Contents

List of Figures

Introduction

What does category theory have to do with cryptography?

Category theory is largely a way of organizing very complicated information so that you can manage to prove things about it rigorously. A good example from mathematics would be homological algebra. It would be very difficult to describe the constructions of homological algebra succinctly without the language of functors and categories.

I think examples like this abound in math, but many mathematicians haven't totally gotten religion. One familiar example is describing parallel transport on a Riemannian manifold M as being a functor P from the category of paths on the manifold (up to reparameterization) to the category of vector spaces which maps each point x to the tangent space $T_x M$. To me this is a nice definition, because it allows me to easily reuse the "mental software" I use to think about functors to think about an aspect of parallel transport.

However, it seems to be somewhat common for people to not say these words, but instead to say that P has the properties which a functor has (it respects composition, takes identity to identity). That is, to simply inline the definition of a functor. I think part of this stems from a kind of suspicion as category theory as being just a language, with not enough mathematical content of its own to merit using.

My personal take is that language matters, and while using circumlocutions and inlining definitions is always possible, it is better to avoid doing so. In addition to being more efficient (in terms of communicating) it also lets you reuse the same mental widgets for understanding disparate things.

As for the present work, as things are in cryptography, things like "secure computation" don't really seem to have formal definitions in practice. One says something about parties who can send messages to each other and maybe draws a diagram with arrows going back and forth 3.

There have been attempts to (https://eprint.iacr.org/2000/067.pdf) create formalisms to address this particular issue, although the only

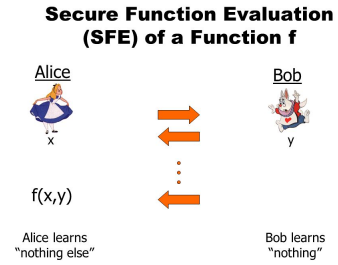


Figure 1: An example of the kind of picture a cryptographer might draw to indicate interactive computation. Part of our goal is to give a mathematical account of such pictures.

one I know of is extremely low-level and not very user-friendly, so to speak.

I predict the reaction to the categorical way of talking proposed herein from mainstream cryptographers would be largely the same as the response of a differential geometer if you tried to tell them to talk about parallel transport as a functor. That is, I imagine they would ask something like "What's the point? This is more complicated than what we already have and doesn't prove anything new." This is in some sense true. There are several counter-arguments:

(1) There is no usable definition of the interactions that are the primary object of study for cryptography. That is, of the interactions in interactive proofs, in secure computation, in encryption, etc. There have been many definitions proposed over the years, but they tend to be "anti-modular", ad hoc, and tied to a particular model of computation. The categorical definition we give is totally modular with respect to what kinds of computations parties may perform.

(2) The theory developed here generalizes cryptography (and also information theory) to a theory of interactive computation. I believe the zoomed out perspective will prove useful in understanding what cryptography is in a broader sense.

I don't intend for the theory described here to be a definitive account of categorical cryptography, only a demonstration of how the notions of category theory may be used to give elegant definitions of cryptography. The basic constructions are flexible to modification for one's particular purposes, and they should be so modified.

Overview

Our primary goal here is to describe what sort of thing is pictured in Figure 2, Figure 3, or Figure 4. That is, what is a good mathematical definition for the kind of interactive computations that cryptographers routinely depict using such diagrams, and which are at the core of many definitions in cryptography?

There are a few desiderata for such a definition which we'll try to achieve. Namely, we'll give a definition of interactive computation which satisfies the following.

(1) **Usable.** The definition should be as easy to work with rigorously as possible.

(2) **Modular.** The definition should be modular (even functorial) over the model of computation used by various parties (e.g., it could be polynomial-time probabilistic Turing machines, non-uniform families of P-size circuits, lambda calculus, etc.).

(3) **General.** The definition should be general enough to capture a wide range of notions of interactive computation, and should allow for easy exploration of nearby ideas in the mathematical space.

Before we begin, it is important to be comfortable with the basic notions of category theory. Namely, categories, functors, and natural transformations. For a short treatment of these topics aimed at cryptographers, see appendix.

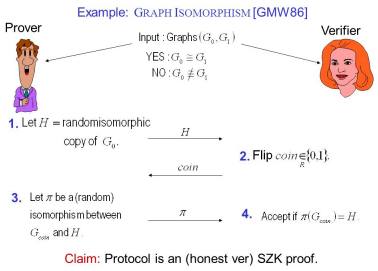


Figure 2: Graph isomorphism zero knowledge proof

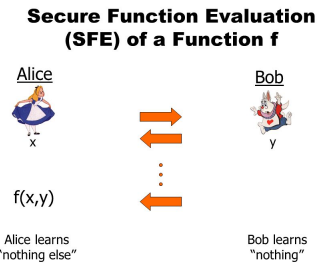


Figure 3: An example of the kind of picture a cryptographer might draw to indicate interactive computation. Part of our goal is to give a mathematical account of such pictures.

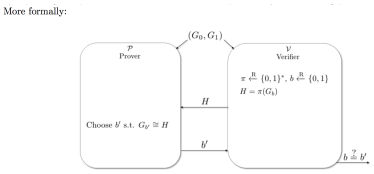


Figure 4: TODO

There is some existing work which try to do things in the same vein. Some references are. We won't discuss these approaches much.

Broad strokes

At a high level, we can describe the approach as follows.

(1) Given a category of computations $\mathcal{D}$ (e.g., PPTs, circuit families, etc.) and a network of channels $\mathcal{C}$, we define a category of interactive computations, called $\mathcal{C}[\mathcal{D}]$, which uses the computations in $\mathcal{D}$ and the channels in $\mathcal{C}$

(2) $\mathcal{C}[\mathcal{D}]$ will give an extremely syntactic model of interactive computations. To study the more concrete models of computation which we care about, like communication over insecure or noisy channels, we will study functors

$$\rho : \mathcal{C}[\mathcal{D}] \to \mathcal{R}$$

where $\mathcal{R}$ is some category of "concrete interactive computations".

We call such a functor a representation.

One concrete communication category of primary interest will be $\mathsf{RPCircWire}_{\mathcal{C}}$, the category of interactive computations over a potentially insecure channel. Two important representations in this category will be

1.

$$\mathsf{leak}_{\mathcal{C}} : \mathcal{C}[\mathsf{RPCirc}] \to \mathsf{RPCircWire}_{\mathcal{C}}$$

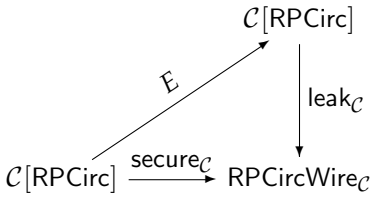
which leaks every message transmitted to an adversary.

2.

$$\mathsf{secure}_{\mathcal{C}} : \mathcal{C}[\mathsf{RPCirc}] \to \mathsf{RPCircWire}_{\mathcal{C}}$$

which transmits every message completely securely, with no information being leaked to an adversary.

A crypto system will then be a functor $E : \mathcal{C}[\mathsf{RPCirc}] \to \mathcal{C}[\mathsf{RPCirc}]$ such that the diagram



commutes up to natural isomorphism. Put simply, E is a way of transforming a communication channel c into an interactive computation Ec such that doing Ec and leaking all messages transmitted to the adversary is indistinguishable from communicating with perfect security over c.

We continue by re-defining other notions from cryptography in this more general setting and exploring their meaning in categories of interactive communication other than that of insecure channels. Most notably, we see what the usual cryptographic definitions mean in the context of noisy channels, for example noting that cryptosystems correspond to error-correcting codes.

Interaction categories

Cryptography studies the interaction between computation and communication. For example, consider the following questions:

1. *Encryption* How can two people modify their messages to communicate securely over an insecure channel?
2. *Secure computation* How can n parties jointly compute a function, while leaking as little information as possible about their respective inputs to the function?
3. *Indistinguishability obfuscation* How can I send you a program which you can run, while learning as little as possible about how it works?

These are typical questions in cryptography and all involve the relationship between computation and information spread during an interaction.

Our first goal will be to investigate the question of what is an interaction, or more specicfically, what is an interactive computation. A typical interactive computation is shown in Figure 5. It represents the following interactive computation: A and B each start off with some data. B does f on his and sends part of the result along channel c to C, who does g on it and sends it back to A along d. Meanwhile, A sends her data to B along e. B then combines all the data and feeds it to h.

