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]

[1] Many more examples can be found in any introductory text on cryptography, such as [KL14; Ros21; Rap10].

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 $\perp$ or $\top$;
  - in the *reveal phase*, S_n gets $m \in \mathcal{M}$, while R_n gets $c \in \mathcal{C}$, returning $\perp$ or $m' \in \mathcal{M}$.

[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.

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

- 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.

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 $m \in \mathcal{M}$,

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

[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).

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 $\texttt{RPS}_A^B(a, b)$ 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 $a, b \in \{R, P, S\}$,

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

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

Proof. Fix a correct commitment protocol. Then

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

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.

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 wins the game if they can break the property. In standard approaches to commitment[4], 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:

[4] See [Golo1, Section 4.4.1].

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[5] better than random.

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

- *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.

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[6] algorithms A'_n,

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

where the randomness is over uniform choice of a and choice of b according to P.

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

- *B-security*: For any family of PPT algorithms B'_n,

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

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*[7], 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,

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

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

Let $h : \{0,1\}^* \to \mathbb{Z}_3$ be a *ternary*[8] *hard-core predicate*[9] 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,

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

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

[9] 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. $\mathcal{S}_n$ gets $m \in \mathcal{M}$.
2. $\mathcal{S}_n$ sends m to T.

Reveal:

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

Notation. As in Protocol 1.12, we will use curly letters to denote parties engaged in the 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 $\texttt{Com}_S^R(m)$ denote the *pair* of outputs of S and R in the commit phase, with S getting input m, and so too for $\texttt{Rev}_S^R(m, c)$. Recall that $\texttt{Com}_{\mathcal{S}}^{\mathcal{R}}$ and $\texttt{IRev}_{\mathcal{S}}^{\mathcal{R}}$ refer to the ideal 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.

2 Category Theory

Categories

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

- a collection[1] 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* $x \xrightarrow{gf} z$.

[1] 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].

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.

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

Example 2.2 (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 $\Bbbk$, $\text{Vect}_{\Bbbk}$ 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 2.3. 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.
- 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 2.4 (Commutative Diagram)**.** A diagram[2] *commutes* if, for any pair of paths through the diagram with the same start and end, the composite morphisms are equal.

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

Example 2.5. In this language, the axioms of Definition 2.1 are expressed by commutativity of the diagrams in Figure 2.1.

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.

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

[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.

[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.

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

[Rap10] "Abhi Shelat" "Raphael Pass". *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.

[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.