RILEY SHAHAR

TITLE

Contents

List of Tables

List of Figures

1 Cryptography

Cryptography is the mathematical study of secure computation. In a computation, we want to use *protocols* to transform *resources*. For the computation to be secure, it must successfully resist *attacks* by *adversaries*. We will make all of these notions precise, but first we discuss a motivating example.[1]

Commitment Protocols

Suppose we want to build an online rock-paper-scissors game. Two players, Alice and Bob, should both be able to send each other a move and so determine the winner. However, something needs to prevent the players from waiting until after they learn the other's move to choose their own move. This is an ideal use-case for a *commitment protocol*.

Informally, a commitment protocol proceeds as follows. The sender S has a message m that they wish to commit to—in our case, m is one of $\{R, P, S\}$. In the *commit phase*, they send a commitment c to the receiver R. At some later time, S may reveal m—plus maybe some auxillary data, for example their random bits—to R, in which case R should be able to verify that c was indeed a commitment to m.

We can formalize commitment schemes as follows:

Definition 1.1 (commitment protocol)**.** A *commitment protocol* consists of the following data:

- the *message space* $\mathcal{M}$ and *commit space* $\mathcal{C}$;
- a pair of families of probabalistic interactive algorithms[2] S_n and R_n (indexed by the *security parameter* $n \in \mathbb{N}$), respectively called the *sender* and *receiver*, such that:
  - in the *commit phase*, S_n gets $m \in \mathcal{M}$ and returns $c \in \mathcal{C}$, while R_n returns $\bot$ or $\top$;
  - in the *reveal phase*, S_n gets $m \in \mathcal{M}$, while R_n gets $c \in \mathcal{C}$, returning $\bot$ or $m' \in \mathcal{M}$.

Notation. Let S and R be interactive algorithms as in Definition 1.1.

[1] Many more examples can be found in any introductory text on cryptography, such as [KL14; Ros21; PS10].

[2] We have been imprecise about our formal notion of algorithm. For our purposes in this chapter, Turing machines suffice; we will be more precise about this later. In particular, it will be important that we consider the security parameter as indexing a family of algorithms, rather than as a unary input to each algorithm as is common in the literature.

- We will write $\texttt{Com}_S^R(m)$ for the output of S in the commit phase with S getting input $m \in \mathcal{M}$, or $\perp$ if R returns $\perp$.
- We will write $\texttt{Rev}_S^R(m, c)$ for the output of R in the reveal phase with S getting input $m \in \mathcal{M}$ and R getting input $c \in \mathcal{C}$.

  Where S and R are clear, we may omit the respective annotations.

It is worth elaborating on the ubiquitous notion of a *security parameter*, as seen in Definition 1.1.

TODO: this

We would like the commitment scheme to be correct, in that when the parties behave honestly according to the protocol, the receiver returns the correct message. Formally:

Definition 1.2. A commitment protocol (S_n, R_n)[3] is *correct* if for all $n \in \mathbb{N}$ and $m \in \mathcal{M}$,

$$\Pr[\texttt{Rev}_{S_n}^{R_n}(m, \texttt{Com}_{S_n}^{R_n}(m)) = m] = 1.$$

Given a commitment protocol (S_n, R_n), we should be able to define a family of algorithms for our rock-paper-scissors game as follows. Let $\mathcal{M} = \{R, P, S\}$ and let $W : (\perp \sqcup \mathcal{M})^2 \to \{1, 0, -1, \perp\}$ compute whether the first argument beats, ties, or loses to the second, propagating any $\perp$s.

Protocol 1.3 (Rock-Paper-Scissors)**.**

1. A_n receives input $a \in \mathcal{M}$; B_n receives input $b \in \mathcal{M}$.
2. A_n acts as S_n and B_n as R_n to compute $c_a = \texttt{Com}_{S_n}^{R_n}(a)$.
3. A_n acts as R_n and B_n as S_n to compute $c_b = \texttt{Com}_{S_n}^{R_n}(b)$.
4. A_n acts as S_n and B_n as R_n to compute $a' = \texttt{Rev}_{S_n}^{R_n}(a, c_a)$.
5. A_n acts as R_n and B_n as S_n to compute $b' = \texttt{Rev}_{S_n}^{R_n}(b, c_b)$.
6. A_n returns $W(a, b')$.
7. B_n returns $W(a', b)$.

Notation. Given some fixed commitment protocol, we will write $\textsf{RPS}_A^B(a, b)$ for the results returned by A and B, respectively. If A or B are honest, we will omit the corresponding annotation.

