Number theory

explained from first principles

This article and its code were first published on 17 September 2022 and last modified on 10 February 2026. If you like the article, please share it with your friends on social media or support me with a donation. You can also cite this article, download it as a PDF, see what people are saying about it on X.com, join the discussion on Reddit, or use Google Translate to read it in another language.

▼ Cite this article

You can cite this article in various citation styles as follows:

MLA: Etter, Kaspar. "Number theory explained from first principles." *Explained from First Principles*, 17 Sep. 2022, https://explained-from-first-principles.com/number-theory/. Accessed 10 Feb. 2026.

CMOS: Etter, Kaspar. "Number theory explained from first principles." *Explained from First Principles*, September 17, 2022. Accessed February 10, 2026. https://explained-from-first-principles.com/number-theory/.

APA: Etter, K. (2022, September 17). Number theory explained from first principles. *Explained from First Principles*. Retrieved February 10, 2026, from https://explained-from-first-principles.com/number-theory/

IEEE: K. Etter, "Number theory explained from first principles," *Explained from First Principles*, Sep. 17, 2022. [Online]. Available: https://explained-from-first-principles.com/number-theory/. [Accessed: Feb. 10, 2026].

BibTeX:
```
@misc{etter_2022_number_theory,
    title = {Number theory explained from first principles},
    url = {https://explained-from-first-principles.com/number-theory/},
    journal = {Explained from First Principles},
    author = {Etter, Kaspar},
    date = {2022-09-17},
    year = {2022},
    month = {Sep},
    day = {17},
    edition = {2026-02-10},
    urldate = {2026-02-10}
}
```

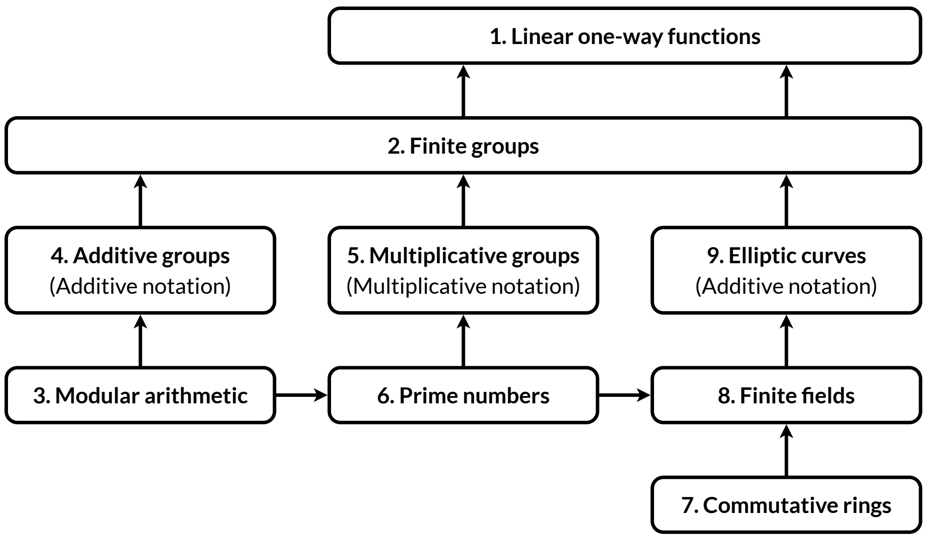
If you are worried about the persistence of this website, you can link to the latest snapshot of the Internet Archive instead.

▼ Preface

Number theory is the study of integers and the relations between them. Carl Friedrich Gauss (1777 − 1855) allegedly said that mathematics is the queen of the sciences and that number theory is the queen of mathematics. What he meant by this is that mathematics, unlike the empirical sciences, studies regularities which are independent from the universe that we live in. Not having to deal with the messiness of the real world makes mathematics the purest of all sciences. Similarly, he must have considered number theory to be the most elegant and the purest branch of mathematics, given its beauty and that it was studied for centuries for its own sake. Or as Leonard Eugene Dickson (1874 − 1954) put it: "Thank God that number theory is unsullied by any application". This changed with the advent of modern cryptography and coding theory in the second half of the 20th century. Unlike the mathematicians who discovered most of what I cover in this article, we're studying number theory only for its applications in these two fields, which will be the topics of the next two articles.

▼ Outline

The goal of this article is to understand how we can construct linear one-way functions using finite groups which have a certain property. There are two families of finite groups which are widely used in modern cryptography: multiplicative groups and elliptic curves. The former are based on modular arithmetic, the latter on finite fields. Prime numbers play a crucial role in both. We'll study these topics in a somewhat peculiar order and also visit additive groups and commutative rings on our journey:



How the various chapters of this article build on one another.

Since it's easy to get lost in theorems and proofs, I start this article with introducing and motivating the concept of linear one-way functions so that we never lose sight of our goal. Afterwards, I cover finite groups in the abstract in order to establish the terminology that we will use for the rest of the article. Establishing such a mental framework first also helps us to process our observations later on. Additionally, proving properties in the abstract saves us a lot of work when looking at multiple examples. If you read about these topics for the first time, feel free to jump ahead and play with concrete examples before understanding the theory behind them. If you really want to grasp the math behind modern cryptography, you likely have to read this article more than once anyway. The reason for covering modular arithmetic and multiplicative groups before prime numbers is to get as far as possible before the math gets more challenging and to understand why prime numbers are so special. After learning about elliptic curves, I cover the best known algorithms for breaking our linear one-way functions in a bonus chapter, before concluding the article with formal proofs of basic group properties as an appendix.

▼ **Contributions**

There exist already plenty of textbooks and blog posts on number theory (see below for some recommendations), so I wasn't sure whether it's worth adding my own. Since future articles will require a thorough understanding of number theory and my ambition is to explain things from first principles, I proceeded anyway. Looking at the result, here's what sets this article apart:

- **Focus**: Number theory is a big field with many advanced concepts. This article focuses on the aspects which are relevant for modern cryptography and coding theory. As you'll see, almost all theorems mentioned in this article are being used later on.
- **Notation**: As we'll see, two notations are used in group theory, and I haven't found any other source which consistently provides both. Not having to translate between the two yourself reduces your cognitive load.
- **Intuition**: I went to great lengths to complement formal proofs with more intuitive explanations. While formal rigor is important, nothing beats an intuitive understanding of a theorem or an algorithm.
- **Completeness**: Apart from the chapter on elliptic curves, I took no shortcuts when explaining why things are the way they are. Therefore, this article contains the complete proofs of almost all covered theorems, including Carmichael's totient function, the Monier-Rabin bound, square roots modulo composite numbers, and the expected running time of Pollard's rho algorithm, which I haven't found online outside of PDFs. While you might not be interested in such advanced proofs for now, this article is designed to serve as a work of reference for your whole journey into modern cryptography.
- **Interactivity**: Depending on how you count them, this article contains up to 40 interactive tools, which I also published on a separate page. Many tools, such as the repetition tables, visualize important group properties. Many other tools show how important algorithms work, such as the extended Euclidean algorithm, the Miller-Rabin primality test, and the Tonelli-Shanks algorithm. I also implemented the main algorithms to solve the discrete-logarithm problem, such as Pollard's rho algorithm, the Pohlig-Hellman algorithm, and the index-calculus algorithm. Besides being fun to play with, these tools allow you to check your own hypotheses, which makes mathematics an empirical science after all. 🙂

▼ **Math aversion**

It should be no surprise that an article about math contains a lot of math. If you're afraid of math, then see this article as a free exposure-therapy session. On a more serious note: Number theory is quite different from the math you endured during high school. Number theory is mostly about understanding why some properties follow from others. You won't have to calculate anything yourself; this is what computers are for. At times, the notation can seem daunting, but this is actually a problem of language and communication, not math. The situation isn't made easier by the fact that different notations are used to talk about the same things. While I tried to keep everything as unconfusing as possible, this is an aspect that I couldn't change.

▼ **Why proofs?**

There is a big difference between understanding how something works and understanding why something works. In mathematics, answers to why questions are called proofs. Proofs explain why a statement is true by showing how the statement can be deduced from already proven statements (called theorems) and statements which are simply assumed to be true (called axioms). If you want to understand only how modern cryptography works, you can skip this whole article and read just the next one. If you want to understand also why modern cryptography works, there is no way around proofs, even if you find them overly formal and tedious. But of course, you can always ignore answers to questions you would not have asked.

▼ **Further reading**

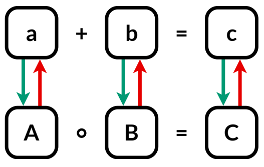
The two most important sources when writing this article were A Computational Introduction to Number Theory and Algebra by Victor Shoup and the Handbook of Applied Cryptography by Alfred Menezes, Paul van Oorschot, and Scott Vanstone (1947 – 2014). Consult the former for mathematical proofs and the latter for algorithmic aspects. Other useful sources are wikis for proofs, such as ProofWiki and Groupprops, and question-and-answer websites, such as Stack Exchange and Quora.

Linear one-way functions

Definition

The term "linear one-way function" consists of three words:

- **Function**: A function maps inputs from one set to outputs in the same or some other set. In mathematics, functions are always deterministic, which means that the output is determined entirely by the input. I write $f(a) = A$, with f being the name of the function, lowercase a denoting the input, and uppercase A referring to the output of the function.
- **One-way**: Computing the output A from the input a is efficient, whereas finding an input a which maps to a given output A is computationally infeasible. I depict efficient calculations with a green arrow and infeasible ones with a red arrow.
- **Linear**: We restrict the input of our function to integers (the whole numbers) and require that there exists some operation $\circ$ so that $f(a + b) = f(a) \circ f(b)$. (The ring operator $\circ$ is often used to denote the composition of functions, but I use it simply as a placeholder for an actual operator.) In other words, if $a + b = c$, then $f(a) \circ f(b) = f(c)$, which can be illustrated as follows:



The properties of a linear one-way function.

▼ **Comparison with cryptographic hash functions**

As we saw in a previous article, cryptographic hash functions work on inputs of arbitrary size and have to fulfill additional requirements, namely second-preimage resistance and collision resistance. We don't care about these properties here because we will limit our linear one-way functions to a finite set of inputs and construct them in such a way that unequal inputs are mapped to unequal outputs. Depending on whether you limit the set of inputs or not, second preimages either don't exist or are trivial to find. On the other hand, cryptographic hash functions are typically not linear (as long as we ignore provably secure hash functions, which are of limited practical use).

▼ **Computational complexity theory**

Computational complexity theory classifies problems according to how many steps are needed to solve them. Determining the precise number of steps required to solve an instance of a problem is of little interest, though. What computer scientists care about is how the required effort depends on the size of the input: If you double the input, does the number of steps that it takes to find a solution grow polynomially, such as quadrupling, or exponentially? Problems whose computational complexity grows exponentially with the input size become intractable if you make the input large enough. One-way functions can exist only if there are problems which cannot be solved in polynomial time but solutions to which can be verified in polynomial time. It is widely believed that such problems exist, but we still lack a proof for this. This conjecture, which is known as P ≠ NP, is the most important open question in computer science. Examples of problems for which no polynomial-time algorithms are known but whose solution can be verified in polynomial time are integer factorization and the discrete-logarithm problem (DLP).

Motivation

Linear one-way functions are useful in cryptography because they allow a party to conceal information by feeding it into a linear one-way function, while another party can still perform computations with the output of the linear one-way function. For example, a party who knows the output $f(a)$ and the input b can calculate $f(a \cdot b)$ without having to learn a by repeating $f(a)$ b times:

$$f(a \cdot b) = f(\underbrace{a + \ldots + a}_{b \text{ times}}) = \underbrace{f(a) \circ \ldots \circ f(a)}_{b \text{ times}}.$$

Notation

Linear one-way functions are an abstract concept, which I made up for this introduction to asymmetric cryptography. Instead of using the generic operator ∘, the operation in the output set of actual linear one-way functions is usually written as multiplication or addition, even when it has nothing to do with multiplication or addition of integers. As a consequence, every equation can be written in a generic, a multiplicative, and an additive notation. I do this with boxes like the following, where I choose the default notation (or "all") depending on the context, and you can select another notation by clicking on the corresponding tab:

Generic | Multiplicative | Additive | **All**

Generic

$$f(a + b) = f(a) \circ f(b)$$

Multiplicative

$$f(a + b) = f(a) \cdot f(b)$$

Additive

$$f(a + b) = f(a) + f(b)$$

You can change the notation throughout this article by double-clicking on the corresponding tab.

Repetitions

As we will see, linear one-way functions are typically constructed by repeating an element from the output set, the so-called generator G, the specified number of times. Repeated multiplication is written as exponentiation, and repeated addition is written as multiplication, where the multiplication sign is usually omitted:

Generic | Multiplicative | Additive | **All**

Generic

$$f(a) = \underbrace{G \circ \ldots \circ G}_{a \text{ times}}$$

Multiplicative

$$f(a) = \underbrace{G \cdot \ldots \cdot G}_{a \text{ times}} = G^a$$

Additive

$$f(a) = \underbrace{G + \ldots + G}_{a \text{ times}} = aG$$

The generic notation is a valuable reminder that the multiplicative and the additive notation stand for the same thing. However, repetitions are a bit cumbersome to write, which is why only the multiplicative and the additive notation are used in practice.

Linearity

It is easy to see why repeating an element from the output set is a linear operation (assuming that the operation is associative):

Generic | Multiplicative | Additive | All

Generic

$$f(a + b) = \underbrace{G \circ \ldots \circ G}_{a+b \text{ times}} = \underbrace{G \circ \ldots \circ G}_{a \text{ times}} \circ \underbrace{G \circ \ldots \circ G}_{b \text{ times}} = f(a) \circ f(b)$$

Multiplicative

$$f(a+b) = G^{a+b} = G^a \cdot G^b = f(a) \cdot f(b)$$

Additive

$$f(a+b) = (a+b)G = aG + bG = f(a) + f(b)$$

Zero as input

If zero is a valid input to a linear function, there has to be an element in the output set which does not affect any other element:

Generic | Multiplicative | Additive | All

Generic

$$f(a) = f(a+0) = f(a) \circ f(0)$$

Multiplicative

$$f(a) = f(a+0) = f(a) \cdot f(0)$$

Additive

$$f(a) = f(a+0) = f(a) + f(0)$$

This element is called the identity element or the neutral element. Given its special role, we assign a specific letter to it:

Generic | Multiplicative | Additive | **All**

Generic

$$f(0) = \underbrace{G \circ \ldots \circ G}_{0 \text{ times}} = E$$

Multiplicative

$$f(0) = G^0 = I$$

Additive

$$f(0) = 0G = O$$

Note that I use the letters I and O instead of 1 and 0 because elements in the output set do not have to be integers. In the case of elliptic curves, for example, O is the point at infinity. The resemblance with the corresponding integer is no coincidence, though.

Negative inputs

If we allow the input of a linear function to be negative, there has to be an inverse element for each element in the output set:

Generic | Multiplicative | Additive | All

Generic

$$f(0) = f(a + (-a)) = f(a) \circ f(-a) = E$$

Multiplicative

$$f(0) = f(a + (-a)) = f(a) \cdot f(-a) = I$$

Additive

$$f(0) = f(a + (-a)) = f(a) + f(-a) = O$$

In the following chapter, we will formalize these concepts and prove interesting properties about them.

▼ **Abstract algebra**

Algebra is the study of how to manipulate equations with symbols, and abstract algebra is the study of algebraic structures, which consist of a set and certain operations on the elements of the set. The goal of abstract algebra is to understand the properties of such structures at the highest level of abstraction so that these properties are applicable to any structure which satisfies the same requirements. Each algebraic structure has its own set of requirements, which are called axioms. Axioms are the premises from which conclusions are derived. The next chapter is about finite groups, but we'll encounter other algebraic structures later on, namely commutative rings and finite fields.

Finite groups

Group axioms

A group consists of a set and a binary operation, which combines any two elements of the set according to the following rules:

- **Closure**: When applying the operation to any two elements of the set, the resulting element is also part of the set.
- **Associativity**: Operations can be evaluated in any order without changing the result (i.e. parentheses don't matter).
- **Identity**: There exists a unique identity element which, when combined with any element, results in the other element.
- **Invertibility**: Each element has a unique inverse. If you combine an element with its inverse, you get the identity element.

The operation can be written in different ways, which leads to different notations for the identity element and the inverses:

Generic | Multiplicative | Additive | **All**

Generic

- **Associativity**: For any elements A, B, and C, $(A \circ B) \circ C = A \circ (B \circ C)$.
- **Identity element E**: For any element A, $A \circ E = E \circ A = A$.
- **$\overline{A}$ for the inverse of A**: For any element A, $A \circ \overline{A} = \overline{A} \circ A = E$.

Multiplicative

- **Associativity**: For any elements A, B, and C, $(A \cdot B) \cdot C = A \cdot (B \cdot C)$.
- **Identity element I**: For any element A, $A \cdot I = I \cdot A = A$.
- **A^{-1} for the inverse of A**: For any element A, $A \cdot A^{-1} = A^{-1} \cdot A = I$.

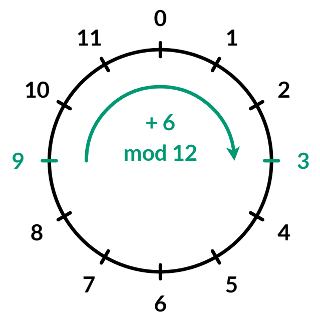
Additive

- **Associativity**: For any elements A, B, and C, $(A + B) + C = A + (B + C)$.
- **Identity element O**: For any element A, $A + O = O + A = A$.
- **$-A$ for the inverse of A**: For any element A, $A + (-A) = (-A) + A = O$.

Usually, the group axioms are presented in a reduced version, from which properties like the uniqueness of the identity are derived. Since this is not important for our purposes, I moved such derivations to the end of this article.

▼ **Example of a finite group**

The hours of an analog clock form a finite group. Its set consists of the elements 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12. In order to make things easier mathematically, we relabel 12 as 0. The operation is addition, where we wrap around to 0 after reaching 11:



6 hours past 9 a.m. is 3 p.m.

As we'll discuss later, the wrapping around corresponds to the modulo operation. Clearly, this group is closed: No matter how much time passes, the hour hand never leaves the clock. Associativity is given by the linearity of time and addition: Waiting for B hours and then for C hours is the same as waiting for B + C hours. 0 is the identity element: A + 0 = 0 + A = A. And finally, every element has an inverse: the number of hours you have to wait until it's noon or midnight again. For example, 2 + 10 = 0.

Unique solution

The defining property of a group is that the following equation has a unique solution for any elements A and B of the group:

Generic | Multiplicative | Additive | All

Generic

$X \circ A = B$ has a unique solution, namely $X = B \circ \overline{A}$
because $(B \circ \overline{A}) \circ A = B \circ (\overline{A} \circ A) = B \circ E = B$.

Any two solutions X_1 and X_2 are the same because

$$\begin{gathered} X_1 \circ A = X_2 \circ A \\ (X_1 \circ A) \circ \overline{A} = (X_2 \circ A) \circ \overline{A} \\ X_1 \circ (A \circ \overline{A}) = X_2 \circ (A \circ \overline{A}) \\ X_1 \circ E = X_2 \circ E \\ X_1 = X_2. \end{gathered}$$

Multiplicative

$X \cdot A = B$ has a unique solution, namely $X = B \cdot A^{-1}$
because $(B \cdot A^{-1}) \cdot A = B \cdot (A^{-1} \cdot A) = B \cdot I = B$.

Any two solutions X_1 and X_2 are the same because

$$\begin{gathered} X_1 \cdot A = X_2 \cdot A \\ (X_1 \cdot A) \cdot A^{-1} = (X_2 \cdot A) \cdot A^{-1} \\ X_1 \cdot (A \cdot A^{-1}) = X_2 \cdot (A \cdot A^{-1}) \\ X_1 \cdot I = X_2 \cdot I \\ X_1 = X_2. \end{gathered}$$

Additive

$X + A = B$ has a unique solution, namely $X = B + (-A)$
because $(B + (-A)) + A = B + ((-A) + A) = B + O = B$.

Any two solutions X_1 and X_2 are the same because

$$\begin{gathered} X_1 + A = X_2 + A \\ (X_1 + A) + (-A) = (X_2 + A) + (-A) \\ X_1 + (A + (-A)) = X_2 + (A + (-A)) \\ X_1 + O = X_2 + O \\ X_1 = X_2. \end{gathered}$$

The same is true if X is on the right side of A. Since the solution is unique, different combinations map to different results:

Generic | Multiplicative | Additive | All

Generic

If $X_1 \neq X_2$, then $X_1 \circ A \neq X_2 \circ A$.

Multiplicative

If $X_1 \neq X_2$, then $X_1 \cdot A \neq X_2 \cdot A$.

Additive

If $X_1 \neq X_2$, then $X_1 + A \neq X_2 + A$.

Commutative groups

An operation is commutative if swapping the inputs does not change the output, i.e. the following holds for any elements A and B:

Generic | Multiplicative | Additive | All

Generic

$$A \circ B = B \circ A$$

Multiplicative

$$A \cdot B = B \cdot A$$

Additive

$$A + B = B + A$$

Commutativity is not required in the above definition. It follows from the reduced group axioms that the right identity is also an identity when applied from the left and that any right inverse is also an inverse when applied from the left, even if the operation itself is not commutative. A group whose operation is commutative is called a commutative group or an abelian group, named after Niels Henrik Abel (1802 – 1829). (Even though “abelian” is derived from a proper name, the first letter is usually not capitalized.)

Element repetitions

Instead of combining two different elements, a single element can be combined repeatedly with itself. We write this as follows:

Multiplicative | Additive | **Both**

Multiplicative

$$A^n = \underbrace{A \cdot \ldots \cdot A}_{n \text{ times}}$$

Additive

$$nA = \underbrace{A + \ldots + A}_{n \text{ times}}$$

As noted earlier, the generic operation ∘ lacks a concise notation for repetitions, which is why I don't include it in this section.

Since the group operation is associative (which means that we can move parentheses around), it follows immediately that:

Multiplicative | Additive | **Both**

Multiplicative

$$A^{m+n} = \underbrace{A \cdot \ldots \cdot A}_{m+n \text{ times}} = (\underbrace{A \cdot \ldots \cdot A}_{m \text{ times}}) \cdot (\underbrace{A \cdot \ldots \cdot A}_{n \text{ times}}) = A^m \cdot A^n$$

Additive

$$(m+n)A = \underbrace{A + \ldots + A}_{m+n \text{ times}} = (\underbrace{A + \ldots + A}_{m \text{ times}}) + (\underbrace{A + \ldots + A}_{n \text{ times}}) = mA + nA$$

And since repetitions of inverses cancel as many repetitions of the element itself, we can make it work for differences as well:

Multiplicative | Additive | **Both**

Multiplicative

$$A^{m-n} = \underbrace{A \cdot \ldots \cdot A}_{m-n \text{ times}} = (\underbrace{A \cdot \ldots \cdot A}_{m \text{ times}}) \cdot (\underbrace{A^{-1} \cdot \ldots \cdot A^{-1}}_{n \text{ times}}) = A^m \cdot (A^{-1})^n$$

Additive

$$(m-n)A = \underbrace{A + \ldots + A}_{m-n \text{ times}} = (\underbrace{A + \ldots + A}_{m \text{ times}}) + (\underbrace{(-A) + \ldots + (-A)}_{n \text{ times}}) = mA + n(-A)$$

If we don't require the difference to be positive, we get the following definitions for zero and a negative number of repetitions:

Multiplicative | Additive | **Both**

Multiplicative

$$A^0 = I$$
$$A^{-n} = (A^{-1})^n$$

Additive

$$0A = O$$
$$(-n)A = n(-A)$$

▼ **Element repetitions example**

In the case of an analog clock, we wrap around to 0 when reaching 12. I write $A =_{12} B$ to denote that $A = B$ up to a multiple of 12. Repeating the element 5, we have $2 \cdot 5 =_{12} 5 + 5 =_{12} 10$, followed by $3 \cdot 5 =_{12} 5 + 5 + 5 =_{12} 10 + 5 =_{12} 3$, and so on:

1 · 5	**2 · 5**	**3 · 5**	**4 · 5**	**5 · 5**	**6 · 5**	**7 · 5**	**8 · 5**	**9 · 5**	**10 · 5**	**11 · 5**	**12 · 5**	**…**
5	10	3	8	1	6	11	4	9	2	7	0	…

Repeating the element 5 in the so-called additive group of integers modulo 12.

Using the above definitions, we have that $0 \cdot 5 =_{12} 0$ and $(-n)5 =_{12} n(-5)$, where $-5 =_{12} 7$ because $5 + 7 =_{12} 0$. Therefore:

(−1) · 5 = 1 · 7	(−2) · 5 = 2 · 7	(−3) · 5 = 3 · 7	(−4) · 5 = 4 · 7	(−5) · 5 = 5 · 7	(−6) · 5 = 6 · 7	(−7) · 5 = 7 · 7	(−8) · 5 = 8 · 7	(−9) · 5 = 9 · 7	(−10) · 5 = 10 · 7	(−11) · 5 = 11 · 7	(−12) · 5 = 12 · 7	…
7	2	9	4	11	6	1	8	3	10	5	0	…

The negative repetitions of the element 5 in the additive group of integers modulo 12.

Fast repetitions

The goal of this article is to construct linear one-way functions. As we've seen in the introduction, repeating an element is a linear operation. As we'll see later, figuring out how many times an element has been repeated is (presumably) computationally infeasible in some groups. In this section, we want to understand why repeating an element a certain number of times is so much easier than determining the number of repetitions when given the result. Let's start with how we can compute element repetitions efficiently.

Instead of performing one repetition at a time, we can compute A repeated n times for any integer n using the following insight:

Multiplicative | Additive | Both

Multiplicative

$$A^n = \begin{cases} (A^{-1})^{-n} & \text{if } n < 0, \\ I & \text{if } n = 0, \\ A^{n-1} \cdot A & \text{if } n \text{ is odd}, \\ (A^{\frac{n}{2}})^2 & \text{if } n \text{ is even}. \end{cases}$$

Additive

$$nA = \begin{cases} (-n)(-A) & \text{if } n < 0, \\ O & \text{if } n = 0, \\ (n-1)A + A & \text{if } n \text{ is odd}, \\ 2(\frac{n}{2}A) & \text{if } n \text{ is even}. \end{cases}$$

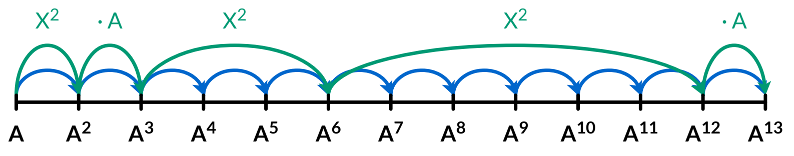
Solving a problem by reducing it to smaller instances of the same problem is known as recursion. The above algorithm terminates because a negative n is transformed into a positive n and then n gets smaller in every iteration until it reaches zero. Since an odd number becomes even when you subtract one, n is halved at least in every other iteration. Therefore, the running time of the algorithm is logarithmic with regard to the input n. