Cryptographic Secret Sharing

Girls Talk Math

Introduction

In this problem set, you will learn how to use math to split secrets into pieces. This is the cryptographic technique known as secret sharing and is the basis for many useful tools being deployed by companies today.

One last note about reading mathematical texts: it is very normal when reading math to read a passage or even a single sentence several times before understanding it properly. Also, never trust the author! Check every claim and calculation (time permitting). Take your time and never give up. Let's talk math!

Contents

Optional sections are marked with an asterisk (*).

1 Probability and Randomness

In mathematics, the word "random" has a more precise meaning than in everyday speech. Something is random if its outcome is governed by what's called a *probability distribution*. For example, the outcome c of a coin toss is not fixed, but takes on one of the values in the set $\{H, T\}$, where H means heads and T means tails. (This is called the *sample space* of c.) We write that $c \in \{H, T\}$, where the symbol $\in$ is read aloud as "is an element of" or simply "is in". The value of c is described by the distribution $\left\{\frac{1}{2}, \frac{1}{2}\right\}$ over the set $\{H, T\}$, meaning each outcome (H or T) happens with probability one-half.

Example 1.1 Here are some other everyday probability distributions and their sample spaces:

(a) The probability distribution of phone type used by Americans in 2021 over the sample space {Android, iPhone, Other} is $\{44.0\%^1, 46.9\%^2, 9.1\%\}$.

(b) The distribution over the outcomes of a dice roll $\{1, 2, 3, 4, 5, 6\}$ is $\{\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\}$.

Exercise 1.1 Write the sample space and probability distribution of each of the following random variables:

(a) Card drawn from a shuffled deck.

(b) Registration status of students at a 2000-person high school where all but 100 students register for classes every year.

A variable like c that represents an outcome or event is called a *random variable*, since it can take on one of several values based on an underlying probability distribution. A random variable is *uniformly random* if every possible value is equally likely. We call this probability distribution the *uniform distribution*. The variable c is uniformly random (or simply *uniform*) since H and

[1] https://www.statista.com/statistics/201182/forecast-of-smartphone-users-in-the-us/, https://www.statista.com/statistics/232786/forecast-of-andrioid-users-in-the-us/

[2] https://www.statista.com/statistics/236550/percentage-of-us-population-that-own-a-iphone-smartphone/

T occur with equal probability, namely both with probability $\frac{1}{2}$. We say such a variable is "chosen uniformly at random" (sometimes shortened to "chosen uniformly") and denote this with the symbol $\leftarrow_{\$}$. For instance, the coin toss example is written as $c \leftarrow_{\$} \{H, T\}$, meaning that c is sampled uniformly from the set $\{H, T\}$.

Exercise 1.2 Write "uniform" or "not uniform" for each of the following random variables:

(a) Drawing a card from the top of a shuffled deck.

(b) Drawing a card from the top of an ordered deck.

(c) The weather on a given day.

(d) The outcome of a dice roll.

(e) The birthday of the first person you encounter on the street.

Exercise 1.3 How would you represent a dice roll using the $\leftarrow_{\$}$ notation?

We use Pr to denote the probability of some event. For instance, the probability that c is heads is one-half. In mathematical notation this is written as

$$\Pr[c = H] = \frac{1}{2}$$

Similarly, c is tails with probability one-half:

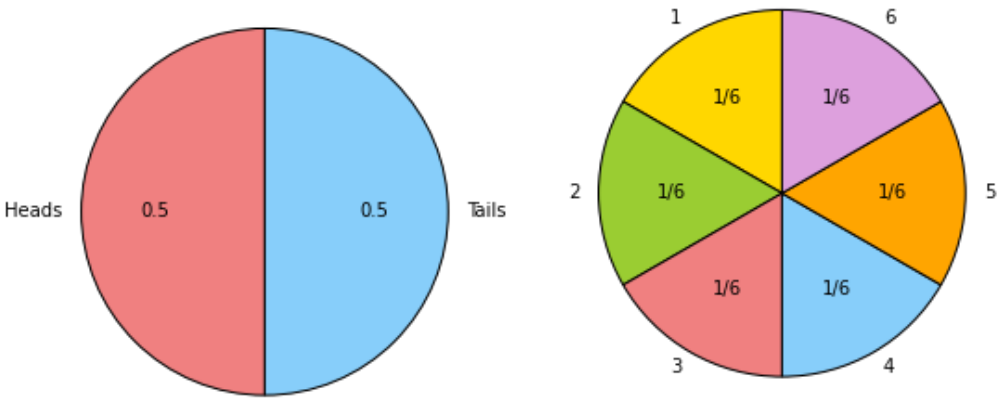
$$\Pr[c = T] = \frac{1}{2}$$

Given these two facts, what is the probability of getting heads *or* tails? Because these two events are *mutually exclusive* (that is, if one occurs, it means the other cannot possibly have occured, and vice versa), we can add their probabilities:

$$\begin{aligned} \Pr[c = H \text{ or } T] &= \Pr[c = H] + \Pr[c = T] \\ &= \frac{1}{2} + \frac{1}{2} = 1 \end{aligned}$$

Notice that these probabilities sum to 1. This is a common requirement of all probability distributions: the sum of their probabilities must be 1. This is the same as saying that *one of* the possible events in the distribution happens with 100% probability. If the sum was less than 1, it would mean that there's some other event that could happen instead, so the distribution is incomplete.

The pie charts below illustrate this idea. On the left is the probability distribution of a coin toss; on the right, the probability distribution of a dice roll. Notice that in both cases, the probabilities sum to 1.



When two events are not mutually exclusive, however, we can't add their probabilities. For example, what's the probability of rolling a multiple of 2 or a multiple of 3? These are not mutually exclusive events, since 6 is a multiple of *both* 2 and 3! Instead, we need to break this question into mutually exclusive events. We could consider the probability of rolling a multiple of two, and add this to the probability of rolling a 3: by doing this, we still cover all the possible outcomes (2, 3, 4, and 6) but broken into mutually exclusive events ($\{2, 4, 6\}$ or $\{3\}$) instead of overlapping events ($\{2, 4, 6\}$ or $\{3, 6\}$). Then

$$\begin{aligned}
&\Pr[d \text{ is a multiple of 2 or 3}] \\
&= \Pr[d \text{ is even}] + \Pr[d = 3] \\
&= \frac{1}{2} + \frac{1}{6} \\
&= \frac{4}{6} = \frac{2}{3}
\end{aligned}$$

where d is the random variable representing the outcome of the die roll.

Conditional probability is the probability that something occurs provided that another event occurred. This is written with the symbol |, which is read aloud as "given". So, $\Pr[A \mid B]$ is read as "the probability of A given B".

For example, it's more likely to be raining if the sky is cloudy than if it's clear. We can write this as

$$\Pr[\text{raining} \mid \text{cloudy sky}] > \Pr[\text{raining} \mid \text{clear sky}]$$

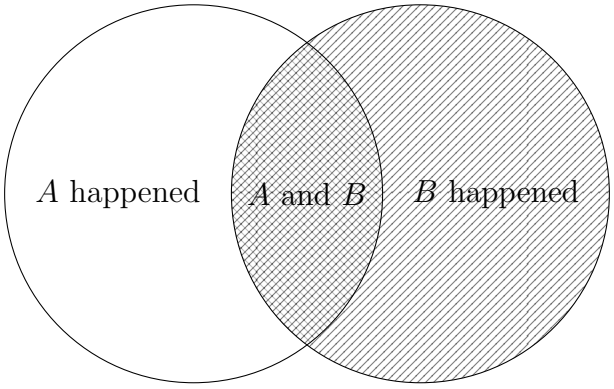
The following equality about conditional probability always holds[1]:

$$\Pr[A \mid B] = \frac{\Pr[A \text{ and } B]}{\Pr[B]} \tag{1a}$$

An alternate form is

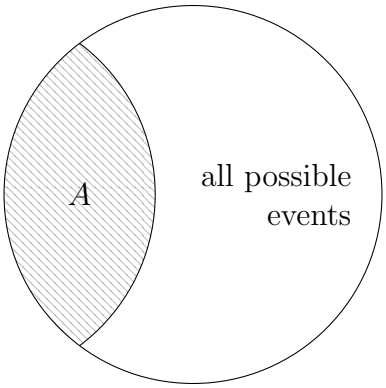
$$\Pr[A \text{ and } B] = \Pr[A \mid B] \cdot \Pr[B] \tag{1b}$$

Understanding why these equations are true can help us remember them. One way to see why they hold is with a Venn diagram:



Let's think through Equation 1a with this diagram in front of us. The event "A given B" means we know B happened, so now our entire realm of possibilities exists only in the circle on the right. Let's zoom in on that circle:

[1]With the caveat is that we cannot condition on a zero-probability event (that is, we require $\Pr[B] \neq 0$) since we cannot divide by zero.



The cases in which A happens next are in the little slice on the left. We want to know the probability A happening given our current information, which is simply the fraction of possible events covered by this area. Zooming back out, that's exactly $\Pr[A \text{ and } B]$ divided by $\Pr[B]$!

The difference is the reference frame: in the first diagram, we are in a world in which we make no assumptions: all possibilities are open. In the second, we are using the extra information that B happened to limit the world of possibilities. Let's see some concrete examples next.

Example 1.2 Consider the following sample space, where each number is chosen with equal probability. Throughout the example, let $B = \{3, 4, 7, 8, 11, 12, 15, 16\}$ (the shaded cells).

B

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Notice that $\Pr[B] = \frac{8}{16} = \frac{1}{2}$.

(a) Suppose $A = \{9, 10\}$:

			B	
	1	2	3	4
	5	6	7	8
A	9	10	11	12
	13	14	15	16

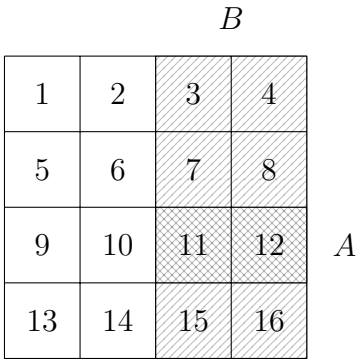
Since A and B don't overlap, $\Pr[A \text{ and } B] = 0$. Then, using Equation 1a, $\Pr[A \mid B] = \Pr[A \text{ and } B] \div \Pr[B] = 0 \div \frac{1}{2} = 0$. Using the visual intuition we gained with the Venn diagrams, we can justify this by saying that A occupies 0 of the space of B.

(b) Now let $A = \{10, 11\}$:

			B	
	1	2	3	4
	5	6	7	8
A	9	10	11	12
	13	14	15	16

Now exactly one number, 11, is in both A and B, so $\Pr[A \text{ and } B] = \frac{1}{16}$. Thus $\Pr[A \mid B] = \frac{1}{16} \div \frac{1}{2} = \frac{1}{8}$. Visually, we notice that A in fact occupies one-eighth of B's space.

(c) Lastly, let $A = \{11, 12\}$:



Now $\Pr[A \text{ and } B] = \frac{2}{16} = \frac{1}{8}$, so $\Pr[A \mid B] = \frac{1}{8} \div \frac{1}{2} = \frac{1}{4}$. Again, this makes sense visually as well, since A occupies one-quarter of B.

Example 1.3 Suppose your friend rolls a die without showing you the result. You know the probability that they rolled a 3 is $\frac{1}{6}$. Now suppose they tell you, "I rolled an odd number." Using this additional information, you can update your estimate of the probability that the die roll is a 3 to $\frac{1}{3}$, since you know the die landed with equal probability on 1, 3, or 5, and definitely didn't land on 2, 4, or 6.

Written in the language of conditional probability, this is

$$\begin{aligned}\Pr[d = 3 \mid d \text{ is odd}] &= \frac{\Pr[d = 3 \text{ and } d \text{ is odd}]}{\Pr[d \text{ is odd}]} \\ &= \frac{\Pr[d = 3]}{\Pr[d \text{ is odd}]} \\ &= \frac{1}{6} \div \frac{1}{2} = \frac{2}{6} = \frac{1}{3}\end{aligned}$$

Exercise 1.4 Repeat Example 1.2, calculating $\Pr[B \mid A]$ instead.

In some cases, such as Example 1.2b, we have $\Pr[A \mid B] = \Pr[A]$. Intuitively, this means that the probability of A is the same regardless of whether or not B happened. For example, a good coin toss should be independent of

the coin used for the toss or the person doing the toss. In these cases, we say that A and B are *independent* events, and their corresponding random variables are independent. Then Equation 1b simplifies to

$$\Pr[A \text{ and } B] = \Pr[A] \cdot \Pr[B] \tag{2}$$

This means that for independet random variables, we can just multiply the probabilities of two events together to get the probability that both events happen simultaneously.

Example 1.4 Suppose I roll a die and flip a coin. What's the probability that I roll a 1 *and* get heads?

Our first instinct might be to list out all of the possible combinations of events; for each outcome of the roll, there is a tails and a heads option:

$$1H, 1T, 2H, 2T, 3H, 3T, 4H, 4T, 5H, 5T, 6H, 6T$$

Each of these outcomes occurs with equal probability, and there are twelve of them, so rolling a 1 and getting heads ($1H$) occurs with probability $\frac{1}{12}$.

If we realize that the dice and coin outcomes are independent, however, we can get to this answer in a simpler way. Looking at Equation 2, we see that we can multiply the probabilities of the two events together the get the probability of *both* events occuring. Let d be the random variable representing the outcome of the die roll and c be the random variable representing the outcome of the coin toss. Then

$$\begin{aligned} &\Pr[d = 1 \text{ and } c = H] \\ =&\Pr[d = 1] \cdot \Pr[c = H] \\ =&\frac{1}{6} \cdot \frac{1}{2} = \frac{1}{12} \end{aligned}$$

Exercise 1.5 Suppose I flip two coins. What's the probability of both coins landing on heads?

Let's put everything we've learned so far together and work through some examples. Remember, for the **or** of two mutually exclusive events, we add; for the **and** of two independent events, we multiply. In the latter case, if the events aren't mutually exclusive, we use conditional probability (Equations 1a and 1b).

Example 1.5 Say I flip a fair coin twice. What's the probability of getting the same result both times?

There are two ways of getting the same result on both flips: we either get heads twice (HH) or tails twice (TT):

$$\Pr[HH] + \Pr[TT]$$

Because subsequent coin flips are independent of each other, this is equal to

$$\Pr[H] \cdot \Pr[H] + \Pr[T] \cdot \Pr[T]$$

Since the coin is fair, each outcome (heads or tails) happens with equal probability, so this equals

$$\begin{aligned} &\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} \\ =&\frac{1}{4} + \frac{1}{4} \\ =&\frac{1}{2} \end{aligned}$$

Exercise 1.6 Suppose I only flip a coin again if my first coin toss came out tails. What's the probability of getting one heads outcome?

Exercise 1.7 The following questions deal with rolling a 6-sided die.

(a) What's the probability of rolling an even number?

(b) What's the probability of rolling a 1 followed immediately by a 2?

(c) If I roll twice, what's the probability that my second roll will be higher than my first?

(d) Suppose that when I roll a 1, I get to reroll and use the second number instead. What's the probability of rolling a 5 or higher?

When two random variables have the same underlying probability distribution, we say they are *perfectly indistinguishable*. For example, if $d_1 \in \{1, 2, 3, 4, 5, 6\}$ is the random variable representing the outcome of rolling a 6-sided die, the probability distribution of d_1 is

$$\left\{\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right\}$$

where each probability is the probability that c takes on the corresponding value in the sample space.

Let's compare this to rolling a 12-sided die and dividing the result by 2, rounding up to the nearest whole number. That is, if we roll a 1 or 2 on the die, the result is 1, if we roll a 3 or 4, the result is 2, and so on. Then the probability distribution over this random variable (call it d_2) is over the sample space $\{1, 2, 3, 4, 5, 6\}$, with values

$$\left\{\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right\}$$

The sample space is the same in both cases, and each outcome in that sample space occurs with the same probability in both distributions. So they're the same distribution! Therefore d_1 and d_2 are perfectly indistinguishable.

Exercise 1.8 Write "perfectly indistinguishable" or "not perfectly indistinguishable" for each pair of probability distributions D_1, D_2:

(a) D_1: Rolling an even or an odd number on a 6-sided die, D_2: getting heads or tails when you flip a coin

(b) D_1: Drawing a random card from a shuffled deck, D_2: rolling a 6-sided die and flipping 3 coins

(c) D_1: Whether or not you get at least one heads when flipping two coins, D_2: whether or not you draw a club from a shuffled deck.

1.1 Randomness in Cryptography

As we'll see, randomness plays an extremely important role in modern cryptographic schemes. A cryptographic *scheme* is a well-defined procedure for accomplishing some blueprint with desirable properties (called a *cryptographic primitive*). For example, the Caesar cipher is an encryption scheme[2], meaning it is a procedure for accomplishing the cryptographic primitive of encryption. In this packet, you'll learn about several *secret sharing* schemes.

When a scheme's security rests only on the randomness used in it, and not on solving a difficult or very long problem, we say that scheme is *information-theoretically secure*. This is different from, for example, the schemes in the RSA cryptography packet, whose security depends on the assumed difficulty of factoring[3].

[2]Note, however, that it is *not* a secure encryption scheme! You can learn more about how the Caesar cipher works and its weakness here: `https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/caesar-cipher`.

[3]When a scheme is protected by how hard it is to compute something, it is said to be *computationally secure*.

2 Secret Sharing

Secret sharing is a way to "split" a secret value (call it s for "secret") into pieces, called *shares*. Two useful properties a secret sharing scheme might have are *correctness* and *privacy*. Informally, correctness means that if we put shares back together, we get back the original secret; privacy says that each share by itself reveals nothing about the secret s.

These properties are important for practical uses of secret sharing. For example, using secret sharing, we can distribute shares among a large set of people so that no one knows the secret but some subset of them can recover the secret if they pool their information.

We'll be a little more rigorous about these definitions soon, but first, let's see an example.

2.1 A simple secret sharing

Here is a simple scheme for sharing integers using nothing but addition and subtraction.

$\mathsf{Share}(s)$	$\mathsf{Rec}(s_1, s_2)$
$s_1 \leftarrow\!\!\$\ \{0, \ldots, 2^\lambda - 1\}$	**return** $s_1 + s_2 \mod 2^\lambda$
$s_2 \leftarrow s - s_1 \mod 2^\lambda$	
return (s_1, s_2)	

Figure 1: Additive secret sharing scheme

In a future iteration, maybe introduce the naïve scheme (without modulus) and point out the issues to slowly adjust the scheme.

Let's go through the notation together. First, on the left, we see that the Share algorithm is being defined. (An *algorithm* is simply a procedure.) The parentheses after the name tell us that it takes an input s, which in this context is the secret to be shared. The first line tells us to sample[4] an element from the set $\{0, \ldots, 2^\lambda - 1\}$ and call it s_1. The dots in the set are shorthand for all the numbers in between. For instance, when $\lambda = 10$, we sample s_1 uniformly from the set $\{0, 1, 2, 3, \ldots, 1021, 1022, 1023\}$. The value of λ would be chosen beforehand by the person or people running Share and Rec.

[4] We introduced the symbol $\leftarrow\!\!\$$ in Section 1.

The next line says to compute s_2 as the difference $s - s_1$ *modulo* the number 2^λ. This means that we "wrap around" to $2^\lambda - 1$ if the difference goes below zero: -1 is $2^\lambda - 1$ instead, -2 is $2^\lambda - 2$, and so on. For example, if $s = 3$ and $s_1 = 5$, $s - s_1 = 3 - 5 = -2$. With $\lambda = 10$, we instead "wrap around" to $2^\lambda - 2 = 1024 - 2 = 1022$. This modulo operation has important implications for privacy, as we'll talk about soon.

We're almost done working through the Share algorithm. The next and final line says to return (output) a pair of numbers: the random number s_1 and the difference s_2.

So Share converts a secret s into a two shares. Okay, so how can two people, each with one of the shares, get back the original secret? That's what the right side of Figure 1 tells us. The reconstruction algorithm Rec takes two integers (call them s_1 and s_2) and returns their sum, again modulo 2^λ. The "wrapping" goes both ways, which in this case means that if the sum is greater than $2^\lambda - 1$, we wrap back around to 0: 2^λ becomes 0, $2^\lambda + 1$ becomes 1, and so on.

> **Example 2.1** Let $\lambda = 10$. We'll work through an execution of the Share and Rec algorithms with $s = 4$ to get more comfortable with the modulus operation.
>
> First, say we sampled $s_1 = 10$. Then $s_2 = 4 - 10 = -6$, which modulo 2^λ becomes $2^\lambda - 6$. Since $\lambda = 10$, $s_2 = 1024 - 6 = 1018$. So Share returns $(10, 1018)$.
>
> Now let's plug these shares into the reconstruction algorithm: $\mathsf{Rec}(10, 1018)$ returns $10 + 1018 \mod 2^\lambda = 1028 \mod 2^\lambda = 1028 - 1024 = 4$. That's exactly the original secret.

For values of s in $\{0, \ldots, 2^\lambda - 1\}$, If Rec receives two shares that were produced by the Share algorithm, it will return $s_1 + (s - s_1) \mod 2^\lambda = s$. This means the scheme has correctness! If s is not in that set, though, we run into some problems: since the modulus operation will always reduce to a number in the range $\{0, \ldots, 2^\lambda - 1\}$, Rec will always output a number in that range. This means that if s is not one of these numbers, Rec will not output it, violating correctness. We'll have to limit ourselves to sharing only numbers in that set, which just means we need to pick λ in a way that will include all the numbers we expect to share.

What about privacy?

It turns out the scheme is private as well: someone who sees only one of the two shares learns nothing about the secret. This is pretty straightforward if the one share you see is s_1: we picked this value randomly (remember this

means we picked it independently of s), so it has nothing to do with s. This is the case with the other share as well. If we're given a number s_2 calculated as $s - s_1 \pmod{2^\lambda}$, we don't know the other share s_1. With s and s_1 both in $\{0, \ldots, 2^\lambda - 1\}$, s_2 will range anywhere from 0 to $2^\lambda - 1$. Each of these values of s_2 could have occurred for any value of s. For instance, if $s_2 = 0$, it could be that $s = 0$ and $s_1 = 0$; or maybe $s = 1$ and $s_1 = 1$; and so on, as long as $s = s_1$ (but s_1 is unknown!). Another way to think about this is that we don't know s_1, so we can't undo the subtraction and recover s. In this case we say that s_1 "masks" s.

These arguments for correctness and privacy are not very rigorous. We'll take a look at how cryptographers prove these properties in Sections 2.2 and 2.3.

Exercise 2.1 Pick your favorite number and secret share it using the Share algorithm defined above, with $\lambda = 10$. (Hint: $2^{10} = 1024$.) (Repeat this exercise until you are comfortable with this secret sharing scheme.)

Exercise 2.2 Again using $\lambda = 10$, what is

(a) $\mathsf{Rec}(2, 6)$?

(b) $\mathsf{Rec}(4, 1)$?

(c) $\mathsf{Rec}(10, 2)$?

(d) $\mathsf{Rec}(115, 921)$?

(e) $\mathsf{Rec}(559, 480)$?

Exercise 2.3 Can you adapt this additive secret sharing scheme to output 3 shares instead of 2? How does this change the reconstruction algorithm?

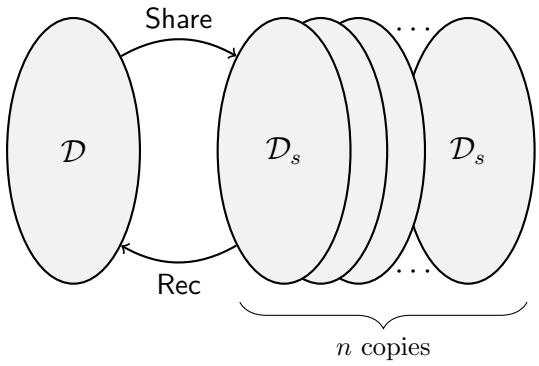
2.2 Formal Definitions*

We'll now formally define what a secret sharing scheme is and the properties such a scheme might have, using the standard notation in cryptography. First, we'll define what a secret sharing scheme does without giving implementation details (there could be multiple ways of achieving the same thing, after all).

Definition 1 (Secret sharing scheme)**.** *Let $\mathcal{D}$ be the input domain and $\mathcal{D}_s$ be the share domain. A secret sharing scheme is a pair of efficient*[5] *algorithms* $(\mathsf{Share}, \mathsf{Rec})$ *and an associated natural number n such that*

- Share *takes as input a secret $s \in \mathcal{D}$ and outputs n shares in $\mathcal{D}_s$.*
- Rec *takes as input m shares $s_1, \ldots, s_m \in D_s$ and outputs some value $y \in \mathcal{D}$ or a special symbol $\perp$ indicating failure. (If $m \neq n$, it outputs $\perp$.)*

Here's a visual representation:



2.2.1 Correctness

Definition 2 (Correctness)**.** *A secret sharing scheme is* correct *if for all $s \in \mathcal{D}$,* $\mathsf{Rec}(\mathsf{Share}(s)) = s$.

Example 2.2 Consider the following secret sharing scheme:

$\mathsf{Share}(s)$	$\mathsf{Rec}(s_1, s_2)$
$s_1 \leftarrow\!\!\$\, \{1, \ldots, 2^\lambda\}$	**return** $s_1 + s_2$
return $(s_1, s + s_1)$	

[5] running in a reasonable time, e.g. no more than a few seconds

This scheme meets Definition 1 with $n = 2$. The input and share domains are both the integers, Share correctly outputs two integers, and Rec takes two integers and outputs another integer. (If we give Rec any other number of integers, it returns $\perp$; for simplicity, we'll leave this detail out of the algorithms in this section.)

Does this scheme have correctness? Answer in your head before reading on.

If we compare this to the simple scheme in Figure 1, the answer is pretty clear: no, this does not meet the correctness requirement. A little bit of arithmetic confirms this: $\mathsf{Rec}(\mathsf{Share}(s)) = \mathsf{Rec}(s_1, s + s_1) = s_1 + (s + s_1) = s + 2s_1$, which is not equal to s (s_1 is never 0 since it is always greater than or equal to 1).

Example 2.3 Let's look at another scheme:

$\mathsf{Share}(s)$	$\mathsf{Rec}(s_1, s_2)$
return $(3, \lfloor s/3 \rfloor)$	**return** $s_1 \cdot s_2$

The $\lfloor \cdot \rfloor$ notation is the "floor" operation, which rounds a decimal number down to the integer below it. For example, $\lfloor 3.725 \rfloor = 3$.

Again, this is a secret sharing scheme according to Definition 1 with both $\mathcal{D}$ and $\mathcal{D}_s$ being the set of integers. But is it correct? Answer to yourself again before reading on.

This example is a little trickier because it works in some cases – but not all! When $s = 15$,

$$\mathsf{Rec}(\mathsf{Share}(15)) = \mathsf{Rec}(3, 5) = 3 \cdot 5 = 15,$$

which is correct. But in the case of $s = 14$,

$$\mathsf{Rec}(\mathsf{Share}(14)) = \mathsf{Rec}(3, 4) = 3 \cdot 4 = 12,$$

which is not equal to 14. So this scheme isn't always correct, and since the definition of correctness is all or nothing[6], the scheme doesn't have correctness.

[6] we said $\mathsf{Rec}(\mathsf{Share}(s)) = s$ must hold for *all* $s \in \mathcal{D}$. But 14 was an integer for which the equality didn't hold.

Example 2.4 So always setting s_1 to 3 works when sharing some numbers but not others. Let's go back to picking s_1 randomly:

$\mathsf{Share}(s)$	$\mathsf{Rec}(s_1, s_2)$
$s_1 \leftarrow\!\!\$\ \{1, \ldots, 2^\lambda\}$	**return** $s_1 \cdot s_2$
return $(s_1, \lfloor s/s_1 \rfloor)$	

This still meets Definition 1 with the same input and share domains. Is the adjusted scheme correct now?

What happens if we share the number 14? First, we pick s_1 randomly. Say we happen to choose 7. Then

$$\mathsf{Rec}(\mathsf{Share}(14)) = \mathsf{Rec}(7, 2) = 14,$$

which is correct. But what if we had randomly chosen s_1 to be 5? Then

$$\mathsf{Rec}(\mathsf{Share}(14)) = \mathsf{Rec}(5, 2) = 10,$$

which is not 14!

When $s = 37$, a similar thing happens: the scheme is correct for some choices of s_1, but not others:

$$\begin{aligned}
s_1 = 1: &\quad \mathsf{Rec}(\mathsf{Share}(37)) = \mathsf{Rec}(1, 37) = 37 = 37 \ \checkmark \\
s_1 = 2: &\quad \mathsf{Rec}(\mathsf{Share}(37)) = \mathsf{Rec}(2, 18) = 36 \neq 37 \ \text{✗} \\
s_1 = 3: &\quad \mathsf{Rec}(\mathsf{Share}(37)) = \mathsf{Rec}(3, 12) = 36 \neq 37 \ \text{✗} \\
\vdots &\qquad\qquad\qquad \vdots
\end{aligned}$$

So even though this scheme *can* work for all $s \in \mathcal{D}$, correctness will only hold for certain choices of s_1, and we have no control over s_1 because it is chosen randomly. So this scheme is still not correct.

Even though all the examples above didn't meet correctness, remember that correct secret sharing schemes do exist. For example, we already argued informally that the simple additive scheme from Figure 1 meets correctness by observing that $\mathsf{Rec}(\mathsf{Share}(s)) = \mathsf{Rec}(s_1, s - s_1) = s_1 + (s - s_1) = s + (s_1 - s_1) = s$.

Now that we understand correctness, let's look at privacy.

2.2.2 Privacy

covers only 1 corrupted party; change this to cover up to $n-1$ corruptions. In cryptography, security properties like privacy are defined using what are called "games". A game is a challenge in which an attacker (called the *adversary* and usually denoted by the curly letter $\mathcal{A}$) is given some information and tries to break the security property of the scheme. $\mathcal{A}$ "wins" the game if it can give an answer that proves it broke the security property of the scheme.

For example, in the case of privacy for a secret sharing scheme, we give the attacker a share and ask it to give us some information about the secret it came from. This should be almost impossible if the scheme is private.

More specifically, the *secret sharing privacy game* captures the following scenario. Alice wants to share some secret s so that she can distribute its pieces to her friends. Maybe s is the combination to her safe holding all her savings, and if something happens to her she wants her friends to be able to access them and distribute them according to her will. Unbeknownst to her, her friend Eve wants to go against Alice's wishes and get into the safe herself to steal Alice's money.

Let's think about the worst case, in which Eve is very powerful: she knows that the safe is locked with equal probability by one of two combinations x_0 or x_1; even worse, she was able to choose x_0 and x_1 by somehow influencing Alice as she chose her combination; and she's even even managed to figure out how to peek at some of the other friends' shares (at most $n-1$, including herself). Once Alice has shared her combination among her friends, Eve will see at most $n-1$ shares (her choice which ones) which are all the result of either $\mathsf{Share}(x_0)$ or $\mathsf{Share}(x_1)$.

What sort of protection does Alice need? Even with all the information known to Eve (her knowledge of x_0, x_1, i, $n-1$ shares), we don't want her to figure out whether she got a share of x_0 or x_1, because if she did, she'd be able to unlock the safe!

Here is the formal privacy game for any secret sharing scheme $\mathcal{S} = (\mathsf{Share}, \mathsf{Rec})$. (You can think of $\mathcal{A}$ as Eve.)

1. The adversary $\mathcal{A}$ picks two values x_0, x_1 and a set I of size at most $n-1$ of numbers between 1 and n.

2. The game flips a coin to randomly choose one of the two values x_0, x_1 to share. This is usually written as picking a random value b from the set $\{0, 1\}$.

3. Now the game runs the Share algorithm on this randomly chosen value x_b to get shares $s_1, \ldots, s_m$. It gives s_i for all $i \in I$ to $\mathcal{A}$.

4. $\mathcal{A}$ tries to guess which of the two values x_0, x_1 were shared. More specifically, it outputs a guess $b' \in \{0, 1\}$.

5. If $b = b'$, we say the result of the game is 1 (to signify **true** or **success**), in which case we say that $\mathcal{A}$ "wins" the game; otherwise, it's 0.

In cryptography, these games are generally written much more compactly by using symbols. The privacy game above would be written as follows (without the comments, which are there to explain the notation):

SS-priv$_{\mathcal{A},\mathcal{S}}$

$(x_0, x_1, I) \leftarrow \mathcal{A}$ // get input values from $\mathcal{A}$
$b \leftarrow\!\!\$\ \{0, 1\}$ // decide randomly which value to share
$s_1, \ldots, s_m \leftarrow \mathsf{Share}(x_b)$ // share the chosen value
$b' \leftarrow \mathcal{A}(\{s_i\}_{i \in I})$ // $\mathcal{A}$ uses the shares indicated by I to guess b'
return $b = b'$ // return 1 if $b = b'$ and 0 otherwise

Figure 2: The secret sharing privacy game.

Where did this definition of the game come from, you might ask? The simple answer is that usually the person who invents a new cryptographic primitive (what we call a general idea like secret sharing or encryption, as opposed to a specific algorithm for actually achieving what it describes) also gives definitions for its potential properties. This includes defining security games. (Sometimes, others come along later and describe new properties for an existing primitive; in this case, they might describe a game for the new property.)

If you look at many security games, though, you'll see that they are usually pretty similar to each other. This is because it's useful for a new game to be easy to work with, since it makes other people more likely to build on that work. If a new game is similar to an already existing game, people who are familiar with the previous game can understand the new game quickly and prove things about a new scheme more easily.

***Bonus Exercise* 2.4** With a partner, play through the privacy game a couple times using the additive secret sharing scheme. One of you should take on the role of the game while the other acts as the adversary. If you're playing the part of the game, make sure

you use something truly random, like a coin flip or the Python command `random.randint(0,1)`[7], to pick your bit b. After you've done this a couple of times, switch roles and repeat.