As with commitment, we can define correctness of this protocol.

Definition 1.4. Protocol 1.3 is *correct* relative to a commitment protocol (S_n, R_n) if for all $a, b \in \{R, P, S\}$,

$$\Pr[\textsf{RPS}(a, b) = (W(a, b), W(a, b))] = 1.$$

Proposition 1.5. *Protocol 1.3 is correct relative to any correct commitment protocol.*

[3] When n is unbound, we use this notation to indicate a family of pairs $\{(S_n, R_n) : n \in \mathbb{N}\}$; when n is bound it refers to the specific pair (S_n, R_n).

Proof. Fix a correct commitment protocol. Then

$$\begin{aligned}
&\Pr[\mathsf{RPS}(a,b) = (W(a,b), W(a,b))] \\
&= \Pr[W(a,b') = W(a,b) \text{ and } W(a',b) = W(a,b)] \\
&= \Pr[W(a,b') = W(a,b)] \cdot \Pr[W(a',b) = W(a,b)] \\
&\geq \Pr[b' = b] \cdot \Pr[a' = a] \\
&= \Pr[\mathsf{Rev}(b, c_b) = b] \cdot \Pr[\mathsf{Rev}(a, c_a) = a] \\
&= \Pr[\mathsf{Rev}(b, \mathsf{Com}(b)) = b] \cdot \Pr[\mathsf{Rev}(a, \mathsf{Com}(a)) = a] \\
&= 1. \qquad \square
\end{aligned}$$

Furthermore, at least in this case, it was easy to define notions of correctness that compose. Our task now is to define security of commitment and rock-paper-scissors such that, whenever a commitment scheme is secure, Protocol 1.3 is likewise secure.

Composition

We are centrally concerned in this thesis with the properties of security definitions under composition. By composition, we informally mean stitching-together various protocols to form some larger protocol[4]. We will say that a security definition *composes* if, whenever each of the smaller protocols is secure, the larger protocol is also secure.

There are two separate types of composition to consider. In *sequential* (or *vertical*) composition, only one of the small protocols is running at any given time. In the more general setting of *parallel* (or *horizontal*) composition, the parties involved may be engaged at various stages of many small protocols at once. There are several in-between notions to consider. For example, one can consider bounded-width composition, where the number of algorithms running in parallel is bounded by some function of the security parameter.

Protocol 1.3 is also somewhere in between sequential and parallel composition. There are two commitment protocols running simultaneously, but each phase of each protocol is running sequentially: while the parties are actively engaged in a specific commit or reveal phase, they are doing nothing else. As we will see when we discuss simulation security, working with multi-phase protocols like commitment adds a layer of complexity to many frameworks.

Game-Based Security

In *game-based* approaches to security, we define security by determining the winner of an abstract game. Here, we encode specific properties we want the algorithm to have, and say that an adversary

[4] The exact formal definition depends on the underlying model of computation, which we have deliberately left unspecified.

wins the game if they can break the property. In standard approaches to commitment[5], there are two desirable properties. *Hiding* means that the receiver should not learn anything about the message until the reveal phase. *Binding* means that the sender should not be able to trick the receiver into anything other than the committed message. Formally:

Definition 1.6. A commitment scheme (S_n, R_n) is *game-secure* if the following hold.

- *Hiding:* consider the following game against a family of adversaies R'_n.

  Game 1.7.

  1. R'_n outputs $m_0, m_1 \in \mathcal{M}$.
  2. A random bit $b \in \{0,1\}$ is chosen; m_b is given to S_n.
  3. S_n and R'_n participate in the commit phase.
  4. R'_n outputs a guess $b' \in \{0,1\}$. R'_n wins if $b' = b$.

  A commitment scheme is *hiding* if for any family of probabalistic polynomial-time algorithms R'_n,

  $$\Pr[R'_n \text{ wins Game 1.7}] \leq \frac{1}{2} + \mathsf{negl}(n).$$

  The idea is that R'_n participates in the commit phase for a randomly chosen message m_b, and then tries to guess b; they should be able to guess no more than negligibly[6] better than random.
- *Binding:* consider the following game against a family of adversaires S'_n.

  Game 1.8.

  1. S'_n outputs $m \in \mathcal{M}$.
  2. S'_n and R_n participate in the commit phase.
  3. A random bit $b \in \{0,1\}$ is chosen and given to S'_n.
  4. S'_n and R_n participate in the reveal phase.
  5. If $b = 0$, S'_n wins if R_n outputs m. If $b = 1$, S'_n wins if R_n outputs any $m' \notin \{m, \perp\}$.