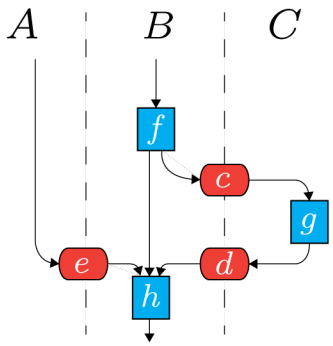


Figure 5: An interactive computation.

We would like to define a category of such things, where, e.g., a morphism of type

$$(A, \alpha) \otimes (B, \beta) \to (C, \gamma) \otimes (D, \delta)$$

will represent an interactive computation in which A starts off with a value of type α, B starts off with a value of type β, and ends with C having a value of type γ and D having a value of type δ.

Communication networks

To have a notion of interactive computation, we'll first off need to have a set of systems which can interact! Let's say our systems

(which will be communicating with each other) are given by a set of names $\{ A, B, C, \dots \}$. Now that we have some systems, we need to know which is capable of sending messages to which. For example, say B and C can both send messages to each other, and A can send messages to B. We could summarize the situation as in Figure 6

We make the following definition.

Definition 1. A communication network (or often just a network) is a directed graph (which may have loops or multiple parallel edges).

So, we think of Graph as the category of networks.

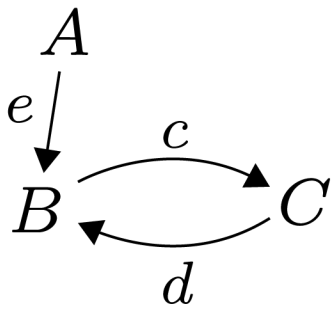


Figure 6: A little network

Free categories

Given a network $\mathcal{C}$, we may consider the "free category" generated by that network, which we will also typically call $\mathcal{C}$. The objects of $\mathcal{C}$ are just the vertices in $\mathcal{C}$ and the morphisms $u \to v$ are the paths in $\mathcal{C}$ from u to v. Clearly we can compose paths, and we have identities given by empty paths.

The interaction category

We have a concept of networks of channels and systems which may communicate over them, but the systems at the ends of the channels don't do much. They ostensibly do anything in fact. We'll remedy that with a construction we'll call the *interaction category*. It's a way of taking a network $\mathcal{C}$ and a category of data $\mathcal{D}$ and getting a category $\mathcal{C}[\mathcal{D}]$ (pronounced, $\mathcal{D}$ along $\mathcal{C}$ or the interaction category of $\mathcal{C}$ and $\mathcal{D}$) which classifies sequences of interactions between network members.

For example, if $\mathcal{C}$ is a network with channels allowing A and B to communicate, and the data category is RPCirc 15, $\mathcal{C}[\mathsf{RPCirc}]$ contains a morphism representing the following interaction:

(1) A picks a prime p, a number $0 < g < p$ at random, and a secret number a at random. She sends B the tuple $(p, g, g^a \mod p)$

(2) B picks a secret number b at random and sends alice $(g^b \mod p)$.

(3) A computes $(g^b \mod p)^a$.

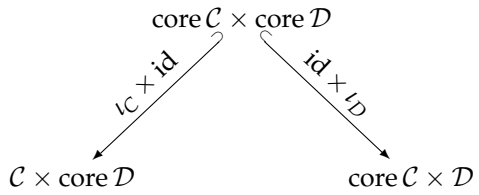
(4) B computes $(g^a \mod p)^b$.

which is of course Diffie-Hellman key exchange.

To that end, let's define the interaction category $\mathcal{C}[\mathcal{D}]$ for $\mathcal{D}$ a monoidal category.

First, we have to define the category of "single threaded interactions".

