UC, Categorically: Rigorous Diagrammatic Proofs

Pooya Farshim*, Martti Karvonen†‡, Andre Knispel§, Markulf Kohlweiss¶‖, Philip Wadler¶
Input Output, *Switzerland, §Germany, ¶UK
†University College London, UK, ‡University of Bath, UK ‖University of Edinburgh, UK

***Abstract*—Category theory is a mathematical theory of composition, widely used in logic, computing, and physics. Here we apply it to give a theory of secure composition. In particular, we provide a categorical treatment of Canetti's Universal Composability (UC) framework for systems with a static number of parties and sessions, often termed UC for static systems, yielding four benefits.**

First, we present our results graphically yet retain rigor by applying a standard categorical technique known as *string diagrams*. In particular, our formulation of the composition theorem can be graphically verified with a short sequence of diagrams, while remaining translatable to equations and amenable to formal verification.

Second, categories let us generalize so that our results extend beyond interactive Turing machines to other forms of computation, such as quantum computation or domain-specific languages.

Third, categories help us drop some unnecessary restrictions of UC (e.g., our adversary can be a computational network rather than a single Turing machine); we prove equivalence between our variant and the usual UC, showing no expressiveness is lost.

Finally, the categorical perspective leads us to identify and correct some minor technical oversights in the standard formulation of simple UC.

***Index Terms*—Category Theory, Universal Composition, Symmetric Monoidal Category, String Diagram.**

I. Introduction

While highly influential, Universal Composability (UC) [1], [2] has divided, rather than unified, industrial and academic cryptographers, alongside game-based and simulation-based provable-security experts. Beyond UC, researchers in simulation-based composable security are further divided into camps such as Constructive Cryptography [3] and IITM [4]. This fragmentation is common and expected in relatively young scientific disciplines. What can be learned from unification in other areas, such as mathematics?

Category Theory is the mathematics of composition. It serves as a unifying language across mathematics and is widely used in logic, computing, and physics, not least to reveal connections between the three [5]. It is perhaps surprising that despite its unifying successes in mathematics and computer science, this mathematics of composition has not yet been comprehensively applied to secure composition.

This paper expresses a minor variant of UC for static systems [2, Section 2] (a.k.a. simple UC, henceforward simply "UC") and its composition theorem in categorical terms. While this restriction is made for technical simplicity, simple UC is already fairly expressive. For example, it can model multi-party computation with an a priori known set of participants [6] or any protocol using a static number of idealized building blocks. Indeed, EasyUC [7] works essentially in this setting. This restriction also simplifies the treatment of global functionalities [8], though they are not the focus of this work.

Our approach offers several benefits:

First, categories let us use *string diagrams*, an intuitive yet rigorous diagrammatic reasoning method developed by category theorists. Consequently, our proofs of the Composition Theorems III.11 and III.12 are easier to follow than textual presentations, yet amenable to equational translation and computer formalization.

Second, categories are highly useful for generalization. To quote Mac Lane [9, IX.6], a co-founder of category theory,

> The point of these observations is not the reduction of the familiar to the unfamiliar [...] but the extension of the familiar to cover many more cases.

Indeed, our proof clearly holds for models of computation beyond UC's networks of interactive Turing machines (ITMs). Specifically, it provides a template for developing UC-like theories that replace ITMs with, e.g., quantum systems [10] or code in a suitable domain-specific language (DSL) [7], [11], [12]. Crucially, one does not need to reprove the composition theorem for these variants. To invoke the general result, one simply verifies that their network model of "interactive computational systems" admits a computational indistinguishability notion satisfying Definition III.3 (which typically relies on a model of network execution).

Third, categories tend to "point you in the right direction":

- There is no need for machines to have names (identities) in the context of UC: instead, it is more convenient to name the connections between machines.
- The environment and adversary are not restricted to a single machine; they can consist of multiple machines, just like protocols. As a result, absorbing a machine into the environment or adversary is immediate, avoiding the extra step of replacing the network with a single equivalent machine. This also sidesteps a restriction Canetti notes for single-machine environments (see Remark IV.2).

Despite these changes, we show in Section IV-D that our version translates directly to UC (and vice versa), letting us deduce [2, Theorem 3] from our results.

Finally, we identify and fix some technical oversights (in Section IV-D) in the setup of the UC composition theorem. These fixes are guided by the categorical principle that sequential composition requires matching types (i.e., the codomain

M. Karvonen was supported by the Engineering and Physical Sciences Research Council fellowship EP/V040944/1 Resources in Computation. M. Kohlweiss was supported by Input Output through their funding of the Edinburgh ZK-Lab.

of one matches the domain of the other), and some of the corrected conditions appear in a recent tutorial by Canetti [13].

We envision further benefits once the program has been carried out in greater detail: a DSL-based variant of UC, formalizing the widespread practice of expressing computational behavior as (pseudo)code; similar analyses of other composability frameworks, guided by a categorical axiomatization that guarantees composition "for free"; and, ultimately, rigorous security-preserving translations between frameworks, akin to functorial versions of [14]. We expand on these in Section V.

Related works

This paper advances an ongoing program of applying category theory to composable security frameworks [15], [16]. However, prior literature left open the task of formally connecting these abstract treatments to established frameworks. Here, we close this gap for the most prominent paradigm, namely UC. Rather than pursuing maximal generality, our objective is to provide an abstract template for deriving "UC-like" theories with significantly greater tractability than the original framework allows. Consequently, we establish our main results via rigorous diagrammatic reasoning, requiring fewer categorical prerequisites. The theory developed here is not an instance of the general theory in [16], nor is that theory an instance of ours. In Appendix A we outline a more abstract account of our theory, bringing it closer to [16] in spirit but still differing technically.

The use of informal diagrams as a heuristic device is common in cryptography (e.g., [17]–[20]). Recasting them as string diagrams promotes them to a rigorous tool without losing their intuitive nature. String diagrams have already been used in quantum cryptography when analyzing particular protocols (see, e.g., [21]–[26]), whereas here we use them in the foundations of composable security. Facilitating and popularizing string diagrams in cryptography is one of our long-term goals.

Recent work [27], [28] relates UC to Robust Compilation (RC), showing that UC emulation corresponds to robust hyperproperty-preserving compilation: roughly, compiled programs retain the security properties of their source counterparts no matter what malicious code they are linked with. While RC accommodates distinct source and target languages, instantiating it to UC requires the two to coincide; we instead generalize UC to arbitrary models of computation from the outset, proving the composition theorems once at that level of generality. Moreover, whereas their diagrams serve as informal illustration, our string diagrams are rigorous proof objects. We conjecture that RC compilers correspond to security-preserving symmetric monoidal functors (cf. [16, Theorem 4.10]).

In Section III we make extensive use of a construction $\mathbf{C}_{\mathbf{bd}}$ of a category with backdoors built from a category $\mathbf{C}$. An essentially similar construction appears in [29] (there phrased as a quotient of the coPara construction), with the adversarial interface read as a leak.

Structure of the paper: Section II briefly introduces categories and string diagrams. Section III presents our main novelties, expressing UC categorically. The main results are Theorem III.11, which guarantees that security is closed under sequential and parallel composition, and Theorem III.12, which gives our categorical version of the UC composition theorem. Section IV instantiates the theory with a model based on networks of ITMs, and translates between this and UC [2, Section 2]. Section V concludes with further questions.

II. A Crash Course on Category Theory

This section introduces sufficient background in Category Theory for our purposes. String diagrams are surveyed in [30], [31]. A working cryptographer might find texts such as [32]–[34] easier to learn from, as they start with some applications in mind and introduce string diagrams concurrently with the material. General references for category theory include [35], [36].

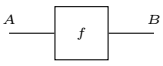
A. Categories

In mathematics and computing we often study objects of a given kind together with maps between those objects, such as:

- sets and functions between sets;
- groups and group homomorphisms;
- finite-dimensional real vector spaces (with a chosen basis) where maps from a space of dimension n to a space of dimension m are given by $m \times n$ matrices;
- types in some type system, where maps from type S to type T consist of a well-typed term t of type T with a free variable x of type S (in shorthand, $x : S \vdash t : T$).

Category theory provides a systematic language to capture this kind of situation. A *category* $\mathbf{C}$ consists of *objects* and *maps* between them. We denote objects $A, B, C, \ldots$ and maps, also called *morphisms*, by $f, g, h, \ldots$. Every morphism has a *source* (also called the *domain*) and a *target* (also called the *codomain*), and we write $f\colon A \to B$ to mean that f is a morphism (in $\mathbf{C}$) from A to B.

A nice feature of categories is that they admit a pictorial syntax, known as *string diagrams*, that is both intuitive and rigorous. We introduce string diagrams in tandem with our discussion of categories. In a string diagram, a morphism $f\colon A \to B$ is depicted by



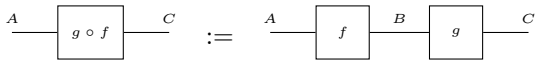
We will often omit the labels for the wires when there is no risk of confusion.

In a category, morphisms can be composed whenever it makes sense to do so. For our four examples above:

- In the category $\mathbf{Set}$ of sets and functions, if $f : A \to B$ and $g : B \to C$, their composition is given by $g \circ f : A \to C$.
- In the category $\mathbf{Grp}$ of groups and group homomorphisms composition is defined similarly.
- In the category $\mathbf{Mat}$ of vector spaces and matrices, composition corresponds to matrix multiplication.

- In the category **Trm** of types and terms, the composition of $x\colon S \vdash t\colon T$ with $y\colon T \vdash s\colon U$ is given by substituting t for y in s, giving $x\colon S \vdash s[t/y]\colon U$.

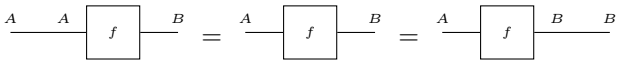
In general, in a category $\mathbf{C}$, there has to be a sequential composition operation $\circ$, which, when given $f\colon A \to B$ and $g\colon B \to C$, returns $g \circ f\colon A \to C$, which in string diagrams is expressed by



To match the diagrammatic order, the notation $f; g$ is sometimes used instead of $g \circ f$.

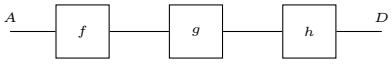
Moreover, for every object A there has to be an identity morphism $\mathrm{id}_A\colon A \to A$ on it, drawn as

What makes id_A and id_B earn their names is that they are required to satisfy $f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f$ for any $f\colon A \to B$. One benefit of string diagrams is how they make the required axioms pictorially intuitive, so this would be expressed by



More informally, this amounts to saying that the length of a wire carries no information.

Moreover, we require composition to be associative, so that whenever we have three morphisms $f\colon A \to B, g\colon B \to C$ and $h\colon C \to D$ that could be composed in sequence, we do not need to specify which pairs we compose first, i.e., $h \circ (g \circ f) = (h \circ g) \circ f$. This means that we can omit the brackets and simply write $h \circ g \circ f$, which justifies the drawing



This concludes the definition of a category: a class of objects and morphisms between them where morphisms compose, there is an identity morphism for each object, the identity morphisms are identities under composition, and composition is associative.

The structure of a category is already sufficiently rich to express many properties of interest. For example, one can define a morphism $f\colon A \to B$ to be an *isomorphism* if there exists a morphism $g\colon B \to A$ such that $g \circ f = \mathrm{id}_A$ and $f \circ g = \mathrm{id}_B$. This general notion specializes to the correct notion of "sameness" in each context, capturing, e.g., bijections between sets and isomorphisms of groups, vector spaces, and types.

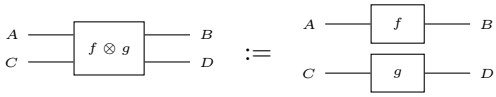
While many examples of interest arise from sets-with-further-structure and structure-preserving maps, the definition of a category does not require the morphisms to consist of some kind of functions. To see that this is not always the right thinking, it is useful to consider a partially ordered set (poset) consisting of a set P equipped with a relation $\leq$ that is reflexive, transitive, and antisymmetric. Any poset $\langle P, \leq \rangle$ induces a category whose objects are given by elements of P, and whose morphisms are defined by setting there to be exactly one morphism $x \to y$ whenever $x \leq y$ and no morphisms $x \to y$ otherwise, with reflexivity and transitivity inducing identities and composition. In fact, we will see another example of a category where the intuition of morphisms as functions is not correct: in Section IV-A we will encounter a category whose morphisms are certain networks of interactive Turing machines.