  A commitment scheme is *binding* if for any family of probabalistic polynomial-time algorithms S'_n,

  $$\Pr[S'_n \text{ wins Game 1.8}] \leq \frac{1}{2} + \mathsf{negl}(n).$$

  The idea is that S'_n first makes a commitment, and then learns whether they want to lie to R_n; they should be able to do no more than negligibly better than guessing which of the two messages to commit to before learning the random bit.

[5] See [Gol01, Section 4.4.1].

[6] A function is *negligible*, written $\mathsf{negl}(n)$, if it is asymptotically smaller than any rational function in n.

Similarly, we can define security of Protocol 1.3 by having the adversary play against an honest party who moves according to some probability distribution.

Definition 1.9. Protocol 1.3 is *game-secure* relative to a commitment protocol (S_n, R_n) if the following hold for any probability distribution P on $\mathcal{M}$:

- *A-security*: For any family of PPT[7] algorithms A'_n,

  $$\Pr[\mathrm{pr}_2(\mathsf{RPS}_{A'_n}(a, b)) = 1] \le \max_{m \in \mathcal{M}} P(m) + \mathsf{negl}(n),$$

  where the randomness is over uniform choice of a and choice of b according to P.
- *B-security*: For any family of PPT algorithms B'_n,

  $$\Pr[\mathrm{pr}_1(\mathsf{RPS}^{B'_n}(a, b)) = -1] \le \max_{m \in \mathcal{M}} P(m) + \mathsf{negl}(n),$$

  where the randomness is over choice of $a \in \mathcal{M}$ according to P and uniform choice of b.

Observe that when the adversary controls A, we care only that B outputs a fair view of the game, and when the adversary controls B, we only care that A outputs a fair view of the game: we can never prevent the adversary from just outputting that they win.

Surprisingly, as we now show, it does not hold that Protocol 1.3 is game-secure relative to any game-secure commitment scheme.

Let $f : \{0,1\}^* \to \{0,1\}^*$ be a *one-way permutation*[8], i.e. an injective function which is easy to compute but hard to invert. Formally, this means that

- f is poly-time computable;
- for any family of PPT algorithms f'_n,

$$\Pr_{x \leftarrow \{0,1\}^n}[f(f'_n(f(x))) = f(x)] \le \mathsf{negl}(n).$$

Let $h : \{0,1\}^* \to \mathbb{Z}_3$ be a *ternary*[9] *hard-core predicate*[10] of f, i.e., a function which is easy to compute, but hard to compute "on f," in that

- h is poly-time computable;
- for any family of PPT algorithms h'_n,

$$\Pr_{x \leftarrow \{0,1\}^n}[h'_n(f(x)) = h(x)] \le \frac{1}{3} + \mathsf{negl}(n).$$

[7] Probabalistic polynomial-time, though this definition can be made relative to any class of adversarial algorithms.

[8] That such functions exist is a standard cryptographic assumption; see any introductory textbook, for example [KL14, Section 7.1].

[9] Typically, by hard-core predicate we mean a function which outputs a Boolean; we extend the notion for the case of rock-paper-scissors.

[10] If one-way permutations exist, then so too do one-way permutations with hard-core predicates [Yao82].

Protocol 1.10 ((f,h)-Commitment)**.**
Commit:

1. S_n gets $m \in \{R, P, S\}$ and interprets as an element of $\mathbb{Z}_3$.
2. S_n picks $x \in \{0,1\}^n$ uniformly at random.
3. S_n sends $(f(x), h(x) + m)$ to R_n.
4. S_n returns $(f(x), h(x) + m)$.
5. R_n returns $\top$.

Reveal:

1. S_n sends x to R_n.
2. R_n verifies the values of $f(x)$ and $h(x) + m$.
3. If the verification succeeds, R_n returns m; else they return $\perp$.

The idea is that secrecy is guaranteed by hardness of h, while bindingness is guaranteed by injectivity of f. Indeed:

Proposition 1.11. *Protocol 1.10 is game-secure.*

Proof. We need to show two things.

- *Hiding:* let R'_n be a family of PPT algorithms. Make a family h'_n which proceeds as follows:

  1. Receive input $y \in \{0,1\}^*$.
  2. Run R'_n to get $m_0, m_1 \in \{R, P, S\}$.
  3. Send $(y, 0)$ to R'_n, get back a guess b'.
  4. Output $-m_{b'}$.