How successful was the adversary? If you were the adversary, what was difficult about your role? What was the key part of the scheme that ensured privacy?

Now that we have an idea of why the game is difficult to win consistently, let's rigorously define privacy by specifying how often the adversary should be able to win:

Definition 3 (secret sharing privacy)**.** *A secret sharing scheme* $\mathcal{S} = (\mathsf{Share}, \mathsf{Rec})$ *is* private *if, for all adversaries* $\mathcal{A}$*,*

$$\Pr[\text{SS-priv}_{\mathcal{A},\mathcal{S}}(n) = 1] - \frac{1}{2}$$

is small[8]*. This quantity is called* $\mathcal{A}$*'s* advantage.

What does this actually mean? If $\mathcal{A}$'s advantage is small, it means that $\Pr[\text{SS-priv}_{\mathcal{A},\mathcal{S}}(n) = 1]$ is very close to one-half. In other words, no matter what adversary $\mathcal{A}$ we are dealing with, the probability that it can win privacy game (i.e., the game outputs 1) should be very close to one-half, which is what it should be if $\mathcal{A}$ were to randomly guess which value was shared.

This might sound very different from the informal definition from Section 2: each share by itself reveals nothing about the secret. The game instead says that $\mathcal{A}$ can't tell the difference between two different secrets. But if you think about this a little more, you can see that they are related: if $\mathcal{A}$ can tell the difference between two secrets based on a single share, it means the share gave away some information about the secret it came from.

Let's work through some examples. For the remainder of this section, you can assume that all schemes are secret sharing schemes that meet Definition 1.

Example 2.5 Here's a secret sharing scheme for sharing integers 1 to 99 (i.e., $\mathcal{D} = \{1, \ldots, 99\}$) among two parties:

[7] Be sure to `import random` first.

[8] In cryptography, this usually means bounded by a *negligible* function. For the purposes of this packet, "small" means very close to 0, for example $\frac{1}{2^\lambda}$, which for $\lambda = 10$ is $\frac{1}{1024}$. Since this quantity ranges from 0 to $\frac{1}{2}$, a value like $\frac{1}{4}$ or $\frac{1}{8}$ is not small.

$\mathsf{Share}(s)$	$\mathsf{Rec}(s_1, s_2)$
$s_1 =$ the tens digit of s	**return** $s_1 \| \| s_2$
$s_2 =$ the ones digit of s	
return (s_1, s_2)	

where $||$ means concatenation (for example, $1||2 = 12$). Do you think this scheme is private?

Let's put ourselves in the adversary's shoes. We want to win the privacy game. First, we get to pick two numbers $x_0, x_1 \in \mathcal{D}$ and an index i. Let's say $x_0 = 18$ and $x_1 = 24$. We'll let $i = 2$ (it doesn't actually matter what i is in this case). Now the game picks a random b unknown to us and shares x_b. It'll give us the second share, s_2.

What are the possibilities for s_2? If $b = 0$, the game runs $\mathsf{Share}(18)$, which outputs $(1, 8)$. Then $s_2 = 8$. If $b = 1$, on the other hand, the game will run $\mathsf{Share}(24)$ to get $(2, 4)$ and $s_2 = 4$. Say we get 8. Then we'll guess $b' = 0$. If we get 4 from the game, we'll guess $b' = 1$. Because of how the secret sharing scheme works ("by definition"), we'll always guess correctly, and $b = b'$ with probability 1. This means that our advantage is $\frac{1}{2}$, which is not small! Therefore, this scheme is not private.

You might say, well, duh! This was obvious from the informal definition of privacy: if we're given a digit of the secret, we're clearly learning something about the secret! Looking at the adversary's advantage in the game, however, makes more sense mathematically. In the case of a much more complicated scheme, it might be unclear what "learning something about the secret" really means.

Another thing to notice is that the adversary's choice of x_0 and x_1 is important. A less clever adversary might choose x_0 and x_1 with the same tens place and then ask for the $i = 1$ share (the tens digit). If it does this, it can't tell the shares apart, since they'll have the same value. Our definition is formulated to get around issues like this: to show a scheme is secure, we need to show that there is *no* adversary with a not-small advantage. This means we only need to think about what the cleverest way to break a scheme would be, and show that even that is impossible.

This is also a place where problems with cryptographic proofs of security can arise. There are examples of cryptographers publishing papers introducing new, "secure" schemes that end up being broken. And yet these schemes had

proofs of security! Especially for complicated schemes, it sometimes turns out that the authors didn't think of the most clever adversary to attack the scheme, so the proof is incorrect. In other cases, they might assume that something is hard to do, so even if a clever adversary realizes that this is the best way to break the scheme it won't be able to pull off this kind of attack — but then someone discovers a way to solve that hard problem. This means the proof is correct, but it rests on an incorrect assumption, so its conclusion doesn't hold in the real world. (For example, RSA cryptography, introduced in 1977[1], is based on the difficulty of factoring integers. But in 1994, Peter Shor discovered a fast way to factor integers[2] — if you have a quantum computer. If you're interested in how RSA cryptography works, you can take a look at the RSA cryptography packet!)

Example 2.6 We already saw that the scheme in Example 2.3 is not correct. But is it private?

You might think so at first, since the number 3 (the first share) is unrelated to the secret, and the second share isn't giving away the secret if the adversary doesn't know that it's defined as the secret divided by 3 (rounded down). But in reality, we can't assume that the adversary doesn't know the inner workings of the Share algorithm: we need to assume the worst.

This means that $\mathcal{A}$ can consistently win the game if it sets $i = 2$. It first picks two values x_0, x_1 that are multiples of 3. When it gets s_2 from the game, it multiplies it by 3 and compares that value to x_0 and x_1. If it equals x_0, it outputs $b' = 0$; otherwise, it outputs $b' = 1$. By the definition of the scheme, $\mathcal{A}$ wins with probability 1, so its advantage is $\frac{1}{2}$, and the scheme is not private.

***Bonus Exercise* 2.5** Is the following scheme private?

$\mathsf{Share}(s)$	$\mathsf{Rec}(s_1, s_2)$
if s is even **then**	**return** $s_1 + s_2$
$\quad s_1 \leftarrow \{1, \ldots, 2^\lambda/2\}$	
else	
$\quad s_1 \leftarrow \{1, \ldots, 2^\lambda\}$	
return $(s_1, s - s_2)$	

2.3 Proving Security*

So far, we've only seen secret sharing schemes that were not private. Let's work through a case in which we *can* prove privacy, using the additive secret sharing scheme as an example.

Theorem 1. *The additive secret sharing scheme defined in Figure 1 is private.*

Proof. Let x_0 and x_1 be any two integers. Define $s_{0,i}$ as the ith share output by $\mathsf{Share}(x_0)$ and $s_{1,i}$ as the ith share output by $\mathsf{Share}(x_1)$. Since $n = 2$ for this scheme, the set I chosen by $\mathcal{A}$ will be of size 1; call the only number in that set i.

If $i = 1$, then $s_{0,1}$ and $s_{1,1}$ are distributed uniformly at random by the definition of Share (the first share is a uniformly random value). Thus, they are perfectly indistinguishable, and any adversary $\mathcal{A}$ can only randomly guess the value of b. So

$$\Pr[b = b' \mid i = 1] = \frac{1}{2}.$$

If $i = 2$, then $s_{0,2} = s - s_{0,1} \mod 2^\lambda$ and $s_{1,2} = s - s_{1,1} \mod 2^\lambda$. But as we said before, $s_{0,1}$ and $s_{1,1}$ are uniform, which implies that $s_{0,2}$ and $s_{1,2}$ are also uniform. Therefore they are perfectly indistinguishable, and

$$\Pr[b = b' \mid i = 2] = \frac{1}{2}.$$

Putting the two cases together, we see that for all possible values of x_0, x_1, i (and thus for all adversaries possible adversaries $\mathcal{A}$),

$$\begin{aligned}
\Pr[b = b'] =& \Pr[b = b' \mid i = 1] \cdot \Pr[i = 1] + \\
& \Pr[b = b' \mid i = 2] \cdot \Pr[i = 2] \\
=& \frac{1}{2}\big(\Pr[i = 1] + \Pr[i = 2]\big) \\
=& \frac{1}{2}(1) \\
=& \frac{1}{2}
\end{aligned}$$

From the definition of the game,

$$\Pr[\text{SS-priv}_{\mathcal{A},\mathcal{S}} = 1] = \Pr[b = b'],$$

so the advantage for any adversary $\mathcal{A}$ is

$$\Pr[\text{SS-priv}_{\mathcal{A},\mathcal{S}} = 1] - \frac{1}{2} = 0$$

which is clearly close to 0! So the additive secret sharing scheme is private. □

3 Shamir's Secret Sharing

Until now, we've only seen n-out-of-n secret sharing schemes. The "n-out-of-n" part means that the reconstruction algorithm needs at least n out of the n total shares to recover the secret; that is, it needs all of the shares to recover the secret. If the Share algorithm outputs 2 shares, we need both shares to reconstruct.

In general, though, $(t+1)$-out-of-n secret sharing schemes exist for any positive integers t and n. (t stands for "threshold", since it determines the minimum number of parties necessary for reconstruction.) In this section, we'll see how to construct such a secret sharing scheme using the properties of polynomials.

3.1 Polynomials

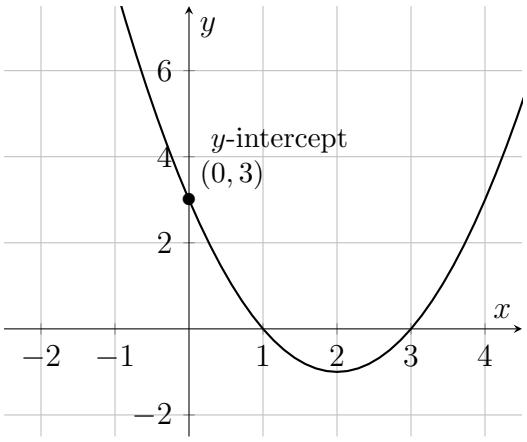
A *polynomial* is an expression consisting of powers of a variable (or several variables, but we'll stick with polynomials in one variable in this packet). Each power is multiplied by a number called its coefficient. Here's an example:

$$x^2 - 4x + 3$$

The standard form for a polynomial in one variable (called a univariate polynomial) is

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0 \tag{3}$$

where the $a_n, \ldots, a_1$ are the coefficients of the polynomials. They are constant (fixed) values, usually integers. n is a positive integer called the *degree* of the polynomial. The example polynomial above is a degree-2 polynomial.



You may have plotted a polynomial before to show how its value changes with different values of x. A plot of our example polynomial is shown above. In that case, we are plotting the equation

$$f(x) = x^2 - 4x + 3$$

where $f(x)$ is read as "f of x" and indicates that the expression to the right of the equal sign is a function of the variable x.

The *y-intercept* of a function is the place where it crosses the y-axis, i.e. its value when $x = 0$. In our example, the y-intercept is 3. Notice that we could have calculated it without plotting the equation:

$$f(0) = (0)^2 - 4(0) + 3 = 3$$

When a polynomial is written in standard form as in Equation 3, the y-intercept is a_0.

Exercise 3.1 Write down the degree and y-intercept of each of the following polynomials (without plotting them!):

(a) $f(x) = x^2 + 3x - 1$

(b) $f(x) = 5x^2 + 11$

(c) $f(x) = -2x^3 - x^2 + 9x$

(d) $f(x) = 3x^5 - 2x^3 - 15$

(e) $f(x) = (x^2 - 1)(x + 3)$

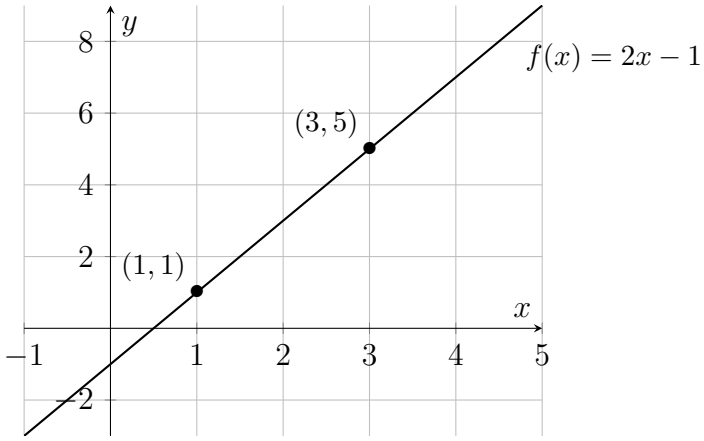
(f) $f(x) = 2(x - 6)(x + 2)(x - 5)$

(g) $f(x) = 2x^3(x + 16)$

3.1.1 Uniqueness

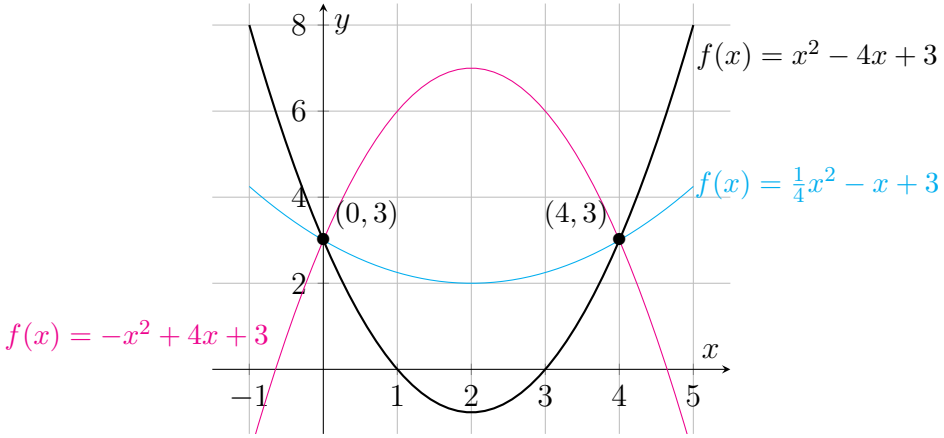
An important property of polynomials is that we can describe any one polynomial with a set of points. If they meet a few simple conditions, these points will correspond to exactly one equation, so we say they *uniquely define* that polynomial.

You've probably learned that two points uniquely define a line:

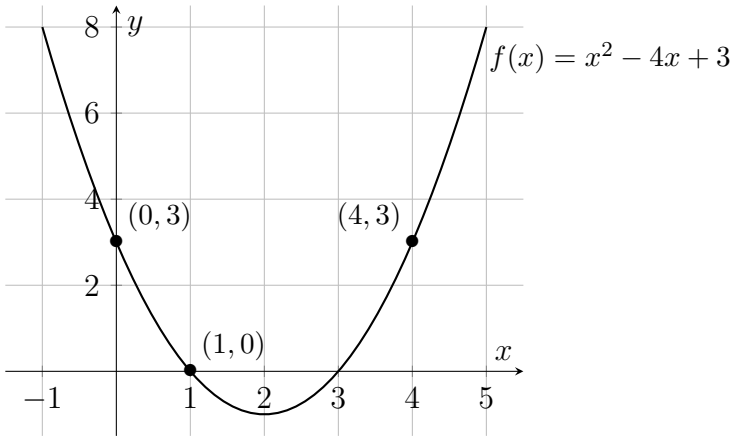


In fact, we can view a line as a degree-1 polynomial, so we can uniquely define it via a set of points! Instead of giving someone the equation for this line, you could give them 2 points on the line (for example, $(1, 1)$ and $(3, 5)$) and they'd still know exactly what line you're talking about.

To specify a particular degree-2 polynomial, two points aren't enough. For example, $x^2 - 4x + 3$ goes through the points $(0, 3)$ and $(4, 3)$, but so do many other degree-2 polynomials:



Instead, 3 points are needed to uniquely define a degree-2 polynomial:



How many points do you think are needed to uniquely define a degree-n polynomial? Once you think you have an answer, flip to the next page.

The process of recovering a polynomial passing through a set of points is called *interpolation*. Accordingly, this general rule about the uniqueness of a polynomial described by a set of points is called the interpolation theorem:

> INTERPOLATION THEOREM.
> Given a set of $t+1$ points, there exists a unique degree-t polynomial passing through those points.

Exercise 3.2 How many points are needed to uniquely define each polynomial in the previous exercise?

If getting from a set of points to an equation sounds difficult, don't worry because there are well-known techniques that always work. Most programming languages actually have these built-in, so you don't need to know the details; you just plug points in and out comes a polynomial! In case you want to know more, though, the next section explains how one of these techniques works.

3.1.2 Lagrange Interpolation*

In this section we'll learn about one method of doing polynomial interpolation called *Lagrange interpolation*. It's described by a single equation:

$$P(x) = \sum_{i=0}^{m-1} y_i \ell_i(x), \text{ where } \ell_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{m-1} \frac{x - x_j}{x_i - x_j} \tag{4}$$

Let's break it down bit by bit. First, in case you aren't familiar with the symbols, $\sum_{i=0}^{m-1}$ is called *summation notation* or *sigma notation*. It means to evaluate the expression after $\sum$ for each value of i (starting at the number below the summation symbol and ending at the number above) and then add all those terms together. For example:

$$\sum_{i=1}^{m} i = 1 + 2 + \ldots + m$$

$\prod$ (*product notation* or *pi notation*) is similar, but we multiply the expressions instead of adding them:

$$\prod_{i=1}^{m} i = 1 \cdot 2 \cdot \ldots \cdot m$$

So, $\sum_{i=1}^{4} i = 1 + 2 + 3 + 4 = 10$ and $\prod_{i=1}^{4} i = 1 \cdot 2 \cdot 3 \cdot 4 = 24$.