B. Monoidal and symmetric monoidal categories

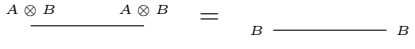
In many categories of interest, there is also a way of composing objects and morphisms *in parallel* and not just sequentially. For example,

- Recall that if A and B are sets then their cartesian product $A \times B = \{\langle x, y\rangle | x \in A, y \in B\}$ is the set of all pairs of elements with the first element in A and the second in B. Given two functions $f\colon A \to B$ and $g\colon C \to D$, there is a function $f \times g\colon A \times C \to B \times D$ which sends $\langle x, y\rangle \in A \times C$ to $\langle f(x), g(y)\rangle \in B \times D$.
- Given two groups G and H, the product $G \times H$ is obtained by taking the cartesian product of the underlying sets, and performing multiplication pointwise. With this definition, the product $f \times g$ of group homomorphisms f and g is again a group homomorphism of product groups.
- Given two vector spaces A and B with bases $a_1, \ldots, a_m$ and $b_1, \ldots, b_n$, their tensor product $A \otimes B$ is an mn-dimensional vector space that has a basis built out of pairs (a_i, b_j). Now, given an $m \times n$-matrix f and a $p \times q$-matrix g, their *Kronecker product* $f \otimes g$ is the $mp \times nq$-matrix essentially obtained by replacing the (i, j)th entry $f_{i,j}$ of f with the matrix $f_{i,j}g$.
- If a type system has product types, one can use terms $x : S \vdash t : T$ and $y : U \vdash v : V$ to obtain a term $z : S \times U \vdash \langle t[\mathsf{fst}(z)/x], v[\mathsf{snd}(z)/y]\rangle : T \times V$, where $\langle t, v\rangle$ builds a pair and fst and snd extract the first and second components of a pair.

Monoidal categories capture this kind of situation. Roughly speaking, a monoidal category is a category equipped with a parallel composition operation that is associative and has a unit. The parallel composition operation is written using the tensor product symbol $\otimes$. Given two morphisms $f\colon A \to B$ and $g\colon C \to D$ in $\mathbf{C}$, their parallel composite is a morphism $f \otimes g\colon A \otimes C \to B \otimes D$. Diagrammatically this is denoted by juxtaposition:



This parallel composition operation must satisfy $\mathrm{id}_{A \otimes B} = \mathrm{id}_A \otimes \mathrm{id}_B$ which pictorially amounts to



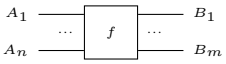
This lets us replace a wire of type $A \otimes B$ with two wires (one for A and one for B) whenever it is convenient to do so. Morphisms between products are not required to be products of morphisms. For example, a function $(x, y) \mapsto$

$(f(x,y), g(x,y))$ is a product only if f depends only on x and g only on y.

An arbitrary box can have multiple input and output wires. Diagrammatically, a morphism

$$f\colon A_1 \otimes \cdots \otimes A_n \to B_1 \otimes \cdots \otimes B_m$$

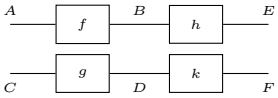
is drawn as



We also require the *interchange law*, which states that

$$(h \otimes k) \circ (f \otimes g) = (h \circ f) \otimes (k \circ g) \qquad \text{(II.1)}$$

for all $f\colon A \to B$, $g\colon C \to D$, $h\colon B \to E$, and $k\colon D \to F$. The interchange law guarantees that the string diagram



is unambiguous.

In a *strict* monoidal category, parallel composition is *unital*. This means that there is a special object I, called *the tensor unit*, satisfying

$$A \otimes I = A = I \otimes A \quad \text{and} \quad \mathrm{id}_I \otimes f = f = f \otimes \mathrm{id}_I$$

on objects and morphisms, respectively. Further, parallel composition is *associative*, meaning

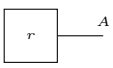
$$A \otimes (B \otimes C) = (A \otimes B) \otimes C \quad \text{and} \quad f \otimes (g \otimes h) = (f \otimes g) \otimes h\ .$$

This results in the notion of a *strict monoidal category*.

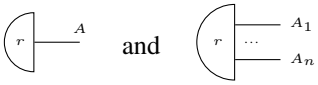
Unfortunately, strict monoidal categories are too strict: they rule out most examples of interest. For instance, the cartesian product of sets is not strictly associative. Rather, it is associative up to the isomorphism $((a,b),c) \mapsto (a,(b,c))$. Loosely speaking, a *monoidal category* is a category with parallel composition $\otimes$ that is associative and unital up to isomorphism. More formally, a monoidal category is a category equipped with a parallel composition operation and associativity and unitality isomorphisms which must satisfy some equations. Technically, these should be natural isomorphisms satisfying the triangle and pentagon identities. We won't delve further into these equations; see [35, Section 7.8] for definitions.

It is a theorem that every monoidal category is monoidally equivalent to a strict monoidal category, which roughly speaking means that for any monoidal category there is a strict monoidal category and suitable mappings between them that translate back and forth. We neither formalize nor prove this theorem here, but use this result implicitly to pretend that all monoidal categories we work with are strict, simplifying notation.

The tensor unit I is denoted by the empty diagram (as juxtaposing any diagram with the empty one results in the same diagram). Thus, a morphism $r\colon I \to A$ would be denoted by a box with no inputs

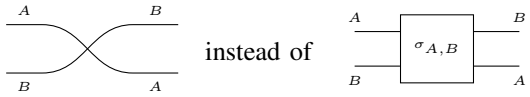


Borrowing terminology from quantum circuits [34], [37], morphisms from I are usually called *states*, and we will draw states $r\colon I \to A$ and $r\colon I \to A_1 \otimes \cdots \otimes A_n$ respectively as

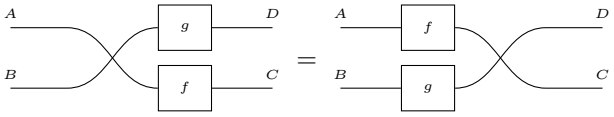


In **Set** states correspond to elements of a set, in **Mat** to vectors, in **Grp** to neutral elements, and in **Trm** to closed terms (i.e., ones with no free variables) of a given type.

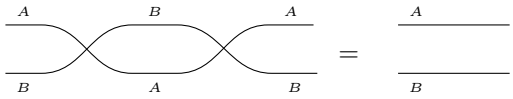
Now, a *symmetric* monoidal category (SMC) is, roughly speaking, a monoidal category where $\otimes$ is suitably commutative. Slightly more formally, an SMC comes with specified *swap* or *symmetry* isomorphisms $\sigma_{A,B}\colon A \otimes B \to B \otimes A$ for all objects A, B, satisfying some further equations. These are more easily explained in string diagrams. To start, we draw $\sigma_{A,B}\colon A \otimes B \to B \otimes A$ as



Given this notation, we then require



or equivalently, $(g \otimes f) \circ \sigma_{A,B} = \sigma_{C,D} \circ (f \otimes g)$ holds for all $f\colon A \to C, g\colon B \to D$. Intuitively, this states that boxes can be slid along wires. (Formally this property corresponds to the family σ being a *natural transformation*, a concept that shows up often in category theory.) Second, we require that crossing the same wires twice amounts to no change:



or equivalently $\sigma_{B,A} \circ \sigma_{A,B} = \mathrm{id}_{A \otimes B}$. The slogan "only connectivity matters" is often used to summarize the pleasant properties of string diagrams for SMCs. While we work somewhat informally with string diagrams, we stress that they are a perfectly rigorous tool to capture exactly what follows from the axioms of an SMC. To this effect, we recall Selinger's formulation [30, Theorem 3.12] of the Joyal–Street coherence theorem [38, Theorem 2.3].

Theorem II.2 (Joyal–Street [38, Theorem 2.3], as formulated in [30, Theorem 3.12])**.** *A well-formed equation between morphisms in the language of symmetric monoidal categories follows from the axioms of symmetric monoidal categories if and only if it holds, up to isomorphism of diagrams, in the graphical language.*

Given an SMC $\mathbf{C}$, by a *sub-SMC* of $\mathbf{C}$ we mean an SMC $\mathbf{D}$ whose objects and morphisms are sub-collections of those of $\mathbf{C}$, and whose SMC-structure (sequential and parallel composition, identity morphisms, unit object, symmetries)

coincides with that of $\mathbf{C}$. Sub-SMCs are closed under arbitrary intersections: as a result, one can always form the smallest sub-SMC containing given collections of morphisms and objects, and we will refer to the end-result as the *sub-SMC generated by* this data.

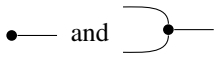
C. Building an SMC from generators and equations

We will now review how to build a symmetric monoidal category from generators and equations, as this will be used in Section IV-A to build a category of networks of interactive Turing machines to model UC. A useful analogy is that of building a group from generators and equations. A more detailed yet introductory account can be found in [31, Chapter 2]. To generate an SMC, one needs

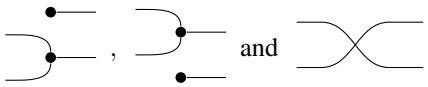
- A set Σ of generating objects, a general object then being an element of the set Σ^* of words in the alphabet Σ.
- For each $V, W \in \Sigma^*$, a (possibly empty) set $\Gamma_{V,W}$ of generating morphisms $V \to W$.

The symmetric monoidal category generated by (Σ, Γ) has Σ^* as its set of objects, and its morphisms can be thought of as given by string diagrams built from identity wires, symmetries and generating morphisms from the sets $\Gamma_{V,W}$ modulo the equations we've asserted for SMCs in the previous section (associativity and unitality of $\otimes$ and the equations for swap).

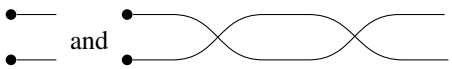
As a warm up for using this construction in later sections, let us consider an example. We let Σ consist of a single element, so that an element of Σ^* is uniquely specified by its length and hence Σ^* can be identified with $\mathbb{N}$. We let $\Gamma_{0,1}$ and $\Gamma_{2,1}$ be singletons and draw their elements as



and let $\Gamma_{i,j}$ be empty for all other choices of $i, j \in \mathbb{N}$. Then in the resulting SMC,



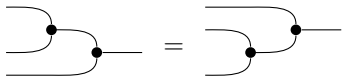
give three distinct morphisms $2 \to 2$ whereas



denote the same morphism $0 \to 2$ due to equations satisfied by any SMC (swapping twice gives the identity).

Having built an SMC from generators, we now discuss building an SMC from generators and equations. It turns out that there is no real need to allow for equations between objects, as equations between morphisms suffice.[1] In turn, a defining equation is given by $f = g$, where f, g are two morphisms $V \to W$ in the monoidal category generated by (Σ, Γ). In the resulting SMC, morphisms can now be thought of as equivalence classes of string diagrams, modulo the defining equations.

Returning to the simple example, we will now add the equations of a commutative monoid. Associativity of the multiplication is captured by



while commutativity is given by

and the unit laws are

One can then build an SMC out of these generators and equations, whose morphisms are equivalence classes of string diagrams modulo the equations we've asserted. In Section IV-A, we will apply this machinery to UC.

III. UC, Categorically

The theory we develop starts from an SMC $\mathbf{C}$, layers some categorical constructions on top, and culminates in Theorems III.11 and III.12 giving our composition theorems. In this section, we leave our starting SMC $\mathbf{C}$ abstract. We will instantiate the general theory with a specific choice of $\mathbf{C}$ in Section IV-A to express UC.

A. Protocols, hybrids, and the adversary

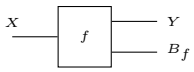
We now assume that the starting category $\mathbf{C}$ is fixed. To help intuition, one may think of the objects of $\mathbf{C}$ as representing communication channels (possibly of some particular type) and of the morphisms of $\mathbf{C}$ as representing networks built by connecting interactive computational systems along channels. In keeping with diagrammatic intuition, we will use the terms "wire," "port," and "interface" interchangeably to denote the endpoints of communication in our networks. Formally they correspond to identity morphisms in $\mathbf{C}$. In contrast, "tape" is reserved for the concrete UC instantiation in Section IV, where it refers to the usual input and subroutine-output tapes of interactive Turing machines. These also correspond to identity morphisms on the generating objects of the SMC constructed in Section IV-A, thus being special types of wires.

We next use $\mathbf{C}$ to construct another SMC $\mathbf{C}_{\mathbf{bd}}$ and then pass to certain sub-SMCs. Ultimately, all of this is only needed to construct our main SMC of interest in Theorem III.11.

As in UC, we will ensure that all protocols have explicit "backdoor" connections for the adversary. This is done by building an SMC $\mathbf{C}_{\mathbf{bd}}$ from $\mathbf{C}$. Objects of this SMC are just those of $\mathbf{C}$. A morphism $X \to Y$ in $\mathbf{C}_{\mathbf{bd}}$ is a "backdoored morphism $X \to Y$", consisting of an object B_f of $\mathbf{C}$ (the type of the backdoor) and a morphism $f : X \to Y \otimes B_f$ in $\mathbf{C}$, under a certain equivalence relation, whose details and motivation we will explain shortly. Intuitively, f represents a network

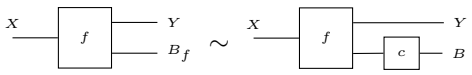
[1] If one did want to identify two words $A, B \in \Sigma^*$, one would instead add generating morphisms $f : A \to B$ and $g : B \to A$ and equations asserting that g is the inverse of f. While this doesn't force A and B to be equal in the resulting SMC, it forces them to be isomorphic, which is sufficient for most categorical purposes.

with honest interfaces given by X and Y and adversarial backdoors given by B_f. We represent such a pair pictorially by the morphism



When the internal structure of a morphism in $\mathbf{C_{bd}}$ is not important to us, we will simply write $f\colon X \to Y$ for a morphism $[B_f, f]\colon X \to Y$.

Pictorially, the equivalence relation is given by setting

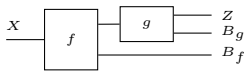


(III.1)

whenever c is an isomorphism.[2] Expressed symbolically, we define $(B_f, f\colon X \to Y \otimes B_f) \sim (B_g, g\colon X \to Y \otimes B_g)$ iff there is an isomorphism $c\colon B_f \to B_g$ such that $g = (\mathrm{id}\otimes c)\circ f$. We will use square brackets when we wish to emphasize that we are working with equivalence classes.

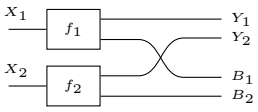
The equivalence relation amounts to saying that it does not matter whether an isomorphism is applied to the adversary's interface. In the case of the concrete UC category built in Section IV, the only isomorphisms are permutations of backdoor tapes, so in this case equivalence amounts to saying that we do not care how the adversary orders its backdoors. Formally, this equivalence relation is needed so that the interchange law of monoidal categories (II.1) holds.

The composite of $[B_f, f]\colon X \to Y$ with $[B_g, g]\colon Y \to Z$ is given pictorially as



and in symbols as $[B_g \otimes B_f, (g \otimes \mathrm{id}_{B_f}) \circ f]$. In words, sequential composition is performed by simply composing along the honest networks while accumulating the backdoors.

This category inherits a symmetric monoidal structure from that of $\mathbf{C}$. The monoidal product on objects is as in $\mathbf{C}$, while the product of morphisms is obtained using both the monoidal product and symmetries in $\mathbf{C}$: the monoidal product of $[B_1, f_1]\colon X_1 \to Y_1$ and $[B_2, f_2]\colon X_2 \to Y_2$ is given by



that is, by composing in parallel and using swaps to collect the backdoors together. In symbols, the monoidal product on morphisms is defined as

$$[B_1, f_1]\otimes[B_2, f_2] := [B_1\otimes B_2, (\mathrm{id}_{Y_1}\otimes\sigma_{B_1,Y_2}\otimes\mathrm{id}_{B_2})\circ(f_1\otimes f_2)]$$

where σ denotes the swap of two wires. We mentioned above that the equivalence relation (III.1) is needed so that the interchange law of monoidal categories (II.1) holds. The reader is invited to compute the two sides of the interchange law and see that they differ by a swap as the adversarial interfaces end up in different orders. This concludes our description of the SMC $\mathbf{C_{bd}}$.

Our theory is defined relative to two nested sub-SMCs $\mathbf{D_{real}} \subseteq \mathbf{D_{bd}}$ of $\mathbf{C_{bd}}$. Passing from $\mathbf{C_{bd}}$ to a sub-SMC $\mathbf{D_{bd}}$ is needed to model some idiosyncrasies of UC. In the case of UC, the main added requirement here is that every machine has exactly one backdoor channel for the adversary. In turn, the choice of a further sub-SMC $\mathbf{D_{real}}$ encodes the model of corruption by specifying the allowed building blocks for real protocols. Formally, this is needed so that the task of constructing some ideal functionality does not become trivial. In the case of UC, $\mathbf{D_{real}}$ will consist of networks of ITMs that are fully corruptible by the adversary, but one could choose differently: if the basic building blocks of $\mathbf{D_{real}}$ are machines which leak all information they have to the adversary without giving any control to the adversary, then one gets a model of honest-but-curious behavior.

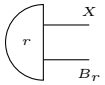
B. Functionalities and security

We now assume that our SMCs $\mathbf{D_{real}} \subseteq \mathbf{D_{bd}}$ have been fixed. We now define what we mean by *resources* in these categories.

Definition III.2. A *resource (a.k.a. state)* r on X is a morphism $r\colon I \to X$ of $\mathbf{D_{bd}}$.

Looking ahead, in our concrete UC category (defined in Section IV) resources will correspond to (hybrid) protocols. In particular, any ideal functionality will be a resource, motivating the terminology.

Formally, resources are equivalence classes of the form $[B_r, r\colon I \to X \otimes B_r]$, however, we will show that our definitions of security do not depend on the particular choice of a representative. Therefore, we will draw a resource as



Before giving our security definition, we need one further ingredient, namely, an abstract notion of indistinguishability. In the case of UC, this will coincide with the usual notion of computational indistinguishability (defined formally in section IV-C). For now, all we need to know about computational indistinguishability is that it gives, for each object X of $\mathbf{C}$, an equivalence relation $\approx$ on states $I \to X$ in $\mathbf{C}$, and that these equivalence relations behave well with respect to sequential and parallel composition in the sense defined below.

Definition III.3. Given an equivalence relation $\approx_X$ on states $I \to X$ in $\mathbf{C}$ for every object X, we say that $\approx := (\approx_X)_{X\in\mathbf{C}}$ *respects (sequential and parallel) composition* if

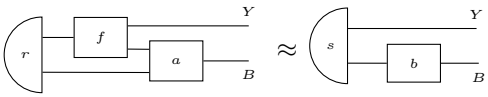
1) If $r \approx_X s$ for $r, s\colon I \to X$ in $\mathbf{C}$ and $f\colon X \to Y$ is a morphism in $\mathbf{C}$, then $f \circ r \approx_Y f \circ s$.
2) If $r_i, s_i\colon I \to X_i$ for $i = 1, 2$ are states in $\mathbf{C}$ such that $r_i \approx_{X_i} s_i$, then $r_1 \otimes r_2 \approx_{X_1\otimes X_2} s_1 \otimes s_2$.

[2] We could equally well restrict to only isomorphisms generated by swaps and identities.

From now on, we omit the subscripts from $\approx$ as they can be inferred from context.

We now give our security definition, which follows the simulation paradigm by quantifying over all possible actions of the adversary.

Definition III.4. Let $r\colon I \to X$ and $s\colon I \to Y$ be resources. A morphism $f\colon X \to Y$ of $\mathbf{D_{real}}$ is a *secure emulation* $r \to s$ if for every object B and every morphism $a\colon B_f \otimes B_r \to B$ (corresponding to the adversary) in $\mathbf{C}$ there exists a $b\colon B_s \to B$ in $\mathbf{C}$ (corresponding to a simulator) such that



(III.5)

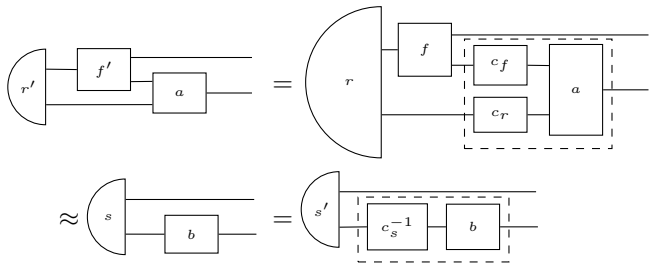
i.e., $(\mathrm{id}_Y \otimes a) \circ (f \otimes \mathrm{id}_{B_r}) \circ r \approx (\mathrm{id}_Y \otimes b) \circ s$ in $\mathbf{C}$.

Note that this is more or less the picture people would usually draw when defining simulation-based security, except that (i) it is now a *rigorous* string diagram and (ii) the equivalence relation $\approx$ implicitly quantifies over all environments, and as a result the environment is not drawn (see Lemma IV.1 for a formal justification of this in the case of UC). Our completeness of the dummy adversary Theorem III.7 will allow us to also omit the universally quantified a from the picture.

A priori, the definition of security depends on the choice of a representative. We now observe that this is not the case. The idea in the proof is simple: equivalence classes only differ by applying an isomorphism on the adversarial port, and using this we can show that one choice of representatives satisfies security if any other does.

Proposition III.6. *Security is well-defined.*

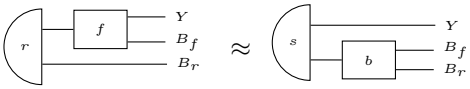
Proof. Assume r, s and f satisfy Definition III.4. Consider arbitrary r', s', f' with $r \sim r'$, $s \sim s'$ and $f \sim f'$ and fix isomorphisms $c_r\colon B_r \to B_{r'}$, $c_s\colon B_s \to B_{s'}$ and $c_f\colon B_f \to B_{f'}$ witnessing this. To show that r', s' and f' also satisfy Definition III.4, consider an arbitrary adversary $a\colon B_{f'} \otimes B_{r'} \to B$ in $\mathbf{C}$, and compute as follows:



where the first equation follows from the defining properties of c_f and c_r, the second step follows by security of $f\colon r \to s$, and the final equation follows from properties of c_s. □

To check security, there is a convenient special case which is sufficient.

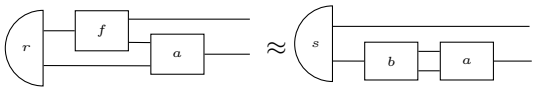
Theorem III.7 (Completeness of the dummy adversary)**.** *Let $r\colon I \to X$ and $s\colon I \to Y$ be resources. If a morphism $f\colon X \to Y$ in $\mathbf{D_{real}}$ satisfies*



(III.8)

i.e., $(f \otimes \mathrm{id}_{B_r}) \circ r \approx (\mathrm{id}_Y \otimes b) \circ s$ for some b, then f is a secure emulation $r \to s$.

Proof. Let us assume (III.8). Then, as $\approx$ respects composition,

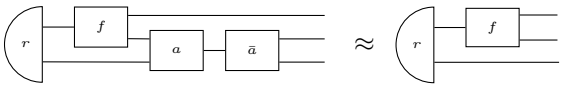


holds for all attackers $a\colon B_f \otimes B_r \to B$ in $\mathbf{C}$, giving us (III.5). Symbolically, this derivation becomes

$$\begin{aligned}(\mathrm{id}_Y \otimes a) \circ (f \otimes \mathrm{id}_{B_r}) \circ r &\approx (\mathrm{id}_Y \otimes a) \circ (\mathrm{id}_Y \otimes b) \circ s \\ &= (\mathrm{id}_Y \otimes (a \circ b)) \circ s. \quad \square\end{aligned}$$

This result is an analogue of the "completeness of the dummy adversary" in UC. However, the identity maps are not literally the dummy adversary in UC, as the latter has to multiplex several channels into the only channel it exposes to the environment. We prove a result capturing completeness for a wide class of adversaries, which will be needed later for our translation Theorem IV.6 into UC.

Corollary III.9. *Let $r\colon I \to X$ and $s\colon I \to Y$ be resources. Assume $f\colon X \to Y$ in $\mathbf{D_{real}}$ satisfies the security equation (III.5) for some fixed a and b in $\mathbf{C}$. Assume moreover that a is* split mono *on f applied to r in the sense that there is a morphism $\bar{a}$ in $\mathbf{C}$ such that*



(III.10)

i.e., $(\mathrm{id}_Y \otimes (\bar{a} \circ a)) \circ (f \otimes \mathrm{id}_{B_r}) \circ r \approx (f \otimes \mathrm{id}_{B_r}) \circ r$. Then f is a secure emulation $r \to s$.

Proof. Attach $\bar{a}$ to bottom wires on both sides of (III.5): then equation (III.10) implies (III.8) which is sufficient by Theorem III.7. □

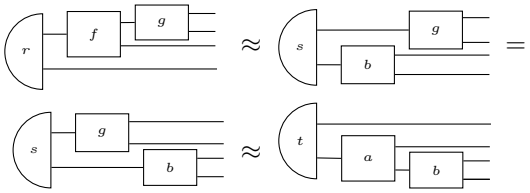
In the context of UC, the morphism a above will be the actual UC dummy adversary.

C. Categorical composition theorems

For us, the core of the composition theorems is that *secure emulation maps form an SMC* as this captures that secure maps are closed under sequential and parallel composition, *and* also that these compositions are suitably well-behaved. In our opinion, this is what Abstract Cryptography gestures at when talking about "composition-order invariance" [39, Definition 13]. We now give our composition theorem.

Theorem III.11 (SMC structure on secure emulations)**.** *Resources and secure emulations between them form a symmetric monoidal category.*

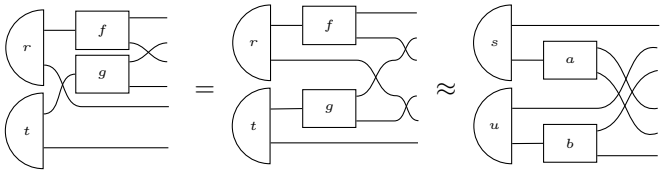
Proof. We repeatedly use Theorem III.7. We first observe that secure emulations are closed under sequential composition. Assuming $f\colon r \to s$ and $g\colon s \to t$ are secure, so is $g \circ f\colon r \to t$:



where the first (third) step follows from security of f (of g) and $\approx$ respecting composition, and the second equation follows from the interchange law. Equationally,

$$
\begin{aligned}
&(g \otimes \mathrm{id}_{B_f \otimes B_r}) \circ (f \otimes \mathrm{id}_{B_r}) \circ r \\
&\approx (g \otimes \mathrm{id}_{B_f \otimes B_r}) \circ (\mathrm{id}_Y \otimes b) \circ s && \text{(security of } f\text{)} \\
&= (g \otimes b) \circ s && \text{(interchange)} \\
&= (\mathrm{id}_{Z \otimes B_g} \otimes b) \circ (g \otimes \mathrm{id}_{B_s}) \circ s && \text{(interchange)} \\
&\approx (\mathrm{id}_{Z \otimes B_g} \otimes b) \circ (\mathrm{id}_Z \otimes a) \circ t && \text{(security of } g\text{)} \\
&= \big(\mathrm{id}_Z \otimes ((\mathrm{id}_{B_g} \otimes b) \circ a)\big) \circ t. && \text{(interchange)}
\end{aligned}
$$

To show that secure maps are closed under parallel composition, we let $f\colon r \to s$ and $g\colon t \to u$ be secure and compute as follows:



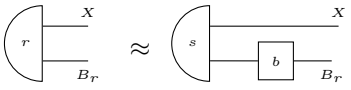
The first equation is true in any monoidal category, and the second equation follows from security of f and g and from $\approx$ respecting composition.

It is easy to show that swaps, and in fact all isomorphisms, are secure. Moreover, as our SMC structure is inherited from the SMC $\mathbf{C_{bd}}$, the further required properties of an SMC automatically hold. □

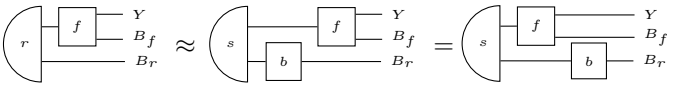
We now move on to a categorical counterpart of the composition theorem of UC (for static systems) [2, Theorem 3].

Theorem III.12 (Categorical universal composition)**.** *Let $r, s\colon I \to X$ be resources. Assume that the identity on X gives a secure emulation $r \to s$. Then any $f\colon X \to Y$ in $\mathbf{D_{real}}$ gives a secure emulation from r to $f \circ s$.*

Proof. By assumption there exists a simulator b such that



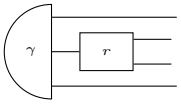
It then follows that



where the first equation follows from the previous equation and indistinguishability respecting composition, and the second from the axioms of an SMC. We have now proved the result for the dummy adversary, which is sufficient by Theorem III.7. □

D. Global subroutines

We now observe that working with shared "global" resources (e.g., a shared clock or a shared PKI) already falls under the theory above. Indeed, a "resource" r potentially calling (and therefore connected to) a global resource $\gamma\colon I \to Y \otimes X$ would be modelled as a morphism $r\colon X \to W$, and the resulting total resource $(\mathrm{id} \otimes r) \circ \gamma$ would be depicted as

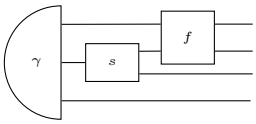


where the top wire represents the (in general composite) remaining interface Y of γ, the one below it is the honest interface of r, and the bottom two wires give the interfaces of the adversary. We can then define secure emulations $r \to s$ in the presence of a global resource γ as ordinary secure emulations from r with γ to s with γ.

Definition III.13. A secure emulation $r \to s$ *with a global resource* γ is a secure emulation $(\mathrm{id} \otimes r) \circ \gamma \to (\mathrm{id} \otimes s) \circ \gamma$.

We now observe that the Universal Composition with Global Subroutines (UCGS) Theorem [40] for static systems can be captured as a special case of Theorem III.12, following [8].

Theorem III.14 (Categorical universal composition with global subroutines)**.** *Assume that the identity map gives a secure emulation from $r\colon X \to W$ with global resource γ to $s\colon X \to W$ with γ. Then any $f\colon Y \otimes W \to W'$ in $\mathbf{D_{real}}$ gives a secure emulation from $(\mathrm{id} \otimes r) \circ \gamma$ (i.e., from r with γ) to*



i.e., to the resource $f \circ (\mathrm{id} \otimes s) \circ \gamma$.

In Section IV-D we discuss how the above theorem can be used to derive the UCGS theorem of [8] as a corollary.

IV. UC, Concretely

One could instantiate the theory developed in the previous section with any of the example SMCs discussed so far. However, these are unlikely to yield cryptographically meaningful theories. Instead, one can extract SMCs fairly directly from (minor variants of) state-separating proofs [19], random systems [41], and perhaps with more effort from elsewhere in

cryptography. In general, finding cryptographically meaningful categories off-the-shelf is difficult (cf. [16, Section 9]).

As a result, one might want to build this SMC directly. This is particularly likely to be useful if one has some notion of an atomic "interactive computational system," which when composed results in *a network* of such systems (e.g., two ITMs can be connected to form a network of two ITMs, rather than a single ITM)[3] instead of, say, how two functions compose into a single function.

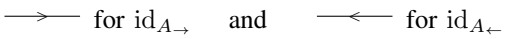
We can conveniently build such categories by means of generators and equations as described in Section II-C. When describing the resulting category, it is useful to borrow intuitions from UC. In this viewpoint, the generating objects Σ correspond to basic types of communication channels. As a first approximation, UC has just one basic type, but one could imagine communication channels parametrized by the type of data flowing on them (so that a channel for bit strings is different from a channel for communicating vectors, graphs, or elements of a group), or perhaps each channel could have a session type [42], capturing the expected communication pattern ("first I send you a vertex of a graph, then you respond with an integer, giving its color" etc.). Given two words V, W, one then thinks of the generating set $\Gamma_{V,W}$ as the set of all atomic/basic computational systems with channels V on one side of the system and channels W on the other. For example, in our formalization of UC these will correspond to interactive Turing machines with a given number of input tapes and a given number of subroutine-output tapes. One may also add further generators and equations if need be. We note that the backdoor tape is not modeled in this base category and will be added later on.

A. The starting category for UC

We will now describe how to build, via generators and relations, an SMC whose morphisms are networks of interactive Turing machines (ITMs). We will fix details of our machines as needed; for now, it is sufficient to assume that in addition to some internal tapes each given ITM has some number n of tapes for receiving inputs from other ITMs (from now on, input tapes), and some number m of tapes for receiving subroutine-outputs from other ITMs (from now on, subroutine-output tapes or just output tapes). We assume that these tapes are enumerated, so that one can always speak without ambiguity of the ith input tape (or the jth subroutine-output tape). Connections between ITMs are then obtained by pairing the ith input tape of one machine with the jth subroutine-output tape of another, letting each machine send messages that are received by the other machine at the corresponding tape. (No tape gets paired twice.) Thus, as in UC, the tapes are only used for receiving messages. However, we differ from UC in that we allow multiple subroutine-output and input tapes, instead of one each. We will return to this in Section IV-D.

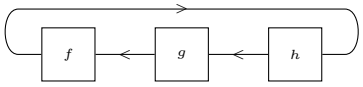
Formally, this pairing of tapes takes place in the SMC that we will build, with its operational interpretation given in Section IV-C. However, the picture to keep in mind is that when such a pairing is done, one machine can receive inputs from the other machine on its ith input tape, and can send outputs to the other machine's jth subroutine-output tape.

As a first approximation, UC has only one type of wire, representing an untyped communication channel. However, the default kind of communication between ITMs in UC is never between equals: given two communicating machines, one is always designated to be the "caller" and the other the "subroutine." While formally UC does not enforce a difference between these two roles, they are often used in practice. We will thus set $\Sigma := \{A_{\rightarrow}, A_{\leftarrow}\}$, with $A_{\rightarrow}$ representing a connection with the caller on the left and $A_{\leftarrow}$ representing a connection with the caller on the right. Instead of labeling the diagrams with $A_{\rightarrow}$ and $A_{\leftarrow}$ for these two objects, we will simply draw corresponding identity morphisms by an arrow pointing away from the caller:



We define $\Gamma_{A_{\leftarrow}^m, A_{\leftarrow}^n}$ to consist of the set of ITMs with m subroutine-output tapes and n input tapes. We will explain shortly why we do not include words containing $A_{\rightarrow}$.

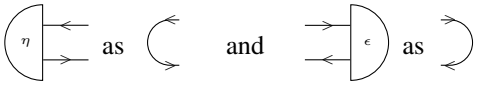
If these were all the generators we had, we could only build DAGs of ITMs whereas UC imposes no restrictions on the shape of connectivity. For example, one could have a cycle of calls, consisting of three machines f, g, h, with h calling g, g calling f, and f calling h, resulting in the diagram



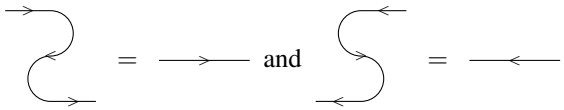
To make sense of this, we need the ability to achieve such connectivity. Formally, we add two additional generating morphisms by setting $\Gamma_{I, A_{\leftarrow} A_{\rightarrow}} = \{\eta\}$ and $\Gamma_{A_{\rightarrow} A_{\leftarrow}, I} = \{\epsilon\}$ and the defining equations

$$(\epsilon \otimes \mathrm{id}_{A_{\rightarrow}}) \circ (\mathrm{id}_{A_{\rightarrow}} \otimes \eta) = \mathrm{id}_{A_{\rightarrow}}$$
$$(\mathrm{id}_{A_{\leftarrow}} \otimes \epsilon) \circ (\eta \otimes \mathrm{id}_{A_{\leftarrow}}) = \mathrm{id}_{A_{\leftarrow}} \ .$$

We first indulge in some syntactic sugar by denoting



The two defining equations above then correspond to the following *snake equations*:



We set the remaining generating sets $\Gamma_{V,W}$ to be empty for all other words V, W. This results in our base SMC $\mathbf{C}$. These *cups* and *caps* (η and ϵ) are added to remain faithful to simple UC, where connecting machines in a cycle is allowed. Moreover, the cups and caps let us build morphisms $V \to W$ even when the generating set $\Gamma_{V,W}$ is empty. However, the ability to

[3] Whether or not the composed network can be simulated by a single ITM is beside the point here.

connect machines cyclically is rarely used in cryptographic practice, as DAGs of ITMs are usually sufficient.

B. Protocols, hybrids, and the adversary in UC

Our starting category $\mathbf{C}$ yields the associated SMC $\mathbf{C_{bd}}$ as in Section III-A. However, we now need to cut down from $\mathbf{C_{bd}}$ to certain sub-SMCs in order to model UC faithfully. We will introduce nested sub-SMCs $\mathbf{D_{bd}} \supseteq \mathbf{D_{real}}$ of $\mathbf{C_{bd}}$.

In each of these the objects are restricted words in the generating object $A_{\leftarrow}$ of $\mathbf{C}$ (instead of words in $\{A_{\rightarrow}, A_{\leftarrow}\}$) and morphisms $[B_f, f]$ will have the same restriction on the object B_f: intuitively, this is to ensure that states in the resulting category are given by networks of ITMs that are subroutines of the environment, ruling out states that also have machines calling the environment. In other words, every object is of the form $A^n_{\leftarrow}$ for some $n \in \mathbb{N}$, and every state on $A^n_{\leftarrow}$ will be an equivalence class of the form $[A^m_{\leftarrow}, r\colon I \to A^n_{\leftarrow} \otimes A^m_{\leftarrow}]$. The interpretation is that m is the number of backdoors (which will coincide with the number of machines in r) and n the number of outgoing (honest) connections. As the other generating object is not referenced from now on, we will simply write A instead of $A_{\leftarrow}$ and draw wires without annotating them by arrows. Moreover, we will impose the following additional restrictions on all morphisms $[B_f, f]$.

- For $\mathbf{D_{bd}}$, we want to ensure that every machine that is part of a network has exactly one tape for the adversary. This is achieved by first setting $\mathbf{D'_{bd}}$ to be the sub-SMC of $\mathbf{C_{bd}}$ generated by (i.e., the smallest sub-SMC containing) cups and caps and all morphisms of the form $[A, f\colon A^m \to A^n \otimes A]$, where f consists of a single ITM with m subroutine-output tapes and $n+1$ input tapes. This implements the restriction to one backdoor tape per machine. However, we also want to ensure that all backdoors and interfaces are oriented similarly, so that resources in the resulting category only expect inputs from the environment (and the adversary) instead of calling them. This is achieved by defining $\mathbf{D_{bd}}$ to be the sub-SMC of $\mathbf{D'_{bd}}$ containing all objects of the form A^n and all morphisms $[A^m, f\colon A^n \to A^k \otimes A^m]$.
- For $\mathbf{D_{real}}$ we additionally require that all of the allowed ITMs behave in some fixed prescribed way (e.g., giving full control and all of its history to the adversary) when instructed by the adversary. For example, in the case of full corruptions, this is achieved by first setting $\mathbf{D'_{real}}$ to be the sub-SMC of $\mathbf{C_{bd}}$ generated by cups and caps and all morphisms of the form $[A, f\colon A^m \to A^n \otimes A]$, where f consists of a single ITM with m subroutine-output tapes and $n+1$ input tapes and *moreover*, when instructed to do so at the $(n+1)$st tape, f gives full control and leaks its current state to the machine it is connected to. We then obtain $\mathbf{D_{real}}$ by defining it to be the sub-SMC of $\mathbf{D'_{real}}$ consisting of all objects of the form A^n and all morphisms $[A^m, f\colon A^n \to A^k \otimes A^m]$ of $\mathbf{D'_{real}}$.

Finer details on corruptions can be layered as additional conventions as in UC. For instance, to ensure that corruptions are (suitably) commensurate between the real and ideal, we might require that the environment learns some fixed function of the set of corrupted identities (e.g., which parties are corrupted).

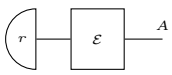
C. The execution model and indistinguishability

We now fill in the details of computational indistinguishability. First, we spell out some further details of ITMs.

- First of all, we allow our ITMs to be probabilistic, which can be modeled by each of them having access to an internal randomness tape, containing an infinite sequence of uniformly random bits.
- Machines are efficient in a suitable sense. Generally speaking, choosing a good notion of efficiency is surprisingly subtle [43]. For us, what matters is that there is a notion of efficiency for machines such that any network of efficient machines can be replaced by an efficient machine emulating the entire network. This justifies allowing the adversary to consist of multiple machines. This property is only required for comparison with UC. For Theorem IV.6 to hold, our notion of efficiency needs to also agree with that of UC (for static systems), which in turn also relies on the above.

We now define, for each object X of $\mathbf{C}$, the equivalence relation of computational indistinguishability on states $I \to X$ of $\mathbf{C}$. First, we define *an environment* of type X to be a morphism $\mathcal{E}\colon X \to A$ in $\mathbf{C}$, i.e., a network of ITMs with one input tape on the right and subroutine-output tapes given by X on the left with some further constraints explained below.

Now, given a state $r\colon I \to X$ in $\mathbf{C}$, we feed this state into the environment, resulting in $\mathcal{E} \circ r$:



We will now explain how this is to be executed when $\mathcal{E}$ is given an initial input z, following the single-threaded execution model of UC.

Initially, the environment $\mathcal{E}$ receives the input z (whose length is interpreted as the security parameter) on the machine to which the only input tape of $\mathcal{E}$ belongs, the *main machine of* $\mathcal{E}$, which is then activated. After that, the machines take turns being executed as follows: the currently active machine is executed until it either sends a message to another machine, after which the message is written on the corresponding tape of the recipient machine (i.e., inputs get written on the input tape and outputs get written on the output tape) which now becomes active, or the machine halts without sending a message. If the machine that halted was not the main machine of $\mathcal{E}$, we activate the main machine of $\mathcal{E}$ again. If it was the main machine of $\mathcal{E}$, the whole execution terminates. We also assume that the main machine of $\mathcal{E}$ never halts without sending a message, whether as input to one of the machines to its left or as a decision bit b written to subroutine-output on its right. Thus the computation goes back and forth between $\mathcal{E}$ and r until $\mathcal{E}$ concludes by returning a bit $b \in \{0,1\}$.

For any state r, input z and $\mathcal{E}$, we thus obtain a probability distribution $\mathsf{EXEC}_{r,\mathcal{E}}(z)$ on $\{0,1\}$, where the randomness comes from internal randomness used by $\mathcal{E}$ and r. We thus obtain a probability ensemble $\{\mathsf{EXEC}_{r,\mathcal{E}}(z)\}_{z\in\{0,1\}^*}$, which

we denote by $\mathsf{EXEC}_{r,\mathcal{E}}$. We say that $r, s\colon I \to X$ in $\mathbf{C}$ are *computationally indistinguishable*, denoted $r \approx s$, if for any environment $\mathcal{E}$ of type X, the (statistical) difference between the ensembles $\mathsf{EXEC}_{r,\mathcal{E}}$ and $\mathsf{EXEC}_{s,\mathcal{E}}$ (i.e., its distinguishing advantage) is negligible in $|z|$. Note that when we say r and s are indistinguishable this in particular means that r and s have the same codomain (and thus have the same type). Note also that in our notation the adversary (resp., simulator) is absorbed into r (resp., s).

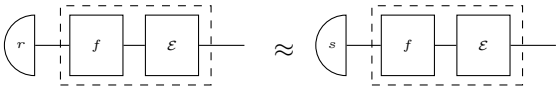
We next observe that computational indistinguishability respects composition i.e., satisfies Definition III.3. This lemma performs, once and for all, the absorption of parts of the network into the environment, which in particular justifies the practice of not drawing the environment in diagrams.

Lemma IV.1. *Computational indistinguishability respects (sequential and parallel) composition.*

Proof. Let us overload notation as follows: given $r, s\colon I \to X$ and an environment $\mathcal{E}$ of the correct type, we will write

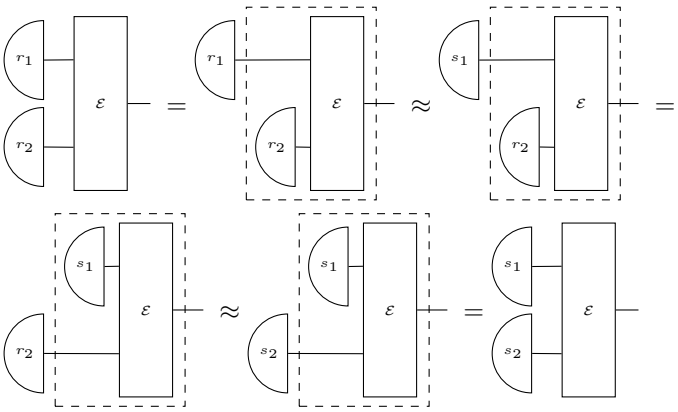
to mean that the (statistical) difference between the ensembles $\mathsf{EXEC}_{r,\mathcal{E}}$ and $\mathsf{EXEC}_{s,\mathcal{E}}$ is negligible. With this notation, the proof is just a matter of absorbing parts into the environment, indicated by dashed lines.

Let $r, s\colon I \to X$ be states in $\mathbf{C}$, and let $f\colon X \to Y$ be a morphism in $\mathbf{C}$. Let us assume $r \approx s$. To show that $f \circ r \approx f \circ s$, let $\mathcal{E}$ be an arbitrary environment and compute as follows:



where the (implicit) step of adding a dashed area corresponds to $\mathcal{E} \circ f$ being a valid environment and follows from associativity of composition, and the displayed step follows from the assumption that $r \approx s$.

For parallel composition, let $r_i, s_i\colon I \to X_i$ be states in $\mathbf{C}$ for $i = 1, 2$ such that $r_i \approx s_i$. Let $\mathcal{E}$ be arbitrary. We then reason as follows:



The first, third and fifth steps follow from axioms of an SMC, the second by $r_1 \approx s_1$ the fourth from $r_2 \approx s_2$. It follows that $r_1 \otimes r_2 \approx s_1 \otimes s_2$. □

Remark IV.2. We now observe that nothing so far has formally required that networks of machines can be simulated by a single machine from the same class, and in particular Theorem III.12 goes through without such an assumption. This contrasts with a point made by Canetti [2], who notes that "the UC theorem is, in general, false in settings where systems of ITMs cannot be simulated on a single ITM from the same class." In our view, this is an artifact of requiring environments to be single machines: the composition theorem holds without assuming that a network can be reduced to a single ITM of the same class, provided one allows environments to be networks of machines as they are then automatically closed under composition.

We conclude this section with a technical result in $\mathbf{C}$, used to relate adversaries in the categorical setting to those of UC.

For each m, let $\mathsf{mux}_m\colon A^m \to A$ be the ITM that relays each message arriving on its ith subroutine output tape (on the left) as a subroutine output to the machine it is connected to on the right, tagged with i, and relays each tagged message (i, msg) it receives on its single input tape back along the ith wire (behaving in some fixed, efficient but otherwise arbitrary way on messages that are not appropriately tagged); let $\mathsf{demux}_m\colon A \to A^m$ be the same relay with the roles of the inputs and subroutine-output exchanged.

Lemma IV.3 (Completeness of the multiplexing adversary)**.** *For every m and every state $r\colon I \to Y \otimes A^m$ in $\mathbf{C}$,*

$$\big(\mathrm{id}_Y \otimes (\mathsf{demux}_m \circ \mathsf{mux}_m)\big) \circ r \;\approx\; r.$$

Consequently, for resources r, s and a morphism $f\colon r \to s$ of $\mathbf{D}_{\mathbf{real}}$ whose combined backdoor is A^m, the adversary $\mathsf{mux}_m\colon A^m \to A$ is split mono on f applied to r.

Proof. The round-trip $\mathsf{demux}_m \circ \mathsf{mux}_m$ sends every message out and back under a faithful tagging, hence reproduces it exactly. As a result, it is indistinguishable from id_{A^m} on any state. This is exactly the hypothesis (III.10) of Corollary III.9 with $a = \mathsf{mux}_m$ and $\bar{a} = \mathsf{demux}_m$. □

In particular, security may always be checked against an adversary whose interface to the environment is the single wire A by Corollary III.9.

D. Relationship to and differences with UC

Now that we have the subcategories $\mathbf{D}_{\mathbf{real}} \subseteq \mathbf{D}_{\mathbf{bd}}$ and a composition-respecting equivalence relation, the theory in Section III readily applies. In particular, Theorems III.11 and III.12 give rise to composition theorems for this specific model. We will now discuss how this model relates to UC.

Our setting differs in some technical details from that of UC, despite being in essence the same theory. First of all, for us, an ITM can have multiple communication tapes. In contrast, in UC every machine has only one input tape and one subroutine-output tape (and one backdoor tape).

This, in turn, gives rise to a second difference as to how the connectivity of a network of interactive Turing machines is captured. In our setting, we explicitly add data, keeping track

of which tapes are paired together. In contrast, in UC each machine is modeled as a triple $\mu = (\mathsf{ID}_\mu, \mathsf{C}_\mu, \tilde{\mu})$, where ID_μ is the *identity of a machine*, i.e., a string giving the machine its name, C_μ is its *communication set*, i.e., a set of pairs of the form $(\mathsf{ID}, \mathsf{tp})$, where ID is the name of a machine and $\mathsf{tp} \in \{\mathsf{input}, \mathsf{subroutine\text{-}output}, \mathsf{backdoor}\}$, and finally $\tilde{\mu}$ is the actual program (code) of the machine.

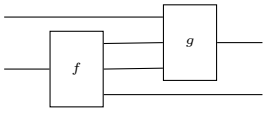
For a set π of such machines to form *a protocol*, the following conditions must be satisfied.

- The identities of the machines in π are distinct from each other and from $\{0, 1\}$ (as 0 and 1 are reserved for the environment and the adversary, respectively).
- Backdoor communication is not used,[4] i.e., every entry $(\mathsf{ID}, \mathsf{tp})$ appearing in the communication set of some machine in π has $\mathsf{tp} \in \{\mathsf{input}, \mathsf{subroutine\text{-}output}\}$.
- If $(\mathsf{ID}, \mathsf{input})$ is in the communication set of some machine $\mu \in \pi$ with identity ID_μ, then there is a machine in π whose identity is ID and whose communication set contains $(\mathsf{ID}_\mu, \mathsf{subroutine\text{-}output})$.
- If $(\mathsf{ID}, \mathsf{subroutine\text{-}output})$ is in the communication set of some machine μ in π *and* ID is the identity of some machine in π, then that machine's communication set contains $(\mathsf{ID}_\mu, \mathsf{input})$.

In the last condition above, if ID is not the identity of any machine in π, we say that μ is a *main machine* of π and that ID is an *external identity* of π.

Another difference is that in UC, the adversary is always a single machine, whereas for us the adversary is allowed to be a network of machines. This difference is not essential, as by our standing assumptions on efficiency, we can replace an efficient network of adversaries with an efficient single adversary simulating the network. We argue that the other differences do not matter either, by giving a formal translation between our setting and UC.

We start by noting that the possible kinds of connectivity do slightly differ between the two approaches. The main difference is that our formalism allows a machine to be connected by multiple wires to itself, to another machine, or to the environment:



We first focus on such connections internal to a resource.

Definition IV.4. Call a resource r in $\mathbf{D_{bd}}$ simple if no machine internal to r is connected by more than one wire of the same type to a machine in r.

We are now in a position to translate from our formalism to UC and back. However, as our machines do not have names (identities) but the ones in UC do, such a naming must be provided. Conversely, when translating a UC-protocol into a state, we need to specify an ordering (so that we know how to order the outgoing wires in a state). The translation theorem is then cleanest to state in terms of functions operating on resources and protocols equipped with the required additional data, which we next define.

Definition IV.5. Fix a (countably infinite) set $\mathcal{N}$ of possible identities of machines. Given a representative $(A^m, r\colon I \to A^n \otimes A^m)$ of a resource on A^n in $\mathbf{D_{bd}}$, we say that a function $\sigma\colon \{1, \ldots, n\} \to \mathcal{N}$ is *boundary-compatible* with r if no machine of r has two external wires both mapped to the same identity under σ. A *valid naming* for (A^m, r) consists of

- a function $\sigma\colon \{1, \ldots, n\} \to \mathcal{N}$ that is boundary-compatible with r, and
- a function $\tau\colon \{1, \ldots, m\} \to \mathcal{N}$ that is injective

such that the images of σ and τ are disjoint, the image of σ does not contain 1 and the image of τ does not contain $0, 1$. A *named resource* is a representative of a resource together with a valid naming for it. Given a named resource $\tilde{r}$, we will write $|\tilde{r}|$ for its underlying state i.e., if $\tilde{r} = (r, \sigma, \tau)$, then $|\tilde{r}| = r$.

Given a UC-protocol π, let $\mathcal{C}(\pi)$ be the set of pairs $(\mathsf{ID}_\mu, \mathsf{ID})$ where μ is a main machine of π and ID is an external identity of π appearing in the communication set of μ, and let $M(\pi)$ be the set of machines in π. An *ordering of the interfaces of* π consists of total orders on $\mathcal{C}(\pi)$ and $M(\pi)$. An *ordered UC protocol* consists of an UC protocol together with an ordering of its interfaces. Given an ordered UC-protocol $\tilde{\pi}$, we will write $|\tilde{\pi}|$ for its underlying protocol.

Theorem IV.6 (Translation)**.** *There is a translation function F sending any named simple resource to an ordered UC protocol, and an inverse translation G sending any ordered UC protocol to a named simple resource. The functions F and G satisfy the following:*

1) *The translation is well-defined: if $r' = (\mathrm{id}_{A^n} \otimes c) \circ r$ for a permutation $c\colon A^m \to A^m$ (witnessing $[A^m, r] = [A^m, r']$ in $\mathbf{D_{bd}}$), then $\left|F(r', \sigma, \tau \circ c^{-1})\right| = |F(r, \sigma, \tau)|$.*
2) *Round-trips are indistinguishable: $|G(F(\tilde{r}))| \approx |\tilde{r}|$ as states in $\mathbf{C}$, and $|F(G(\tilde{\pi}))| \approx |\tilde{\pi}|$ as UC protocols.*
3) *The translation extends to adversaries and environments as follows. A $\mathbf{C}$-morphism $a\colon A^m \to A$ (a candidate adversary acting on the backdoor of r) translates to a UC adversary $F(a)$ for $F(\tilde{r})$. In particular, mux_m from Lemma IV.3 translates to the UC-dummy adversary and vice versa. An environment $\mathcal{E}\colon A^n \otimes A \to A$ in $\mathbf{C}$ translates to a UC environment $F(\mathcal{E})$. Conversely, a UC adversary machine $\mathcal{A}$ attached to a UC hybrid protocol π translates to a $\mathbf{C}$-morphism $A^m \to A$ acting on the backdoor of $|G(\tilde{\pi})|$; a UC environment compatible with $|\tilde{\pi}|$ translates to a $\mathbf{C}$-morphism $G(\mathcal{E})\colon A^n \otimes A \to A$. With these translations, probability ensembles agree in both directions:*

$$\mathsf{EXEC}_{(\mathrm{id}_{A^n} \otimes a) \circ r, \mathcal{E}} = \mathsf{EXEC}^{\mathsf{UC}}_{|F(\tilde{r})|, F(a), F(\mathcal{E})}$$
$$\mathsf{EXEC}^{\mathsf{UC}}_{|\tilde{\pi}|, \mathcal{A}, \mathcal{E}} = \mathsf{EXEC}_{(\mathrm{id}_{A^n} \otimes G(\mathcal{A})) \circ |G(\tilde{\pi})|, G(\mathcal{E})} \ .$$

4) *F and G preserve computational indistinguishability.*

[4] Formally, there are no such restrictions on protocols in [2, Section 2]. However, we believe this is an oversight. Indeed, Canetti's recent tutorial [13] requires this explicitly.

5) *Let $r\colon I \to X$ and $s\colon I \to Y$ be resources and $g\colon X \to Y$ a map in $\mathbf{D}_{\mathbf{real}}$, such that $g \circ r$ and s are simple. Assume (σ, τ) and (σ, τ') give valid namings for $g \circ r$ and s, so that $F(g \circ r, \sigma, \tau)$ and $F(s, \sigma, \tau')$ are compatible in that $\mathcal{C}(F(g \circ r, \sigma, \tau)) = \mathcal{C}(F(s, \sigma, \tau'))$. Then g is a secure emulation $r \to s$ if and only if $F(g \circ r, \sigma, \tau)$ UC-emulates $F(s, \sigma, \tau')$.*

Proof. We begin by defining $F(r, \sigma, \tau)$. Let f_i be the machine connected to the ith adversarial tape of r. Equip f_i with the identity $\tau(i)$, and read off the communication sets from the connectivity of r. As r is simple, this is easy: use the names provided above and read the type of connection in $\{\mathsf{input}, \mathsf{subroutine\text{-}output}\}$ from the object in $\{A_{\rightarrow}, A_{\leftarrow}\}$ corresponding to the wire. We now deal with wires that are not connected to anything: for the jth such wire, add $(\sigma(j), \mathsf{subroutine\text{-}output})$ to the communication set. We also adjust the code of each machine slightly: when a machine gets a message $(\mathsf{ID} \in \mathcal{N}, \mathsf{msg})$, it behaves as the original machine did when getting the message msg on the wire connected to ID. The ordering of interfaces of the resulting protocol is inherited from the order on the wires of r.

For G, we can read off the connections from the communication sets. We modify the machines as follows: when a machine gets a message msg on a wire connected to a machine with identity $\mathsf{ID} \in \mathcal{N}$, it behaves as the original machine did when receiving message $(\mathsf{ID}, \mathsf{msg})$. Moreover, for each machine, we add one more tape connected to the adversarial port: when receiving a message there, the machine will behave as it did when receiving messages via its backdoor tape.

The ordering of the interfaces of π fixes the orders of the honest and backdoor wires. Once those have been fixed, the naming functions σ and τ are inherited from the names of the external identities and the machines of π, and they will result in a valid naming because π is a well-formed protocol.

1) and 2) are now clear from the construction.[5]

3) Let us first discuss the adversaries. For F, we proceed as above, except we first replace the network of adversarial machines by a single (equivalent) machine which we name 1, and add $(1, \mathsf{backdoor})$ to the communication sets of all other machines. For G, we replace all backdoor connections with a wire of type A, and otherwise proceed as above. Any environment can be translated along F and G similarly, and by construction, the resulting probability ensembles are equal.

4) It follows from 3) that if r and s are computationally distinguishable, so are $F(r, \sigma, \tau)$ and $F(s, \sigma, \tau)$ whenever σ, τ give valid namings for both r and s, and a similar claim holds for G. Hence F and G are injective on indistinguishability classes. That F and G also preserve indistinguishability follows from the following general fact: if $(X, \approx_X)$ and $(Y, \approx_Y)$ are two sets equipped with equivalence relations, and $F\colon X \to Y$ and $G\colon Y \to X$ are functions that are injective on equivalence classes satisfying $G(F(x)) \approx_X x$ and $F(G(y)) \approx_Y y$ for all $x \in X, y \in Y$, then F and G also preserve the equivalence relations, i.e., $x \approx_X y \implies F(x) \approx_Y F(y)$ and similarly for G.

5) We now prove that the security notions are equivalent. Assume first that $g\colon r \to s$ is a secure emulation in $\mathbf{D}_{\mathbf{real}}$. Let $\mathcal{A}$ be a UC adversary for $F(g \circ r, \sigma, \tau)$. As $g\colon r \to s$ is secure, for $G(\mathcal{A})\colon A^m \to A$ there is a simulator such that (III.5) holds and hence $F(g \circ r, \sigma, \tau)$ UC-emulates $F(s, \sigma, \tau')$. Conversely, if $F(g \circ r, \sigma, \tau)$ UC-emulates $F(s, \sigma, \tau')$, then for the UC dummy adversary $\mathcal{A}$ there is a corresponding simulator $\mathcal{S}$. As $\mathcal{A}$ translates to mux_m, this implies that $g\colon r \to s$ is a secure emulation by Lemma IV.3 and Corollary III.9. □

Using Theorem IV.6 we see that Corollary III.9 when applied to mux_m from Lemma IV.3 translates to the "completeness of the dummy adversary" in UC. One can also use Theorem IV.6 to deduce that any UC-network is indistinguishable from a naming-invariant one—going back and forth via the translation will hard-code the names, so that renaming machines no longer affects their behavior.

We now discuss the composition theorem of UC for static systems [2, Theorem 3]. Let π, ϕ, ρ be UC protocols. Recall:

- ϕ is a *subroutine* of ρ if $\phi \subseteq \rho$ as a set of ITMs.
- π is *compatible* with ϕ if their main machines have the same set of identities (i.e., there is an identity-preserving *bijective* correspondence between the main machines of π and ϕ), and for each identity in this set, external identities in the communication set of the corresponding machines are equal. Equivalently, π is compatible with ϕ iff $\mathcal{C}(\pi) = \mathcal{C}(\phi)$.
- For ϕ a subroutine of ρ, π is *identity-compatible* with ρ and ϕ if no machine in π has the same identity as one in $\rho \setminus \phi$ or one of the external identities of ρ.

The first two notions above have natural categorical analogues. If ϕ is a subroutine of ρ, then translating them along Theorem IV.6 results in two resources r_ϕ and r_ρ such that r_ρ *factorizes via* r_ϕ, i.e., there exists a map f such that $r_\rho = f \circ r_\phi$. Indeed, this f can be built from $\rho \setminus \phi$ by mimicking how UC protocols are translated. Two compatible protocols are translated by Theorem IV.6 into two resources *on the same object* (equipped with suitably compatible namings). The notion of identity-compatibility seems to be more specific to UC. Namely, if π and ϕ are compatible, then their translates r_ϕ and r_π are resources on the same object, say X. If furthermore ϕ is a subroutine of ρ, we get the factorization $r_\rho = f \circ r_\phi$. We can then readily compose f with r_π obtaining a state $f \circ r_\pi$: no identity-compatibility needed. However, for $(\rho \setminus \phi) \cup \pi$ to be a protocol (and hence for the composition to happen prior to translation), we furthermore need to ensure that there are no clashing identities in π and $\rho \setminus \phi$. We now make this last claim precise.

Claim IV.7. *Let π, ϕ, ρ be UC protocols such that ϕ is a subroutine of ρ, π is compatible with ϕ, and π is identity-compatible with ρ and ϕ. Then $\rho^{\phi \to \pi} := (\rho \setminus \phi) \cup \pi$ is a UC protocol.*

Proof. Since π, ϕ, and ρ are UC protocols and π is identity-compatible with ρ and ϕ, the identities of all machines in

[5] In fact, the round-trips produce networks that are perfectly indistinguishable, but we have not defined that notion here.

$\rho^{\phi\to\pi}$ are distinct from each other and from $\{0,1\}$. Moreover, $\rho^{\phi\to\pi}$ does not use backdoor communication as ρ, π (and ϕ) do not.

Assume now that $(\mathsf{ID}, \mathsf{input})$ is in the communication set of some machine $\mu \in \rho^{\phi\to\pi}$ with identity ID_μ. We need to prove that there is a machine in $\rho^{\phi\to\pi}$ whose identity is ID and whose communication set contains $(\mathsf{ID}_\mu, \mathsf{subroutine\text{-}output})$. We split into cases using the decomposition $\rho^{\phi\to\pi} = (\rho\setminus\phi)\cup\pi$. If $\mu \in \pi$, then the required machine exists in $\pi \subseteq \rho^{\phi\to\pi}$ since π is a UC protocol. If $\mu \in (\rho \setminus \phi)$, then since ρ is a UC protocol, there is a machine in ρ with identity ID and whose communication set contains $(\mathsf{ID}_\mu, \mathsf{subroutine\text{-}output})$. Either this machine exists in $\rho \setminus \phi$, or it exists in ϕ. In the first case, the machine in question exists in $\rho^{\phi\to\pi}$. In the second case, the machine in question is a main machine of ϕ, so by compatibility of ϕ and π, the bijection from main machines of ϕ to those of π gives a machine with the same identity in π whose communication set contains $(\mathsf{ID}_\mu, \mathsf{subroutine\text{-}output})$. In any case, the required machine exists in $\rho^{\phi\to\pi}$.

Assume now that $(\mathsf{ID}, \mathsf{subroutine\text{-}output})$ is in the communication set of some machine μ in $\rho^{\phi\to\pi}$ *and* ID is the identity of some machine μ' in $\rho^{\phi\to\pi}$. We need to prove that the communication set of μ' contains $(\mathsf{ID}_\mu, \mathsf{input})$. We now split into four cases using the decomposition $\rho^{\phi\to\pi} = (\rho \setminus \phi) \cup \pi$:

- $\mu \in \pi$, $\mu' \in \pi$: follows since π is a UC protocol.
- $\mu \in (\rho \setminus \phi)$, $\mu' \in (\rho \setminus \phi)$: follows since ρ is a UC protocol.
- $\mu \in \pi$, $\mu' \in (\rho \setminus \phi)$: let $\tilde{\mu} \in \phi$ correspond to μ under the bijection from main machines of π to those of ϕ: then compatiblity of π and ϕ implies that $(\mathsf{ID}, \mathsf{subroutine\text{-}output})$ is in the communication set of $\tilde{\mu}$. As $\phi \subseteq \rho$ and ρ is a UC protocol, we conclude that the communication set of μ' contains $(\mathsf{ID}_\mu, \mathsf{input})$.
- $\mu \in (\rho \setminus \phi)$, $\mu' \in \pi$: by identity-compatibility ID is not an external identity of ρ. Therefore there is a machine $\mu'' \in \phi$ with identity ID and whose communication set contains $(\mathsf{ID}_\mu, \mathsf{input})$, but this contradicts ϕ being a protocol. Therefore, this case can never arise. □

Note that the proof uses the bijection between the main machines of π and ϕ in both directions.

A similar claim is made in [2, Section 2] but we do not believe it is correct as stated there for the following reasons: defining compatibility there only requires an "identity-preserving *injective* correspondence between the main machines of π and those of ϕ." Assuming this means an injection from the main machines of π to those of ϕ, this is not enough for the claim, since $\rho \setminus \phi$ might want to call a main machine of ϕ that is missing from π. If instead this was intended to mean an identity-preserving injection from the main machines of ϕ to those of π, the end result could still fail to be a UC protocol, for one of the new main machines in π might want to send subroutine outputs to a machine in $\rho \setminus \phi$, in which case $(\rho \setminus \phi) \cup \pi$ fails to be a well-formed UC protocol. We note that the tutorial [13] likewise defines compatibility via an identity- and external-identity-preserving *bijection*, matching our corrected definition.

Similarly, [2, Section 2] only requires an *inclusion* of external identities. However, this is not enough for the claim, since a machine in $\rho\setminus\phi$ might want to call a main machine in ϕ that is missing from the communication set of the corresponding machine in π.

Corollary IV.8 ([2, Theorem 3])**.** *Assume that π, ϕ, ρ are UC protocols, ϕ is a subroutine of ρ, and π is identity-compatible with ρ and ϕ. If π UC-emulates ϕ, then UC protocol $\rho^{\phi\to\pi}$ UC-emulates ρ.*

Proof. Note that π UC-emulating ϕ implies their compatibility (recall that UC-emulation relies on the notion of indistinguishability, which presupposes compatibility for execution to be well-defined in both cases). Therefore $\rho^{\phi\to\pi}$ is a protocol by Claim IV.7. Now order $\mathcal{C}(\pi) = \mathcal{C}(\phi)$ and the machines of π and ϕ arbitrarily. Then extend the order on the machines of ϕ to those of ρ so that the machines of ϕ come last. Let us then translate π, ϕ, ρ along Theorem IV.6. Now the translates of π and ϕ under G give rise to states $r_\pi, r_\phi \colon I \to X$ and $r_\rho \colon I \to Y$ in $\mathbf{D_{bd}}$. Moreover, $\rho \setminus \phi$ gets translated to a morphism f such that $f \circ r_\phi = r_\rho$. Therefore, Theorem III.12 applies, so that $f \circ r_\pi$ emulates $f \circ r_\phi = r_\rho$, which translates back via F to the fact that $\rho^{\phi\to\pi}$ UC-emulates ρ. □

Applying the above blueprint to Theorem III.14 we recover the UCGS theorem of [8] for static systems. This, in turn, paves the way for composition in the presence of global subroutines in other UC-like settings, such as Quantum UC [10].

In practice, compatibility can be achieved by having the ideal functionality s (and its translated UC counterpart) "block" any mismatched ports. This is a natural choice since the environment is universally quantified over. Moreover, this makes any restrictions explicit in the functionality specification. An analogous observation also applies to Theorem III.14, where it is once again natural to leave the global functionality unchanged and instead impose the blocking restriction on the functionalities that use it.

V. Conclusions and Outlook

In this work we have initiated a categorical study of UC for static systems, giving rigorous diagrammatic proofs of basic results such as the composition theorems. Our theory holds generally and gives a template for producing similar theories where ITMs are replaced by other computational systems. Moreover, categorical thinking fixed some prior oversights and resulted in several beneficial changes to UC (doing away with identities of machines altogether, allowing both the environment and the adversary to consist of multiple machines) while remaining inter-translatable with UC.

Our work also opens up fertile grounds for future work:

- Techniques used in [16] can readily be used to develop other variants of the theory. For example, we can develop a version where honest machines are partitioned into n sets (n parties), with all communication across the partitions mediated by a shared functionality. We expect that the resulting categorical theory stands in a similar relationship

to the simpler variant of UC in [6] as our theory here relates to UC for static systems. Similarly, we could readily reason about (abstract) distances up to $\varepsilon > 0$, rather than up to indistinguishability.

- Can we model full UC similarly? Existing categorical tools are not well-suited for studying a dynamically changing network, making this a difficult question.
- As mentioned in the introduction, our theory is not an instance of [16] nor vice versa. This suggests finding a unified categorical framework capturing both. However, we believe that in order to avoid "premature generalization", one should first study categorically other approaches to composability such as Abstract/Constructive Cryptography [3], [39] in general and random systems [41] as a specific instance, quantum UC [10], reactive simulatability [44]–[46], the IITM model [47], and programming-language-based approaches such as IPDL [11], ILC [12], state-separating proofs [19] and others [48], [49].
- The long-term goal is to investigate whether recasting various frameworks under a categorical umbrella would facilitate their comparison and pave the way for translating results across them. (Indeed, [16, Theorem 4.10] demonstrates an instance where symmetric monoidal functors preserve security.) This could be achieved via a web of security-preserving functors, thereby establishing when a protocol secure in one framework remains secure in others, and allowing for the derivation of a unified composition theorem.

AI disclosure: This manuscript was proofread and edited with the assistance of large language models and subsequently revised by the authors, who are responsible for its content.

References

[1] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *42nd Annual Symposium on Foundations of Computer Science—FOCS 2001*, 2001, pp. 136–145.

[2] ——, "Universally composable security," *J. ACM*, vol. 67, no. 5, pp. 28:1–28:94, 2020. [Online]. Available: https://doi.org/10.1145/3402457

[3] U. Maurer, "Constructive cryptography–a new paradigm for security definitions and proofs," in *Joint Workshop on Theory of Security and Applications—TOSCA 2011*, 2011, pp. 33–56.

[4] R. Küsters, "Simulation-based security with inexhaustible interactive turing machines," in *19th IEEE Computer Security Foundations Workshop, (CSFW-19 2006), 5-7 July 2006, Venice, Italy*. IEEE Computer Society, 2006, pp. 309–320. [Online]. Available: https://doi.org/10.1109/CSFW.2006.30

[5] J. Baez and M. Stay, "Physics, Topology, Logic and Computation: A Rosetta Stone," in *New Structures for Physics*, B. Coecke, Ed. Berlin, Heidelberg: Springer, 2011, pp. 95–172. [Online]. Available: https://doi.org/10.1007/978-3-642-12821-9_2

[6] R. Canetti, A. Cohen, and Y. Lindell, "A simpler variant of universally composable security for standard multiparty computation," in *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, ser. Lecture Notes in Computer Science, R. Gennaro and M. Robshaw, Eds., vol. 9216. Springer, 2015, pp. 3–22. [Online]. Available: https://doi.org/10.1007/978-3-662-48000-7_1

[7] R. Canetti, A. Stoughton, and M. Varia, "EasyUC: Using EasyCrypt to mechanize proofs of universally composable security," in *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*. IEEE, Jun. 2019, p. 167–16716. [Online]. Available: http://dx.doi.org/10.1109/CSF.2019.00019

[8] C. Badertscher, R. Canetti, J. Hesse, B. Tackmann, and V. Zikas, "Universal composition with global subroutines: Capturing global setup within plain UC," in *Theory of Cryptography Conference*. Springer, 2020, pp. 1–30.

[9] S. Mac Lane, *Categories for the Working Mathematician*, 2nd ed. Springer, 1998.

[10] D. Unruh, "Universally composable quantum multi-party computation," in *Advances in Cryptology—EUROCRYPT 2010*, 2010, pp. 486–505.

[11] J. Gancher, K. Sojakova, X. Fan, E. Shi, and G. Morrisett, "A core calculus for equational proofs of cryptographic protocols," *Proc. ACM Program. Lang.*, vol. 7, no. POPL, pp. 866–892, 2023. [Online]. Available: https://doi.org/10.1145/3571223

[12] K. Liao, M. A. Hammer, and A. Miller, "ILC: a calculus for composable, computational cryptography," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, Jun. 2019, pp. 640–654.

[13] R. Canetti, "Universally composable security: A tutorial," Slides, UC/EasyUC Summer School, Boston University, Aug. 2025, available at https://www.bu.edu/riscs/files/2025/08/Canetti.pdf.

[14] D. Rausch, R. Küsters, and C. Chevalier, *Embedding the UC Model into the IITM Model*. Springer International Publishing, 2022, p. 242–272. [Online]. Available: http://dx.doi.org/10.1007/978-3-031-07085-3_9

[15] A. Broadbent and M. Karvonen, "Categorical composable cryptography," in *Foundations of Software Science and Computation Structures (FoSSaCS)*, ser. Lecture Notes in Computer Science, vol. 13242. Springer, 2022, pp. 161–183.

[16] ——, "Categorical composable cryptography: extended version," *Logical Methods in Computer Science*, vol. 19, pp. 30:1–30:46, 2023.

[17] M. Prabhakaran and M. Rosulek, "Cryptographic complexity of multi-party computation problems: Classifications and separations," in *Advances in Cryptology—CRYPTO 2008*, 2008, pp. 262–279.

[18] M. Rosulek, *The Joy of Cryptography: An Undergraduate Course in Provable Security*. MIT Press, 2026.

[19] C. Brzuska, A. Delignat-Lavaud, C. Fournet, K. Kohbrok, and M. Kohlweiss, "State Separation for Code-Based Game-Playing Proofs," in *Lecture Notes in Computer Science*. Springer International Publishing, 2018, pp. 222–249. [Online]. Available: https://doi.org/10.1007/978-3-030-03332-3_9

[20] C. Brzuska and S. Oechsner, "A State-Separating Proof for Yao's Garbling Scheme," in *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*. IEEE, 2023, pp. 137–152.

[21] S. Breiner, C. A. Miller, and N. J. Ross, "Graphical methods in device-independent quantum cryptography," *Quantum*, vol. 3, p. 146, 2019.

[22] B. Coecke, Q. Wang, B. Wang, Y. Wang, and Q. Zhang, "Graphical calculus for quantum key distribution (extended abstract)," *Electronic Notes in Theoretical Computer Science*, vol. 270, no. 2, pp. 231–249, 2011.

[23] S. Breiner, A. Kalev, and C. A. Miller, "Parallel self-testing of the GHZ state with a proof by diagrams," in *Proceedings of QPL 2018*, ser. Electronic Proceedings in Theoretical Computer Science, vol. 287, 2018, pp. 43–66.

[24] A. Hillebrand, "Superdense coding with GHZ and quantum key distribution with W in the ZX-calculus," in *Proceedings of QPL 2011*, ser. Electronic Proceedings in Theoretical Computer Science, vol. 95, 2011, pp. 103–121.

[25] M. Stay and J. Vicary, "Bicategorical semantics for nondeterministic computation," in *Proceedings of the Twenty-ninth Conference on the Mathematical Foundations of Programming Semantics, MFPS XXIX*, ser. Electronic Notes in Theoretical Computer Science, vol. 298, 2013, pp. 367 – 382.

[26] L. Colisson, D. Markham, and R. Yehia, "All graph state verification protocols are composably secure," 2024, arXiv:2402.01445.

[27] M. Patrignani, R. Künnemann, R. S. Wahby, and E. Cecchetti, "Universal composability is robust compilation," *ACM Transactions on Programming Languages and Systems*, vol. 46, no. 4, p. 1–64, Dec. 2024. [Online]. Available: http://dx.doi.org/10.1145/3698234

[28] R. Künnemann, M. Patrignani, and E. Cecchetti, "Computationally bounded robust compilation and universally composable security," in *2024 IEEE 37th Computer Security Foundations Symposium (CSF)*. IEEE, Jul. 2024, p. 265–278. [Online]. Available: http://dx.doi.org/10.1109/CSF61375.2024.00024

[29] N. Voorneveld, A. Di Giorgio, and P. Sobociński, "Information leakage in resource theories," in *Proceedings of Applied Category Theory (ACT*

2026), 2026, to appear. Available at https://actconf2026.github.io/papers/ACT_2026_paper_72.pdf.
[30] P. Selinger, "A survey of graphical languages for monoidal categories," in *New structures for physics*. Springer, 2010, pp. 289–355.
[31] R. Piedeleu and F. Zanasi, *An Introduction to String Diagrams for Computer Scientists*, ser. Elements in Applied Category Theory. Cambridge University Press, 2025.
[32] B. Coecke and E. O. Paquette, "Categories for the practising physicist," in *New Structures for Physics*. Springer, 2010, pp. 173–286.
[33] B. Fong and D. I. Spivak, *An Invitation to Applied Category Theory: Seven Sketches in Compositionality*. Cambridge University Press, 2019.
[34] C. Heunen and J. Vicary, *Categories for Quantum Theory: an introduction*. Oxford University Press, USA, 2019.
[35] S. Awodey, *Category theory*. Oxford University Press, 2010.
[36] T. Leinster, *Basic category theory*. Cambridge University Press, 2014, vol. 143.
[37] S. Abramsky and B. Coecke, "A categorical semantics of quantum protocols," in *Logic in Computer Science 19*. IEEE Computer Society, 2004, pp. 415–425.
[38] A. Joyal and R. Street, "The geometry of tensor calculus, I," *Advances in Mathematics*, vol. 88, no. 1, pp. 55–112, 1991.
[39] U. Maurer and R. Renner, "Abstract cryptography," in *Innovations in Computer Science—ICS 2011*, 2011.
[40] R. Canetti, Y. Dodis, R. Pass, and S. Walfish, "Universally composable security with global setup," in *Theory of Cryptography Conference*. Springer, 2007, pp. 61–85.
[41] U. Maurer, *Indistinguishability of Random Systems*. Springer Berlin Heidelberg, 2002, p. 110–132. [Online]. Available: http://dx.doi.org/10.1007/3-540-46035-7_8
[42] M. Bartoletti, I. Castellani, P.-M. Deniélou, M. Dezani-Ciancaglini, S. Ghilezan, J. Pantovic, J. A. Pérez, P. Thiemann, B. Toninho, and H. T. Vieira, "Combining behavioural types with security analysis," *Journal of Logical and Algebraic Methods in Programming*, vol. 84, no. 6, pp. 763–780, Nov. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352220815000851
[43] D. Hofheinz, D. Unruh, and J. Müller-Quade, "Polynomial runtime and composability," *Journal of Cryptology*, vol. 26, no. 3, p. 375–441, Aug. 2012. [Online]. Available: http://dx.doi.org/10.1007/s00145-012-9127-4
[44] B. Pfitzmann and M. Waidner, "A model for asynchronous reactive systems and its application to secure message transmission," in *2001 IEEE Symposium on Security and Privacy—S&P 2001*, 2001, pp. 184–200.
[45] M. Backes, B. Pfitzmann, and M. Waidner, "A general composition theorem for secure reactive systems," in *1st Theory of Cryptography Conference—TCC 2004*, 2004, pp. 336–354.
[46] ——, "The reactive simulatability (RSIM) framework for asynchronous systems," *Information and Computation*, vol. 205, no. 12, pp. 1685–1720, 2007.
[47] R. Küsters, M. Tuengerthal, and D. Rausch, "The IITM model: a simple and expressive model for universal composability," *Journal of Cryptology*, vol. 33, no. 4, pp. 1461–1584, 2020.
[48] D. Micciancio and S. Tessaro, "An equational approach to secure multi-party computation," in *4th Conference on Innovations in Theoretical Computer Science—ITCS 2013*, 2013, pp. 355–372.
[49] D. Hofheinz and V. Shoup, "GNUC: A new universal composability framework," *Journal of Cryptology*, vol. 28, no. 3, pp. 423–508, 2015.
[50] J. Moeller and C. Vasilakopoulou, "Monoidal Grothendieck construction," *Theory and Applications of Categories*, vol. 35, no. 31, pp. 1159–1207, 2020.

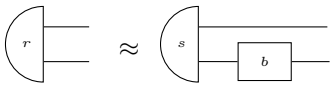
Appendix A
Composition Theorems, Abstractly

In this section we briefly sketch how our theory fits an abstract and general theory in the style of [15], [16]. However, the theory we develop here is not an instance of [15], [16], nor does [15], [16] (or even its motivating examples) fit the theory developed here. We will leave the question of finding a categorical foundation capturing both for future work.

We will assume more categorical knowledge, and in particular the notion of a lax monoidal functor. Recall that a *preorder* is a set equipped with a binary relation $\leq$ that is reflexive and transitive. The category $\mathbf{PreOrd}$ has preorders as its objects and its morphisms consist of monotone functions: functions f satisfying $x \leq y \Rightarrow f(x) \leq f(y)$. The categorical product of preorders (which takes the cartesian product of the underlying sets) equips $\mathbf{PreOrd}$ with a symmetric monoidal structure.

If $\mathbf{C}$ is a category and $F\colon \mathbf{C} \to \mathbf{PreOrd}$ is a functor, then the *Grothendieck construction of* F is the category $\int F$ whose objects are pairs (X, r), where X is an object of $\mathbf{C}$ and r is an element of $F(X)$, and morphisms $(X, r) \to (Y, s)$ are given by morphisms $f\colon X \to Y$ in $\mathbf{C}$ such that $F(f)r \leq s$. It is known that a symmetric lax monoidal structure on the functor F induces an SMC structure on $\int F$ [50].

Definition A.1. Let $r, s\colon I \to X$ be resources, i.e., states in $\mathbf{D_{bd}}$. Define $r \leq s$ iff the identity on X gives a secure emulation $r \to s$, i.e., iff there exists a simulator b such that



Picking b to be the identity shows that $\leq$ is reflexive, and by composing simulators we see that $\leq$ is transitive. Thus, $\leq$ gives a *preorder* on resources of type X. Moreover, the proof of Theorem III.12 shows that if $r \leq s$, then $f \circ r \leq f \circ s$ for any $f\colon X \to Y$ in $\mathbf{D_{bd}}$. Therefore, the functor $\hom(I, -)\colon \mathbf{D_{bd}} \to \mathbf{Set}$ can be promoted to a functor $\hom(I, -)\colon \mathbf{D_{bd}} \to \mathbf{PreOrd}$ landing in the category of preordered sets. Moreover, the functor $\hom(I, -)$ has a canonical symmetric lax monoidal structure with respect to the monoidal structures on $\mathbf{D_{bd}}$ and $\mathbf{PreOrd}$.

We therefore have a composite symmetric lax monoidal functor

$$\mathbf{D_{real}} \hookrightarrow \mathbf{D_{bd}} \xrightarrow{\hom(I,-)} \mathbf{PreOrd}$$

which we will simply denote by F, so that $\int F$ is an SMC.

Theorem A.2. *The SMC $\int F$ is isomorphic to the SMC of resources (states in $\mathbf{D_{bd}}$) and secure emulations (maps in $\mathbf{D_{real}}$ satisfying Definition III.4) between them.*

Proof. The result follows by unwinding definitions: an object of $\int F$ consists of pairs (X, r) where X is an object of $\mathbf{D_{real}}$ and $r\colon I \to X$ is a state in $\mathbf{D_{bd}}$, i.e., a resource. Moreover, morphisms $(X, r) \to (Y, s)$ in $\int F$ are given by morphisms $f\colon X \to Y$ in $\mathbf{D_{real}}$ such that $f \circ r \leq s$. By Definition A.1 and Theorem III.7 these are exactly the secure emulations $r \to s$. The monoidal structure on $\int F$ is obtained from the lax monoidal structure on F, and it is straightforward but tedious to check that this sends the resources (X, r) and (Y, s) to the resources $(X \otimes Y, r \otimes s)$, and thus ultimately results in the same SMC structure. □