  If $h(x) + m_b = 0$ for some b, then h'_n guesses $h(x)$ whenever R'_n guesses b. However, since $h(x)$ must look uniformly distributed, this occurs negligibly close to $\frac{2}{3}$ of the time. Hence since h'_n guesses right no more than negligibly more than $\frac{1}{3}$ of the time, R'_n guesses right no more than negligibly more than $\frac{1}{2}$ of the time.
- *Binding:* By injectivity of f, there is a unique x such that $f(x)$ is committed. Once R_n gets this unique x, they can deterministically verify $f(x)$ and $h(x) + m$, returning $\perp$ if this fails. □

This is an example of *proof by reduction*, a common technique in computer science. The idea is to reduce one problem into another by starting with a solution to the second and showing how to solve the first. If the first problem is known to be hard, in some precise sense, then by studying the structure of the reduction we can learn about the hardness of the second problem. In this case, we reduce the problem of computing the hard-core predicate into the problem of breaking the commitment scheme. To do this, we show that if we have some strategy R'_n to attack the commitment scheme, then we can use it to build a strategy h'_n to compute the hard-core predicate. We can then show hidingness of the commitment scheme using our assumption about the hardness of the hard-core predicate.

Regardless, we now observe that Protocol 1.3 is not game-secure relative to Protocol 1.10. The problem is that Protocol 1.10 is *malleable*, in that the receiver can compute the commitment of a message related to the message committed by the receiver. In particular, to win at rock-paper-scissors, Bob only needs to take Alice's commitment $(y, z) = (f(x), h(x) + m)$ and return $(y, z + 1) = (f(x), h(x) + m + 1)$, which is a valid commitment to $m + 1$. This does not violate secrecy, as the receiver still cannot learn anything about the message from this new computed commitment.

Realizing this, we could now add non-malleability to the list of security properties in Definition 1.6. However, this example reveals a broader issue with game-based security definitions: they rely on ad-hoc specifications of security properties, and we need to trust that we have thought hard enough to ensure that the properties we have specified are enough. For this reason, there are no general composition theorems for game-based security, and so large composite protocols of the kind used in real-world applications have to be proven secure by hand. In the next section, we outline a more principled approach.

Simulation-Based Security

In *simulation-based* approaches to security, we define security by comparing a protocol in the real world to an ideal world in which the desired resource is produced by a trusted black-box. We say a protocol is secure if a real-world adversary has no more influence on the outcome than they would in the ideal case. The proofs typically proceed by showing that any adversary in the real world can be "simulated" by an adversary in the ideal world.

It's easy to define the ideal world for commitment protocols, using a trusted party T:

Protocol 1.12 (Ideal Commitment)**.**
Commit:

1. $\overline{S}_n$ gets $m \in \mathcal{M}$.
2. $\overline{S}_n$ sends m to T.

Reveal:

1. T sends m to $\overline{R}_n$.
2. $\overline{R}_n$ outputs m.

Notation. As in Protocol 1.12, we will generally use overlines to denote parties engaged in an ideal protocol.

It remains to define the correct sense in which to compare attacks on the actual protocol to the ideal protocol.

Definition 1.13 (computataional indistinguishability)**.** Two families of random functions $\{X_n : A \to B\}$ and $\{Y_n : A \to B\}$ are *computationally indistinguishable* if, for any PPT algorithm D and $a \in A$, we have

$$|\Pr[D(X_n(a)) = 1] - \Pr[D(Y_n(a)) = 1]| \leq \mathsf{negl}(n).$$

In this case, we write $X \overset{c}{\equiv} Y$.

The idea is that D attempts to distinguish between X and Y, and must fail to do so all but negligibly often. If the combined output of some real protocol is computationally indistinguishable from the output of the ideal protocol, then this indicates that *no* adversary can distinguish between execution of the real protocol and the ideal protocol. As such, when we eventually prove composition theorems, we will be able to substitute the ideal protocol in place of the real protocol in order to prove security of some larger protocol.

Notation. We now let $\mathtt{Com}_S^R(m)$ denote the *pair* of outputs of S and R in the commit phase, with S getting input m, and so too for $\mathtt{Rev}_S^R(m,c)$. Recall that $\mathtt{Com}_{\overline{S}}^{\overline{R}}$ and $\mathtt{IRev}_{\overline{S}}^{\overline{R}}$ refer to the ideal protocol. Note that in this case, we omit the trusted party T, who always acts according to the protocol.