Definition 2. Given categories $\mathcal{C}$ and $\mathcal{D}$, the category of "single threaded interactions" on the two is the free product $\mathcal{C} * \mathcal{D}$ of the two. Specifically, it is the pushout of the following diagram.



where ι_C, ι_D denote the inclusions. More explicitly, we describe it as follows. The objects of $\mathcal{C} * \mathcal{D}$ are pairs (A, α) with $A \in \mathcal{C}$ and $\alpha \in \mathcal{D}$. The morphisms are generated by

$$(A, f) : (A, \alpha) \to (A, \beta)$$
$$(p, \alpha) : (A, \alpha) \to (B, \alpha)$$

for $f \in \mathsf{Hom}_{\mathcal{D}}(\alpha, \beta)$ and $p \in \mathsf{Hom}_{\mathcal{C}}(A, B)$ subject to the relations ensuring that $(A, -)$ and $(-, p)$ are functorial. I.e.,

$$(A, f_2) \circ (A, f_1) = (A, f_2 \circ f_1)$$
$$(p_2, \alpha) \circ (p_1, \alpha) = (p_2 \circ p_1, \alpha)$$
$$(A, \mathrm{id}_\alpha) = \mathrm{id}_{(A,\alpha)} = (\mathrm{id}_A, \alpha)$$

$\mathcal{C} * \mathcal{D}$ is called the free product of $\mathcal{C}$ and $\mathcal{D}$, since if $\mathcal{C}$ and $\mathcal{D}$ are monoids (resp., groups), $\mathcal{C} * \mathcal{D}$ will be the free product as monoids (resp., groups).

We will also write $\mathsf{at}_A(f)$ for (A, f) (thinking of it as meaning "do f at location A") and $\mathsf{along}_\alpha(p)$ for (p, α) (thinking of it as meaning "send data of type α along channel p"). So, rewriting the above, we'll have

Doing f_1 at A and then f_2 at A is the same as doing $f_2 \circ f_1$ at A:

$$\mathsf{at}_A f_2 \circ \mathsf{at}_A f_1 = \mathsf{at}_A(f_2 \circ f_1)$$

Sending data along p_2 and then p_1 is the same as sending data along $p_2 \circ p_1$:

$$\mathsf{along}_\alpha p_2 \circ \mathsf{along}_\alpha p_1 = \mathsf{along}_\alpha(p_2 \circ p_1)$$

A morphism in $\mathcal{C} * \mathcal{D}$ should be thought of as a single thread of an interaction. E.g., if we have

$$f : \alpha \to \beta$$
$$g : \beta \to \gamma$$
$$p : A \to B$$
$$q : B \to A$$

then

$$\begin{aligned} &\mathsf{along}_\gamma q \\ &\circ \mathsf{at}_B(g) \\ &\circ \mathsf{along}_\alpha(p) \\ &\circ \mathsf{at}_A(f) \end{aligned}$$

Represents the following thread of interaction:

(1) Compute f at location A

(2) Send the result to B along channel p

(3) Compute g at location B

(4) Send the result to A along channel q

The whole process is a morphism $(A, \alpha) \to (A, \gamma)$. That is, an interaction with begins which turns a value of type α at A into a value of type γ at A.

Multiple threads

Now that we have a category representing single threads of interactive computation, we would like a way of representing multiple threads executing concurrently. This will be the category $\mathcal{C}[\mathcal{D}]$. We assume that $\mathcal{D}$ is symmetric monoidal with monoidal product $\otimes_{\mathcal{D}}$. When there is no confusion, we will also write $\otimes$ for the monoidal structure on $\mathcal{D}$.

First take the free symmetric monoidal category on $\mathcal{C} * \mathcal{D}$ with monoidal product $\otimes$ and freely adjoin maps

$$\begin{aligned} \mathrm{fork}^A_{\alpha,\beta} &: (A, \alpha \otimes_{\mathcal{D}} \beta) \to (A, \alpha) \otimes (A, \beta) \\ \mathrm{sync}^A_{\alpha,\beta} &: (A, \alpha) \otimes (A, \beta) \to (A, \alpha \otimes_{\mathcal{D}} \beta) \end{aligned}$$

natural in α and β and such that

$$\begin{aligned} \mathrm{fork}^A_{\alpha,\beta} \circ \mathrm{sync}^A_{\alpha,\beta} &= \mathrm{id}_{(A,\alpha)\otimes(A,\beta)} \\ \mathrm{sync}^A_{\alpha,\beta} \circ \mathrm{fork}^A_{\alpha,\beta} &= \mathrm{id}_{(A,\alpha\otimes_{\mathcal{D}}\beta)} \end{aligned}$$

and such that fork^A witnesses $\mathsf{at}_A{:}\mathcal{D} \to \mathcal{C}[\mathcal{D}]$ as a symmetric monoidal functor.