***Bonus Exercise* 3.3** Evaluate the following expressions. Keep a close eye on the starting value of i!

(a) $\sum_{i=1}^{5} 2i$

(b) $\sum_{i=1}^{5} 1$

(c) $\sum_{i=0}^{4} x_i$ where x_i means the ith elements of the set $\{1, 5, -3, 0, 8\}$ and the numbering starts at 0 (that is, $x_0 = 1$).

(d) $\sum_{i=0}^{2} x_i$ for the same set.

***Bonus Exercise* 3.4** Repeat the previous exercise, substituting $\prod$ for $\sum$.

Now, back to Lagrange interpolation. We start with a set of points $(x_0, y_0), \ldots, (x_n, y_n)$. For each point (x_i, y_i), we compute the expression $\ell_i(x)$. For instance, for $i = 2$, it will be of the form

$$\frac{x - \boxed{x_0}}{x_2 - \boxed{x_0}} \frac{x - \boxed{x_1}}{x_2 - \boxed{x_1}} \frac{x - \boxed{x_3}}{x_2 - \boxed{x_3}} \cdots \frac{x - \boxed{x_n}}{x_2 - \boxed{x_n}}$$

Then we plug that expression $\ell_i(x)$, along with the y-value y_i for that point, into the summation, and we'll end up with a polynomial. If we did everything right, that will be exactly the polynomial defined by those points.

This looks pretty complicated, but there's a pattern to be seen here. If we let $x = x_j$ for any $j \neq 2$, what does $\ell_i(x)$ evaluate to? In every case, one of the numerators will end up being 0. Since anything multiplied by 0 is 0, the whole expression will be 0—except when $x = x_2$ in our example, or more generally, when $x = x_i$. In fact, in these cases, the expression always evaluates to 1.

$$\ell_i(x_j) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

What does this pattern mean? Looking back at Equation 4, when we plug some x_j into $P(x)$, we'll get a sum of expressions that are all 0 except for one,

which will be $y_i \ell_i(x_i) = y_i$. This means that, by construction, $P(x_i) = y_i$ for all i, which is just what we want.

Let's work through an example using the degree-1 polynomial we plotted in Section 3.1.1: we'll use the points $(1, 1), (3, 5)$ to recover the equation of the line. Remember that, in this packet, we start numbering the elements of a set at 0, so $(x_0, y_0) = (1, 1)$ and $(x_1, y_1) = (3, 5)$.

Example 3.1 Since we have 2 points, $m = 2$:

$$\begin{aligned} P(x) &= \sum_{i=0}^{1} y_i \ell_i(x) \\ &= y_0 \ell_0(x) + y_1 \ell_1(x) \end{aligned}$$

Next, let's substitute in the y-coordinates of our points:

$$= 1 \cdot \ell_0(x) + 5 \cdot \ell_1(x)$$

Now let's evaluate the polynomials $\ell_i(x)$:

$$\begin{aligned} \ell_0(x) &= \prod_{\substack{j=0 \\ j \neq 0}}^{1} \frac{x - x_j}{x_0 - x_j} \\ &= \frac{x - x_1}{x_0 - x_1} \\ &= \frac{x - 3}{1 - 3} = \frac{x - 3}{-2} \\ \ell_1(x) &= \prod_{\substack{j=0 \\ j \neq 1}}^{1} \frac{x - x_j}{x_1 - x_j} \\ &= \frac{x - x_0}{x_1 - x_0} \\ &= \frac{x - 1}{3 - 1} = \frac{x - 1}{2} \end{aligned}$$

Let's pause for a moment to investigate the pattern with the ℓ_is. We'll fix the value of x and evaluate both ℓ_0 and ℓ_1. Starting with $x = x_0 = 1$:

$$\begin{aligned} \ell_0(x_0) &= \frac{x_0 - 3}{-2} = \frac{1 - 3}{-2} = \frac{-2}{-2} = 1 \\ \ell_1(x_0) &= \frac{x_0 - 1}{2} = \frac{1 - 1}{2} = 0 \end{aligned}$$

Now $x = x_1 = 3$:

$$\ell_0(x_1) = \frac{x_1 - 3}{-2} = \frac{3-3}{-2} = 0$$
$$\ell_1(x_1) = \frac{x_1 - 1}{2} = \frac{3-1}{2} = \frac{2}{2} = 1$$

Just as we deduced above!

To return to our main task of computing $P(x)$, let's plug the expressions for ℓ_0 and ℓ_1 back into the summation:

$$\begin{aligned} P(x) &= 1 \cdot \ell_0(x) + 5 \cdot \ell_1(x) \\ &= 1\left(\frac{x-3}{-2}\right) + 5\left(\frac{x-1}{2}\right) \\ &= \frac{-(x-3)}{2} + \frac{5(x-1)}{2} \\ &= \frac{5x - 5 - (x-3)}{2} \\ &= \frac{4x-2}{2} = 2x - 1 \end{aligned}$$

...and we've recovered the same equation we graphed earlier!

Example 3.2 Let's do the degree-2 example from Section 3.1.1 next. Our set of points is $\{(0,3),(1,0),(4,3)\}$ and $m = 3$.

$$\begin{aligned} P(x) &= \sum_{i=0}^{2} y_i \ell_i(x) \\ &= y_0\ell_0(x) + y_1\ell_1(x) + y_2\ell_2(x) \\ &= 3 \cdot \ell_0(x) + 0 \cdot \ell_1(x) + 3 \cdot \ell_2(x) \end{aligned}$$

The polynomials $\ell_i(x)$ are:

$$\begin{aligned}\ell_0(x) &= \prod_{\substack{j=0\\j\neq 0}}^{2} \frac{x-x_j}{x_0-x_j} \\ &= \frac{x-x_1}{x_0-x_1}\frac{x-x_2}{x_0-x_2} \\ &= \frac{x-1}{0-1}\frac{x-4}{0-4} \\ &= \frac{x-1}{-1}\frac{x-4}{-4} = \frac{(x-1)(x-4)}{4}\end{aligned}$$

$$\begin{aligned}\ell_1(x) &= \prod_{\substack{j=0\\j\neq 1}}^{2} \frac{x-x_j}{x_1-x_j} \\ &= \frac{x-x_0}{x_1-x_0}\frac{x-x_2}{x_1-x_2} \\ &= \frac{x-0}{1-0}\frac{x-4}{1-4} \\ &= \frac{x}{1}\frac{x-4}{-3} = \frac{x(x-4)}{-3}\end{aligned}$$

$$\begin{aligned}\ell_2(x) &= \prod_{\substack{j=0\\j\neq 2}}^{2} \frac{x-x_j}{x_2-x_j} \\ &= \frac{x-x_0}{x_2-x_0}\frac{x-x_1}{x_2-x_1} \\ &= \frac{x-0}{4-0}\frac{x-1}{4-1} \\ &= \frac{x}{4}\frac{x-1}{3} = \frac{x(x-1)}{12}\end{aligned}$$

Now we can simplify $P(x)$ to get:

$$
\begin{aligned}
P(x) &= 3 \cdot \ell_0(x) + 0 \cdot \ell_1(x) + 3 \cdot \ell_2(x) \\
&= 3 \cdot \frac{(x-1)(x-4)}{4} + 0 \cdot \frac{x(x-4)}{-3} + 3 \cdot \frac{x(x-1)}{12} \\
&= \frac{3(x-1)(x-4)}{4} + \frac{3x(x-1)}{12} \\
&= \frac{9(x-1)(x-4) + 3x(x-1)}{12} \\
&= \frac{9(x^2-5x+4) + (3x^2-3x)}{12} \\
&= \frac{9x^2-45x+36+3x^2-3x}{12} \\
&= \frac{12x^2-48x+36}{12} \\
&= x^2 - 4x + 3
\end{aligned}
$$

and again we've arrived at the same polynomial we graphed.

Exercise 3.5 Using the points and expressions in Example 3.2, evaluate $\ell_0(x), \ell_1(x), \ell_2(x)$ at the following points:

(a) x_0

(b) x_1

(c) x_2

***Bonus Exercise* 3.6** Use Lagrange interpolation to find the unique degree-2 polynomial through the points $\{(-1,-16),(1,-2),(2,14)\}$.

3.2 Sharing Secrets Using Polynomials

Shamir secret sharing is a $(t+1)$-out-of-n secret sharing scheme, for some positive integers t and n. This means that we split the secret s into n values and distribute them to n people. Then, at least $t+1$ of those people must work together to recover s.

Let's update our definition of secret sharing from Section 2.2 to include $(t+1)$-out-of-n secret sharing schemes where $t+1 \neq n$.

Definition 4 (Secret sharing scheme (updated))**.** *Let $\mathcal{D}$ be the input domain and $\mathcal{D}_s$ be the share domain. A secret sharing scheme is a pair of efficient algorithms* $(\mathsf{Share}, \mathsf{Rec})$ *and two associated natural numbers t, n such that*

- Share *takes as input a secret $s \in \mathcal{D}$ and outputs n shares in $\mathcal{D}_s$.*
- Rec *takes as input m shares $s_1, \ldots, s_m \in D_s$ and output some value $y \in \mathcal{D}$ or a special symbol $\perp$ indicating failure. (If $m < t+1$, it outputs $\perp$.)*

Now we're ready to put everything we've learned together and define Shamir's secret sharing scheme[9].

$\mathsf{Share}(s)$	$\mathsf{Rec}(s_1, \ldots, s_m)$
$a_1, \ldots, a_t \leftarrow\!\!\$\, \{1, \ldots, 2^\lambda\}$	**if** $m < t+1$ **then**
$a_0 = s$	**return** $\perp$
$f(x) = a_t x^t + \ldots + a_1 x + a_0$	**else**
return $((1, f(1)), \ldots, (n, f(n)))$	$f(x) = \mathsf{interpolate}(s_1, \ldots, s_m)$
	return $f(0)$

Figure 3: Shamir's secret sharing scheme

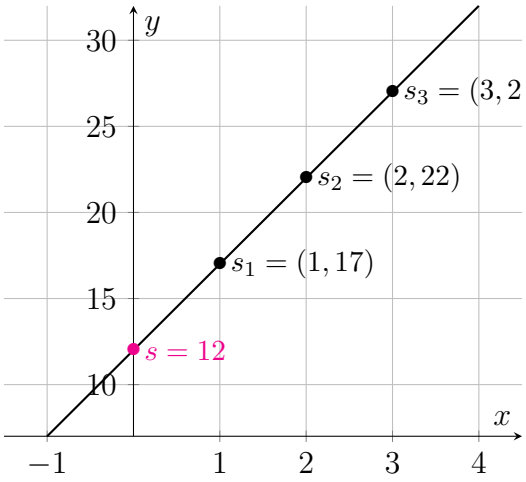
To share a secret s in $\mathcal{D}$, we choose a random degree-t polynomial f by sampling t random coefficients and setting f's y-intercept to the secret. Then

[9] Adi Shamir introduced this scheme in a short 1979 paper entitled "How to share a secret"[3]. The paper is only two pages long, so if you're feeling adventurous you could have a go at it! You can find it online at `http://web.mit.edu/6.857/OldStuff/Fall03/ref/Shamir-HowToShareASecret.pdf`.

we pick n points on f (the convention is to evaluate f at 1, 2, and so on, up to n)[10]. These n points are the shares.

To reconstruct, we need at least $t+1$ points. The reconstruction algorithm takes these points $s_i = (x_i, y_i)$ and tries to recover f using interpolation. (If there are not enough points, interpolation would fail, so we don't even try and instead return the symbol $\perp$ to indicate an error.) Once we recover a polynomial, we evaluate it at 0 to find its y-intercept and output that.

Example 3.3 Here's how we would compute $\mathsf{Share}(12)$ with $t = 1$ and $n = 3$. First, we pick one random integer a_1, say 5. Then $f(x) = 5x + 12$. Our three shares are $(1, 17), (2, 22), (3, 27)$. Below is a visual representation.



Example 3.4 Now let's say we receive two of the previous shares to reconstruct: $(1, 17), (3, 27)$. Say we know they are 2-out-of-3 shares (this is realistic; in practice, t and n would be known to all the parties, or the points would be labeled with the values of t and n used to generate them.) How do we reconstruct?

[10]To be exact, Shamir's secret sharing evaluates f in a way that ensures the values $f(x_i)$ are elements of something called a *finite field*. We won't go into details here about what that means, since such polynomials can't be graphed in two dimensions, but just know that this is important for the scheme to be truly private.

First of all, we know that reconstruction is possible because we have $m = 2$ shares and $t + 1 = 2$, so $m \geq t + 1$ as required for the reconstruction algorithm not to fail immediately. Now we need to interpolate to recover the degree-2 polynomial they represent. (If you didn't read Section 3.1.2, you can skip to the evaluation of $f(0)$ at the bottom of this page.)

Let's re-number the input shares starting with 0, so $(x_0, y_0) = (1, 17)$ and $(x_1, y_1) = (3, 27)$.

$$\begin{aligned} f(x) &= \sum_{i=0}^{m-1} y_i \ell_i(x) \\ &= y_0 \ell_0(x) + y_1 \ell_1(x) \\ &= 17\ell_0(x) + 27\ell_1(x) \end{aligned}$$

At this point, we can take a little shortcut. We know we only care about finding the value of f at 0, which means we only need to find $\ell_0(0)$ and $\ell_1(0)$ instead of the full expressions. So,

$$\ell_0(0) = \prod_{\substack{j=0 \\ j \neq 0}}^{1} \frac{0 - x_j}{x_0 - x_j} = \frac{-3}{1 - 3} = \frac{3}{2}$$

$$\ell_1(0) = \prod_{\substack{j=0 \\ j \neq 1}}^{1} \frac{0 - x_j}{x_1 - x_j} = \frac{-1}{3 - 1} = -\frac{1}{2}$$

Then

$$\begin{aligned} f(0) &= 17\ell_0(0) + 27\ell_1(0) \\ &= 17 \cdot \frac{3}{2} + 27 \cdot -\frac{1}{2} \\ &= \frac{51 - 27}{2} \\ &= \frac{24}{2} = 12 \end{aligned}$$

***Bonus Exercise* 3.7** Work in a small group. Everyone in the group should pick a secret number to share. Let n be the number of people in your group and pick t so that $t + 1 < n$. Use Share to compute shares of your secret and give each person in the group one share. Now a subgroup of $t + 1$ people should work together to reconstruct the secret using Rec[11]. Once you succeed, form a different group of $t + 1$ people and run Rec again using this new group of points. You should get the same result!

Notice that in the scheme presented in Figure 3, it's possible for someone to lie about their point, thereby causing the interpolation algorithm to return a different polynomial besides f. In that case, $f(0)$ might not equal s and we'd recover the wrong secret!

This doesn't violate correctness, however, because in that case we aren't running Rec on outputs of the Share algorithm, so the correctness definition doesn't apply. Instead, what's happening is that our scheme fails when the parties don't behave honestly. In cryptography, we say that the scheme is only secure in the presence of *semi-honest* adversaries (the scheme has *semihonest security*.) There are ways of fixing this scheme to guarantee security against *malicious* adversaries (*malicious security*), but that's outside the scope of this packet.

Exercise 3.8 Visit `https://bit.ly/ShamirSS` in your browser. This is a Google Colab notebook written in the Python programming language. It already has the functions Share and Rec from Shamir's secret sharing scheme. Work through the examples in the notebook to share and reconstruct any numbers you want using this scheme, then read on to find out how to share secret messages!

[11] You can use `https://www.dcode.fr/lagrange-interpolating-polynomial` to do the Lagrange interpolation.

4 Beyond secret sharing

Congratulations on making it through this packet! You have not only learned about secret sharing, a building block for many other cryptographic primitives that are used in the real world[12], but have also gotten a glimpse into what modern cryptography research looks like. The same definitions and proof techniques you saw in this packet pop up in many other areas of cryptography, and you now have a head start in understanding the techniques in many other subfields.

[12]Most notably, secret sharing forms the basis of many *multiparty computation (MPC)* schemes. These allow multiple parties, as the name implies, to compute functions over their secret inputs. For example, the Boston Women's Workforce Council organized an effort which used MPC[4, page 12] to calculate statistics about the salaries of the workforce in about 16% of the Boston area to show that the gender pay gap was wider than what the U.S. Bureau of Labor Statistics reported[4]. MPC was necessary because companies needed to keep their data on employee salaries secret due to privacy concerns.

References

[1] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[2] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[3] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[4] Boston Women's Workforce Council. Boston Women's Workforce Council report 2017. `https://www.boston.gov/sites/default/files/document-file-01-2018/bwwc_2017_report.pdf`, January 2018. Retrieved June 2021.