Combining the pair of outputs allows us to deal with issues like malleability, where (by hidingness) the receiver's output on its own is no different from that in the ideal protocol, but the pair of outputs may be correlated in the real protocol but not the ideal protocol.

Definition 1.14. A commitment protocol (S_n, R_n) is *simulation-secure* if the following hold:

- *Secure commit*: for any family of PPT algorithms R'_n, there exists a family of PPT algorithms $\overline{R}'_n$ such that for any $m \in \mathcal{M}$,

$$\mathtt{Com}_{S_n}^{R'_n}(m) \overset{c}{\equiv} \mathtt{Com}_{\overline{S}_n}^{\overline{R}'_n}(m).$$

- *Secure reveal*: for any family of PPT algorithms S'_n, there exists a family of PPT algorithms $\overline{S}'_n$ such that for any $m \in \mathcal{M}$ and $c \in \mathcal{C}$,

$$\mathtt{Rev}_{S'_n}^{R_n}(m,c) \overset{c}{\equiv} \mathtt{Rev}_{\overline{S}'_n}^{\overline{R}_n}(m,c).$$

To demonstrate the definition, we observe that Protocol 1.10 is *not* simulation-secure.

TODO: the definition is not quite right, so we can't finish this thought yet. We need to talk about the difficulty of proof in the paradigm, and also just how simulation proofs work.

Indeed, simulation security is sufficiently strong as to obtain a general sequential composition theorem:

Theorem 1.15 ([MR92]). *Let* $\{f_1, \cdots, f_{p(n)}\}$ *be a set of polynomially-many* two-party functionalities, *meaning functions* $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$. *Let* $\{\pi_i\}$ *be simulation-secure protocols for computing each of the* $\{f_i\}$. *Then the sequential composition of the* π_i*s is a simulation-secure protocol for computing the composite functionality* $f_{p(n)} \circ \cdots \circ f_1$.

We interpret the functionalities f_i as computational problems where two parties, given inputs $x, y \in \{0,1\}^*$, work together to compute outputs $\mathrm{pr}_1(f_i(x,y)), \mathrm{pr}_2(f_i(x,y))$. Such functionalities encompass a huge class of cryptographic protocols. The theorem says that if we know how to compute each f_i in a simulation-secure way—where the ideal world involves both parties giving their inputs to the trusted party and recieving their outputs back—then we can compute the composite simulation-securely, as well. For a more careful treatment of this theorem, and simulation security as a whole, see [Lin17].

It's important to observe that Theorem 1.15 works only for polynomially many functionalities. Recall that computational indistinguishability requires only a neglible difference between the real and ideal worlds, rather than requiring no difference. As a consequence, the theorem cannot hold for arbitrarily many functionalities, as the super-polynomial composition of negligible functions may fail to be negligible. In fact, computational indistinguishability is not even a real equivalence relation, as transitivity fails with more than polynomially many compositions. We make an attempt to illustrate the issue in Figure 1.1. Regardless, this is a fundamental issue: as long as our definitions of security are computational, we should not expect a super-polynomial composition of secure processes to remain secure.

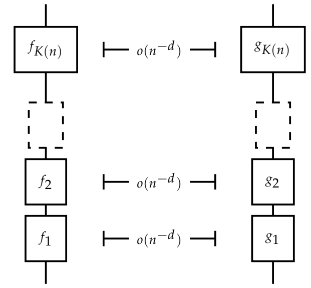

Figure 1.1: Even if $K(n)$ processes are computationally indistinguishable—meaning the adversary has only $o(n^{-d})$ chance to distinguish them—if $K(n)$ is super-polynomial, the composition of these processes may be distinguishable.

Regardless, while simulation security is generally a stronger guarantee than game security, it is not perfect. In 1996, Goldreich and Krawczyk constructed a protocol for *zero-knowledge proof* which is simulation-secure, but is not secure under parallel composition [GK96]. The construction is quite artificial, and we are not aware of any natural examples in which simulation security fails to compose. Nonetheless, for the purpose of provable security, we need a more powerful framework. There are several options, but the most popular by far is *universal composability*.

Universally Composable Cryptography

Universally composable cryptography, due to Ran Canetti, is a framework for writing composible proofs of security [Can00]. While a full treatment is far outside our scope, we give some intuition here, and conclude with a discussion of some drawbacks of the framework.