Finally, adjoin maps

$$\begin{aligned} \pi_1 &: \sigma \otimes \tau \to \sigma \\ \pi_2 &: \sigma \otimes \tau \to \tau \end{aligned}$$

natural in σ and τ respectively. Note that these are not cartesian projections. This is primarily because (assuming a map $\Delta : \alpha \to \alpha \otimes \alpha$) we do not necessarily expect to have an equality

$$(A,\alpha) \xrightarrow{\Delta} (A,\alpha\otimes\alpha) \xrightarrow{\text{fork}^A_{\alpha,\alpha}} (A,\alpha)\otimes(A,\alpha) \xrightarrow{f\otimes g} (B,\beta)\otimes(C,\gamma) \xrightarrow{\pi_1} (B,\beta)$$
$$=$$
$$(A,\alpha) \xrightarrow{f} (B,\beta)$$

The first morphism involves C in the computation, and the communication with C doesn't disappear just because we forget C was involved.

The above example illustrates the way in which interactive computations are represented in this category. The type $(A,\alpha)\otimes(B,\beta)$ may be thought of as representing a thread of execution holding a value of type α at A and a thread holding a value of type β at B.

Under this interpretation, $\otimes$ lets us run interactions at different locations in parallel, fork^A lets us fork off a thread of execution at A with one piece of data to become two threads with two pieces of data, and sync^A lets us join two threads, combining their respective data. Thought of this way, $\mathcal{C}[\mathcal{D}]$ can be seen as a sort of programming language for concurrency [1].

[1] To be a bit more precise, you can make this a programming language by thinking of the type (A,α) as a value of type α at a location A. fork^A lets you fork off a thread at A, channels let you communicate between processes and wait^A lets you join threads

Proposition 3. *The map $-[\mathcal{D}]$ is a functor* $\mathsf{Graph} \to \mathsf{Cat}$.

Proof. We will not give a detailed proof. Intuitively, if we have a map $F : \mathcal{C} \to \mathcal{C}'$ which associates to each channel in $\mathcal{C}$ a channel in $\mathcal{C}'$, we get a map from $\mathcal{C}[\mathcal{D}]$ to $\mathcal{C}'[\mathcal{D}]$ which, whenever an interaction uses a channel $c : A \to B$, we replace it with the channel $Fc : FA \to FB$. □

Realizing communication

$\mathcal{C}[\mathcal{D}]$ is an abstract model of interactions. We would like to construct several concrete descriptions of interactions to compare it against.

In general, we define a model, or a representation, or a realization, or a theory of communication, as follows:

Definition 4. A model of communication (with computational category $\mathcal{D}$) is a functor $\mathcal{R} : \mathsf{Graph} \to \mathsf{Cat}$ equipped with a natural transformation $\rho : -[\mathcal{D}] \to \mathcal{R}$.

Models of communication form a category $\mathsf{Models}_{\mathcal{D}}$ with a mor-

phism $(\widehat{\tau}, \tau) : (\mathcal{R}, \rho) \to (\mathcal{R}', \rho')$ given by a commutative square

$$\begin{array}{ccc} -[\mathcal{D}] & \xrightarrow{\widehat{\tau}} & -[\mathcal{D}] \\ \rho \downarrow & & \downarrow \rho' \\ \mathcal{R} & \xrightarrow{\tau} & \mathcal{R}' \end{array}$$

The most trivial example of a model is $-[\mathcal{D}]$ itself, equipped with the identity natural transformation.

There are however, many more models of communication. One of the most important for cryptography is a functor we will call RPCircWire. It is the category where transmissions over $\mathcal{C}$ are "recorded". It is only well defined when $\mathcal{C}$ is a free category. But first, we need a notion of when two maps give the same computational information.

Informational equivalence

Consider maps $f_1, f_2 : \alpha \to \mathsf{Dist}\,(2^* \times \beta)$. I.e., f_i takes $a : \alpha$ and probabilistically generates a value $\pi_2\,(f_i(a)) : \beta$ along with some auxilliary sidechannel information $\pi_1\,(f_i(a)) : 2^*$. This models a probabilistic computation which also leaks some information.

We would like to identify computations which roughly compute the same values and also roughly leak the same information. When should f_1 and f_2 be considered to leak the same information? To describe this, we need the notion of informational equivalence.

