPEC 2012*, volume 7232 of *LNCS*, pages 63–79. Springer, 2012. doi:10.1007/978-3-642-29101-2_5.

18 Ugo Dal Lago and Paolo Di Giamberardino. On session types and polynomial time. *Mathematical Structures in Computer Science*, 26(8):1433–1458, 2016. doi:10.1017/S0960129514000632.

19 Ugo Dal Lago and Francesco Gavazzo. Resource transition systems and full abstraction for linear higher-order effectful programs. In *Proc. of FSCD 2021*, volume 195 of *LIPIcs*, pages 23:1–23:19, 2021. doi:10.48550/arXiv.2106.12849.

20 Ugo Dal Lago and Giulia Giusti. On session typing, probabilistic polynomial time, and cryptographic experiments (long version). Available at arXiv:2207.03360, 2022.

21 Ugo Dal Lago and Martin Hofmann. Bounded linear logic, revisited. *Log. Methods Comput. Sci.*, 6(4), 2010. doi:10.1007/978-3-642-02273-9_8.

22 Ugo Dal Lago, Sara Zuppiroli, and Maurizio Gabbrielli. Probabilistic recursion theory and implicit computational complexity. *Sci. Ann. Comput. Sci.*, 24(2):177–216, 2014. doi:10.48550/arXiv.1406.3378.

23 Ornela Dardha, Elena Giachino, and Davide Sangiorgi. Session types revisited. *Inf. Comput.*, 256:253–286, 2017. doi:10.1016/j.ic.2017.06.002.

24 Ankush Das and Frank Pfenning. Session Types with Arithmetic Refinements. In *Proc. of CONCUR 2020*, volume 171, pages 13:1–13:18, 2020. doi:10.4230/LIPIcs.CONCUR.2020.13.

25 Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE*, 29(2):198–208, 1983. doi:10.1109/TIT.1983.1056650.

26 Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987. doi:10.1016/0304-3975(87)90045-4.

27 Jean-Yves Girard, Andre Scedrov, and Philip J Scott. Bounded linear logic: a modular approach to polynomial-time computability. *Theor. Comput. Sci.*, 97(1):1–66, 1992. doi:10.1016/0304-3975(92)90386-T.

28 Oded Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, 2006.

29 Martin Hofmann and Philip J. Scott. Realizability models for bll-like languages. *Theor. Comput. Sci.*, 318(1-2):121–137, 2004. doi:10.1016/j.tcs.2003.10.019.

30 Kohei Honda. Types for dyadic interaction. In *Proc. of CONCUR 1993*, pages 509–523. Springer, 1993. doi:10.1007/3-540-57208-2_35.

31 Hans Hüttel, Ivan Lanese, Vasco T Vasconcelos, Luís Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, António Ravara, Emilio Tuosto, et al. Foundations of session types and behavioural contracts. *ACM Computing Surveys (CSUR)*, 49(1):1–36, 2016. doi:10.1145/2873052.

32 Atsushi Igarashi, Peter Thiemann, Yuya Tsuda, Vasco T. Vasconcelos, and Philip Wadler. Gradual session types. *J. Funct. Program.*, 29:e17, 2019. doi:10.1017/S0956796819000169.

33 Russell Impagliazzo and Bruce M. Kapron. Logics for reasoning about cryptographic constructions. *J. Comput. Syst. Sci.*, 72(2):286–320, 2006. doi:10.1016/j.jcss.2005.06.008.

34 Omar Inverso, Hernán C. Melgratti, Luca Padovani, Catia Trubiani, and Emilio Tuosto. Probabilistic analysis of binary sessions. In *Proc. of CONCUR 2020*, volume 171 of *LIPIcs*, pages 14:1–14:21, 2020. doi:10.4230/LIPIcs.CONCUR.2020.14.

35 Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

36 Wen Kokke, J. Garrett Morris, and Philip Wadler. Towards races in linear logic. In *Proc. of COORDINATION 2019*, volume 11533 of *LNCS*, pages 37–53. Springer, 2019. doi:10.1007/978-3-030-22397-7_3.

37 Ralf Küsters, Max Tuengerthal, and Daniel Rausch. The IITM model: A simple and expressive model for universal composability. *J. Cryptol.*, 33(4):1461–1584, 2020. doi:10.1007/s00145-020-09352-1.

38 Kevin Liao, Matthew A. Hammer, and Andrew Miller. Ilc: a calculus for composable, computational cryptography. In *Proc. of PLDI 2019*, pages 640–654. ACM, 2019. doi:10.1145/3314221.3314607.

39 Patrick Lincoln, John C. Mitchell, Mark Mitchell, and Andre Scedrov. Probabilistic polynomial-time equivalence and security analysis. In *Proc. of FM 1999*, volume 1708 of *LNCS*, pages 776–793. Springer, 1999. doi:10.1007/3-540-48119-2_43.

40 John C. Mitchell, Mark Mitchell, and Andre Scedrov. A linguistic characterization of bounded oracle computation and probabilistic polynomial time. In *Proc. of FOCS 1998*, pages 725–733. IEEE, 1998. doi:10.1109/SFCS.1998.743523.

41 John C. Mitchell, Ajith Ramanathan, Andre Scedrov, and Vanessa Teague. Probabilistic polynominal-time process calculus and security protocol analysis. In *Proc. LICS 2001*, pages 3–5. IEEE, 2001. doi:10.1109/LICS.2001.932477.

42 John C. Mitchell, Ajith Ramanathan, Andre Scedrov, and Vanessa Teague. A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. *Theor. Comput. Sci.*, 353(1-3):118–164, 2006. doi:10.1016/S1571-0661(04)80968-X.

43 David Nowak and Yu Zhang. A calculus for game-based security proofs. In *Proc. of RPROVSEC 2010*, volume 6402 of *LNCS*, pages 35–52. Springer, 2010. doi:10.1007/978-3-642-16280-0_3.

44 Jorge A Pérez, Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logical relations and observational equivalences for session-based concurrency. *Inf. Comput.*, 239:254–302, 2014. doi:10.1016/j.ic.2014.08.001.

45 Paula Severi, Luca Padovani, Emilio Tuosto, and Mariangiola Dezani-Ciancaglini. On sessions and infinite data. In Alberto Lluch-Lafuente and José Proença, editors, *Proc. of COORDINATION 2016*, volume 9686 of *LNCS*, pages 245–261. Springer, 2016. doi:10.48550/arXiv.1610.06362.

46 Jianxiong Shao, Yu Qin, and Dengguo Feng. Computational soundness results for stateful applied π calculus. In *Proc. of POST 2016*, volume 9635 of *LNCS*, pages 254–275. Springer, 2016. doi:10.1007/978-3-662-49635-0_13.

47 Bernardo Toninho, Luís Caires, and Frank Pfenning. Dependent session types via intuitionistic linear type theory. In *Proc. of PPDP 2011*, pages 161–172. ACM, 2011. doi:10.1145/2003476.2003499.

48 Philip Wadler. Propositions as sessions. *J. Funct. Program.*, 24(2-3):384–418, 2014. doi:10.1145/2398856.2364568.