UC maintains the same basic idea as simulation security—the actual execution of the protocol is compared to some ideal world in which a trusted, unbounded third party is available. The main difference is how the ideal world is compared to the actual protocol. In simulation security, we use the straightforward notion of computational indistinguishability of outputs—at the end of the protocol, both parties output some value, and we compare the distribution of outputs in the ideal world to the real world. Even with commitment, we saw that this poses some difficulties when we want the protocol to satisfy some property midway through its execution.

In contrast, in universal composability, the adversary freely interacts with a separate (poly-time) algorithm, the *environment*. To be UC-secure, it should be impossible for the environment to distinguish between the ideal protocol and the real protocol *at any point during the execution*. This much stronger guarantee allows us to obtain a general parallel composition theorem, the intuition being that the environment could include artbirary other protocols, running simultaneously with the protocol under proof, so UC security implies security under even (polynomial-width) parallel composition.

On the other hand, this framework comes with onerous proof buderns. Whereas in the simulation paradigm, the ideal-world adversary must simulate interaction with the actual protocol, in the UC paradigm, the ideal-world adversary must also simulate the real-world adversary's interaction with the environment. Because this interaction is not constrained other than by computational bounds on the environment, even the simplest UC proofs require complex arguments about the flow of information through the protocol.

As a consequence, many cryptographers find UC proofs hard to trust. The framework is fundamentally tied to certain technical choices, such as the computational model of turing machines, meaning it requires modification to deal with, for example, quantum cryptography [HM03; Unr10]. The proofs tend to be extremely technical, often not even included in published work but instead left to supplemental material. The framework has been modified seven times since its introduction in 2000, including as recently as 2020 [Can20], and many of these versions have introduced incompatible technical changes. The cumulative result of all of this is that, while UC is undoubtedly the most powerful framework for composable proof in common use today, it is not as widespread as one might expect.

Conclusion

In this chapter, we considered three frameworks for security definitions in cryptography: game-based security, simulation-based security, and universal composability. Each of these frameworks exhibits tradoffs between complexity, composability, and ease of proof. While game-based proofs are simple to write, game-based definitions tend to be ad-hoc, making the definitions hard to trust and composition theorems impossible to obtain. On the other hand, universal composability obtains extremely powerful composition theorems, and leaves no room for error in writing definitions, but leads to complex proofs ridden with technical details. Simulation-based security lies somewhere in the middle, with good properties under sequential composition, but falls short of the generality of the UC framework.

This situation calls out for a new framework which takes advantage of powerful mathematical abstractions to simplify proofs, while maintaining the composition guarantees of universal composability. We believe that category theory is the right mathematical abstraction on which to base such a framework. In the next chapter, we will introduce the basic notions of this field, motivated by their potential relationship to cryptography.

2 *Category Theory*

The notion of a *category*, originally developed as an abstraction for certain ideas in pure mathematics, turns out to be the natural algebraic axiomatization of a collection of strongly typed, composable processes, such as functions in a strongly typed programming language. In this section, we will develop the basic theory of categories, prioritizing examples from computer science where possible[1].

Categories

Definition 2.1 (Category)**.** A *category* $\mathcal{C}$ consists of the following data:

- a collection[2] of objects, overloadingly also called $\mathcal{C}$;
- for each pair of objects $x, y \in \mathcal{C}$, a collection of *morphisms* $\mathcal{C}(x, y)$;
- for each object $x \in \mathcal{C}$, a designated *identity morphism* $x \xrightarrow{1_x} x$;
- for each pair of morphisms $x \xrightarrow{f} y \xrightarrow{g} z$, a designated *composite morphism* $x \xrightarrow{gf} z$.

This data must satisfy the following axioms:

- *unitality*: for any $x \xrightarrow{f} y$, $1_y f = f = f 1_x$;
- *associativity*: for any $x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w$, $(hg)f = h(gf)$.

$$\begin{tikzcd}
x \arrow[r, "f"] \arrow[rr, bend left, "gf"] & y \arrow[r, "g"] \arrow[rr, bend right, "hg"'] & z \arrow[r, "h"] & w
\end{tikzcd}$$

$$\begin{tikzcd}
x \arrow[r, "1_x"] \arrow[dr, "f"'] & x \arrow[d, "f"] \\
& y
\end{tikzcd}
\qquad
\begin{tikzcd}
x \arrow[r, "f"] \arrow[dr, "f"'] & y \arrow[d, "1_y"] \\
& y.
\end{tikzcd}$$

Figure 2.1: The category axioms.

Example 2.2 (Functional Programming Languages)**.** Consider some strongly-typed functional programming language $\mathcal{L}$, whose functions are never side-effecting. Then under very modest assumptions about $\mathcal{L}$, we can make a category $\mathcal{C}(\mathcal{L})$, as follows:

- the objects of $\mathcal{C}(\mathcal{L})$ are the types in $\mathcal{L}$;
- the morphisms $\mathcal{C}(\mathcal{L})(A, B)$ are the functions of type $A \to B$;
- the identities 1_A are the identity functions $\lambda(x : A).x$;
- composition of morphisms are the usual function composition.

If $\mathcal{L}$ is truly non-side-effecting, then it's straightforward to check that this construction does indeed satisfy the axioms of a category; see for instance [BW90, Section 2.2] to see the necessary assumptions spelled out.

[1] Basic texts on category theory include [Mac71] and [Rie17], while the connection to computer science is explored in [Pie91] and [BW90]. A more advanced treatment of the connection, especially applications to programming language theory, is [Jac99].

[2] We use the word *collection* for foundational reasons: in many important examples, the objects and morphisms do not form sets. We ignore such foundational issues here; they are discussed in [Mac71, Section 1.6].

Categories are also widespread in mathematics, as the following examples show.

Example 2.3 (Concrete Categories)**.** The following are all categories:

- Set is the category of sets and functions.
- Grp is the category of groups and group homomorphisms.
- Ring is the category of rings and ring homomorphisms.
- Top is the category of topological spaces and homeomorphisms.
- For any field $\Bbbk$, $\textsc{Vect}_{\Bbbk}$ is the category of vector spaces over $\Bbbk$ and linear transformations.

We call such categories, whose objects are structured sets and whose morphisms are structure-preserving set-functions, *concrete*. On the other hand, many categories look quite different.

Example 2.4. The following are also categories:

- The *empty category* has no objects and no morphisms.
- The *trivial category* has a single object and its identity morphism.
- Any group (or, more generally, monoid) can be thought of as a category with a single object, a morphism for every element, and composition given by the monoid multiplication.
- Any poset (or, more generally, preorder) $(P, \leq)$ can be thought of as a category whose objects are the elements of P, with a unique morphism $x \to y$ if and only if $x \leq y$. In this sense, composition is a "higher-dimensional" transitivity, and identities are higher-dimensional reflexivity.
- Associated to any directed graph is the *free category* on the graph, whose objects are nodes and whose morphisms are paths. In particular, the identities are just the empty paths, while composition concatenates two paths.
- There is a category whose objects are (roughly) multisets of molecules and whose morphisms are chemical reactions. See [BP17] for a formalization of this notion.

When working with categories, we often want to show that two complex composites equate. In this case, we prefer graphical notation to the more traditional symbolic equalities of Definition 2.1. The key idea is that such diagrams can be "pasted", allowing us to build up complex equalities from simpler ones.

Definition 2.5 (Commutative Diagram)**.** A diagram[3] *commutes* if, for any pair of paths through the diagram with the same start and end, the composite morphisms are equal.

Example 2.6. In this language, the axioms of Definition 2.1 are expressed by commutativity of the diagrams in Figure 2.1.

Functors

[3] The notion of a diagram can be made precise fairly easily; see [Rie17, Section 1.6].

Bibliography

[BK22] Anne Broadbent and Martti Karvonen. "Categorical composable cryptography". In: *Foundations of software science and computation structures*. Vol. 13242. Lecture Notes in Comput. Sci. Springer, Cham, 2022, pp. 161–183. ISBN: 9783030992538. DOI: 10.1007/978-3-030-99253-8_9. URL: https://doi.org/10.1007/978-3-030-99253-8_9.

[BP17] John C. Baez and Blake S. Pollard. "A compositional framework for reaction networks". In: *Reviews in Mathematical Physics* 29.09 (Sept. 2017), p. 1750028. DOI: 10.1142/s0129055x17500283. URL: https://doi.org/10.1142%2Fs0129055x17500283.

[BW90] Michael Barr and Charles Wells. *Category theory for computing science*. USA: Prentice-Hall, Inc., 1990. ISBN: 0131204866.

[Can00] Ran Canetti. *Universally Composable Security: A New Paradigm for Cryptographic Protocols*. Cryptology ePrint Archive, Paper 2000/067. https://eprint.iacr.org/2000/067. 2000. URL: https://eprint.iacr.org/2000/067.

[Can08] Ran Canetti. *Lecture 11*. Spring 2008. URL: https://www.cs.tau.ac.il/~canetti/f08-materials/scribe11.pdf.

[Can20] Ran Canetti. "Universally Composable Security". In: *J. ACM* 67.5 (Sept. 2020). ISSN: 0004-5411. DOI: 10.1145/3402457. URL: https://doi.org/10.1145/3402457.

[CF01] Ran Canetti and Marc Fischlin. "Universally Composable Commitments". In: *Advances in Cryptology — CRYPTO 2001*. Ed. by Joe Kilian. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 19–40. ISBN: 978-3-540-44647-7.

[CFS16] Bob Coecke, Tobias Fritz, and Robert W. Spekkens. "A mathematical theory of resources". In: *Information and Computation* 250 (2016). Quantum Physics and Logic, pp. 59–86. ISSN: 0890-5401. DOI: https://doi.org/10.1016/j.ic.2016.02.008. URL: https://www.sciencedirect.com/science/article/pii/S0890540116000353.

[GK96] Oded Goldreich and Hugo Krawczyk. “On the Composition of Zero-Knowledge Proof Systems”. In: *SIAM Journal on Computing* 25.1 (1996), pp. 169–192. DOI: 10.1137/S0097539791220688. eprint: https://doi.org/10.1137/S0097539791220688. URL: https://doi.org/10.1137/S0097539791220688.

[GL89] O. Goldreich and L. A. Levin. “A Hard-Core Predicate for All One-Way Functions”. In: *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*. STOC ’89. Seattle, Washington, USA: Association for Computing Machinery, 1989, pp. 25–32. ISBN: 0897913078. DOI: 10.1145/73007.73010. URL: https://doi.org/10.1145/73007.73010.

[Gol01] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2001. ISBN: 9780521791724.

[HM03] Dennis Hofheinz and Jörn Müller-Quade. *A Paradox of Quantum Universal Composability*. 2003. URL: https://www.quiprocone.org/Hot%20Topics%20posters/muellerquade_poster.pdf.

[Jac99] B. Jacobs. *Categorical Logic and Type Theory*. Studies in Logic and the Foundations of Mathematics 141. Amsterdam: North Holland, 1999.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC Cryptography and Network Security Series. Taylor & Francis, 2014. ISBN: 9781466570269.

[Lin17] Yehuda Lindell. “How to Simulate It—A Tutorial on the Simulation Proof Technique”. In: *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*. Springer International Publishing, 2017, pp. 277–346. ISBN: 9783319570488. DOI: 10.1007/978-3-319-57048-8_6.

[Mac71] Saunders MacLane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics, Vol. 5. New York: Springer-Verlag, 1971, pp. ix+262.

[MR92] Silvio Micali and Phillip Rogaway. “Secure Computation”. In: *Advances in Cryptology — CRYPTO ’91*. Ed. by Joan Feigenbaum. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 392–404. ISBN: 978-3-540-46766-3.

[Ped91] Torben Pryds Pedersen. “Non-interactive and information-theoretic secure verifiable secret sharing”. In: *Annual international cryptology conference*. Springer. 1991, pp. 129–140.

[Pie91] Benjamin C. Pierce. *Basic Category Theory for Computer Scientists*. The MIT Press, Aug. 1991. ISBN: 9780262288460. DOI: 10.7551/mitpress/1524.001.0001. URL: https://doi.org/10.7551/mitpress/1524.001.0001.

[PS10] Raphael Pass and Abhi Shelat. *A Course in Cryptography*. 2010. URL: https://www.cs.cornell.edu/courses/cs4830/2010fa/lecnotes.pdf.

[Rie17] Emily Riehl. *Category theory in context*. en. Courier Dover Publications, Mar. 2017.

[Ros21] Mike Rosulek. *The Joy of Cryptography*. 2021. URL: https://joyofcryptography.com.

[Tre09] Luca Trevisan. *Notes for Lecture 27*. Apr. 2009. URL: https://theory.stanford.edu/~trevisan/cs276/lecture27.pdf.

[Unr10] Dominique Unruh. "Universally Composable Quantum Multi-Party Computation". In: *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*. EUROCRYPT'10. French Riviera, France: Springer-Verlag, 2010, pp. 486–505. ISBN: 3642131891. DOI: 10.1007/978-3-642-13190-5_25. URL: https://doi.org/10.1007/978-3-642-13190-5_25.

[Yao82] Andrew C. Yao. "Theory and application of trapdoor functions". In: *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*. 1982, pp. 80–91. DOI: 10.1109/SFCS.1982.45.