Definition 5. For two ensembles of distributions 16 D_1, D_2, we say that D_1 has more computational information than D_2 (written $D_1 \gtrsim_I D_2$) if there exists a PPT machine M such that

$$M(D_1) \cong_c D_2$$

That is, if D_1 can be approximately transformed into D_2. We write $D_2 \lesssim_I D_1$ if $D_1 \gtrsim_I D_2$.

We say D_1 and D_2 are informationally equivalent (written $D_1 \simeq_I D_2$) if $D_1 \gtrsim_I D_2$ and $D_2 \gtrsim_I D_1$. I.e., the same information can be recovered from both distributions (up to indistinguishability). [2]

Example 6.

The wire category

We'll now define a category called $\mathsf{RPCircWire}_{\mathcal{C}}$ which may be thought of as a concrete realization of the abstract interactions of $\mathcal{C}[\mathcal{D}]$.

[2] This definition is inherent in many of the notions of theoretical cryptography and is usually framed in terms of "simulators".

For example, the claim that a proof system Π is zero knowledge is equivalent to the claim that for any verifier V and any input x, $\Pi(V, x) \lesssim_I x$. I.e., the view of the verifier in the interaction contains at most as much computational information as the input x itself.

The objects of $\mathsf{RPCircWire}_{\mathcal{C}}$ are products of the form

$$(A_1, \alpha_1) \times \cdots \times (A_n, \alpha_n)$$

with $A_i \in \mathcal{C}, \alpha_i \in \mathsf{RPCirc}$. A map

$$(A_1, \alpha_1) \times \cdots \times (A_n, \alpha_n) \to (B_1, \beta_1) \times \cdots \times (B_m, \beta_m)$$

is given by a PPT which maps $\alpha_1 \times \cdots \times \alpha_n$ to a value of type $\beta_1 \times \cdots \times \beta_m$ along with some auxilliary output. In other words, it is a map

$$f : \alpha_1 \times \cdots \times \alpha_n \to 2^* \times (\beta_1 \times \cdots \times \beta_m)$$

But, we identify maps according to the following equivalence relation. If $f, g : \alpha \to \mathsf{Dist}\,(2^* \times \beta)$, then f is equivalent to g if for all $a \in \alpha$,

(1)

$$\pi_1\,(f(a)) \simeq_I \pi_1\,(g(a))$$

(2)

$$\pi_2\,(f(a)) \cong_c \pi_2\,(g(a))$$

In other words, two functions are identified if they compute roughly the same values and the data they leak on the wire have the same computational information.

There is a monoidal structure $\times$ on $\mathsf{RPCircWire}_{\mathcal{C}}$ given by

$$(f \times g)(x) = \text{let } (w_1, y_1), (w_2, y_2) = f(x), g(x) \text{ in } (w_1 +\!\!+ w_2, (y_1, y_2))$$

In other words, if you run protocols f and g in parallel, then what's written on the wire is what gets written by f along with what gets written by g and the value returned is the values returned by f and g together. We will use this monoidal structure to give a representation of interactive computation in $\mathsf{RPCircWire}_{\mathcal{C}}$.

Cryptosystems

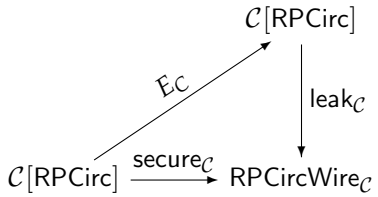
We're now in a position to define what a cryptosystem is. Essentially, a crypto system will be a way of factoring an ideal conception of channels as being perfectly secure through the "real" (or more accurately "pessimistic") conception of channels which has them leaking all their data to an adversary. To that end, we define the following

representations:

$$\begin{aligned}
&\mathsf{secure}_{\mathcal{C}} : \mathcal{C}[\mathsf{RPCirc}] \to \mathsf{RPCircWire}_{\mathcal{C}} \\
&\mathsf{secure}_{\mathcal{C}}(A,\alpha) = (A,\alpha) \\
&\mathsf{secure}_{\mathcal{C}}(c : (A,\alpha) \to (B,\alpha)) = x \mapsto (\text{“”}, x)
\end{aligned}$$

$$\begin{aligned}
&\mathsf{leak}_{\mathcal{C}} : \mathcal{C}[\mathsf{RPCirc}] \to \mathsf{RPCircWire}_{\mathcal{C}} \\
&\mathsf{leak}_{\mathcal{C}}(A,\alpha) = (A,\alpha) \\
&\mathsf{leak}_{\mathcal{C}}(c : (A,\alpha) \to (B,\alpha)) = x \mapsto (x, x)
\end{aligned}$$

A crypto system is then a natural transformation $E_C : \mathcal{C}[\mathsf{RPCirc}] \to \mathsf{RPCircWire}_{\mathcal{C}}$ such that the diagram



commutes up to natural isomorphism for every C.

This is a significantly weaker

The information theoretic setting

One important one for information theory is the following. Let Info be the category whose objects are families of finite sets of bitstrings $\alpha, \beta, \ldots$ and whose maps $\alpha \to \beta$ are maps $\alpha \to \mathsf{Dist}(\beta)$ modulo asymptotic statistical equivalence. I.e., maps $f, g : \alpha \to \mathsf{Dist}(\beta)$ are identified if

$$\lim_{n\to\infty} \sup_{x\in\alpha_n} \sup_{B\subseteq\beta_n} \left|\mathsf{Pr}_{f(x)}(B) - \mathsf{Pr}_{g(x)}(B)\right| = 0$$

I.e., the distributions the two maps induce on each input agree asymptotically.

We then define a realization $(\mathsf{Flip}_p, \rho^{\mathsf{Flip}_p})$ by $\mathsf{Flip}_p(\mathcal{C}) = \mathsf{Info}$ and $\rho_{\mathcal{C}}^{\mathsf{Flip}_p} : \mathcal{C}[\mathsf{RPCirc}] \to \mathsf{Info}$ to be the map which sends $\otimes$ to cartesian product and each channel $c : (A,\alpha) \to (B,\alpha)$ to the map $\mathsf{flip}_{p,\alpha} : \alpha \to \mathsf{Dist}(\alpha)$ which flips each bit in its input with probability p.

Info is, roughly speaking, the Kleisli category for some kind of probability distribution monad and thus we get a representation $\mathsf{perfect}_{\mathcal{C}} : \mathcal{C}[\mathsf{RPCirc}] \to \mathsf{Info}$ which sends $\otimes$ to cartesian product as above and sends each channel $c : (A,\alpha) \to (B,\alpha)$ to the map $x \mapsto \delta_x$, where δ_x is the distribution with all its probability on x. This is a realization of channels where the values are not at all distorted.

We observe that a map of representations $\mathsf{perfect}_{\mathcal{C}} \to \mathsf{Flip}_p$ is essentially an error correcting code.

Appendices

Basic definitions

Notation and basic definitions

Definition 7. 2 is the set $\{ 0, 1 \}$.

Computation categories

Polysets and circuits

Categories are very useful for organizing models of computation. They are frequently used in describing typed programming languages, but they're quite useful in describing the models of computation of interest to complexity theorists, algorithmists, and cryptographers.

Definition 8. A *polysize ensemble of sets* is a sequence of sets $\{ \alpha_n \}_{n \in \mathbb{N}}$ where α_n is a subset of $2^{f(n)}$ for some P-computable function f and such that there is a polytime algorithm A which given x and n decides whether $x \in \alpha_n$.

We may also refer to such a thing as a *polyset*.

We use lowercase Greek letters $\alpha, \beta, \gamma, \ldots$ as variables for polysets.

Example 9. One simple polyset is $\alpha = \{ \alpha_n \}_{n \in \mathbb{N}}$ where α_n is the set of binary strings of length n^2 which represent adjacency matrices of undirected graphs. This is a polyset since one can decide in polynomial time whether a string of length n^2 represents an adjacency matrix of a graph. All you have to do is check that it gives a symmetric matrix.

Definition 10. An family of circuits f from α to β is a sequence $\{ f_n \}_{n \in \mathbb{N}}$ such that f_n is a ciruit mapping from α_n to β_n. A

Notation 11. When the intent is clear, we drop the sequence notation for polysets and refer to them "pointwise". For example, instead of writing $\left\{ 2^{n^2} \right\}_{n \in \mathbb{N}}$ to describe the polyset whose nth element is the set of all strings of length n^2, we would simply write 2^{n^2}.

Example 12. Let α be the polyset of adjacency matrices as above. Let

$$\mathsf{numComponents} : \alpha \to 2^{\lceil \log_2(n^2) \rceil}$$

be a family of circuits which computes the number of connected components in a graph in binary. This can be done with circuits of polynomial size.

Now we define a category PCirc of circuits. PCirc is a very familiar object for the theoretical cryptographer.

Definition 13. PCirc is a category. The objects of PCirc are polysets.

$\mathsf{Hom}_{\mathsf{PCirc}}(\alpha, \beta)$ is the set of uniform P computable families of circuits from α to β and composition is given by sequential composition of circuits. I.e., $(f \circ g)_n$ the circuit given by connecting the outputs of g_n to the inputs of f_n. This is plainly associative and has the trivial circuit families as identities.

There are other models of computation which will be of interest to us. One important one is randomized polynomial time computation. We formalize it as follows:

Definition 14. A randomized circuit of type $\alpha \to \beta$ is a circuit of type

$$\alpha \times 2^r \to \beta$$

where α, β are polysets and r is a function $\mathbb{N} \to \mathbb{N}$.

If we have randomized circuits

$$f : \alpha \times 2^r \to \beta$$
$$g : \beta \times 2^s \to \gamma$$

we may compose them as in Figure 7 to get a circuit $g \circ f : \alpha \times 2^{r+s} \to \gamma$. $g \circ f$ passes the first r bits of randomness to f and then passes the result of f along with the last s bits of randomness to g. This composition is associative and has for its identity the identity circuits which use no randomness. So, we make the following definition:

Definition 15. RPCirc is the category whose objects are polysets and whose maps are polysize randomized circuits, with composition defined as above.

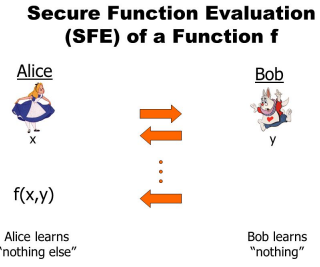


Figure 7: How to compose randomized circuits

Ensembles of distributions

We will also need the notion of an ensemble of distributions.

Definition 16. An ensemble of distributions is a sequence of probability distributions $D = \{ D_n \}_{n \in \mathbb{N}}$ where each D_n is a distribution on $2^{f(n)}$ for some function f.

Negligibility Often in cryptography we want to show that some event happens infrequently. The way we formalize this is with the concept of *negligibility*.

Definition 17. A function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for every $c \geq 0$, for all sufficiently large x we have $|f(x)| < \frac{1}{x^c}$.

An sequence of events $e_0, e_1, e_2, \ldots$ will be said to occur with negligible probability if the function $P(n) = \mathsf{Pr}[e_n]$ is negligible.

Indistinguishability An important notion in cryptography is that of indistinguishability of distributions.

Definition 18. Two ensembeles of distributions D and E (with D_n and E_n distributions on the same polyset α) are indistinguishable if for every non-uniform randomized polynomial size circuit $\mathcal{A} = \{ \mathcal{A}_n \}_{n \in \mathbb{N}}$, $\mathcal{A} : \alpha \to 2$, we have that the quantities

$$|\mathsf{Pr}_{x \sim D_n}[\mathcal{A}_n(x) = 0] - \mathsf{Pr}_{x \sim E_n}[\mathcal{A}_n(x) = 0]|$$

are negligible in n (where the probabilites are also taken over the random bits for $\mathcal{A}_n$).

Graphs

The kinds of graphs we will use most here are directed graphs or digraphs which, for us, may have multiple edges between any two vertices and also may have loops. These are sometimes called quivers.

Graphs form a category Graph where the morphisms $G \to H$ are pairs of maps $f_V : V(G) \to V(H)$ and $f_E : E(G) \to E(H)$ such that if e is an edge from u to v then $f_E(e)$ is an edge from $f_V(u)$ to $f_V(v)$. [3]

[3] A slick definition of this category would be that Graph is the category of functors from $Q = E \rightrightarrows V$ to FinSet. To be clear Q is the category with two objects E and V and two parallel morphisms source, target : $E \to V$.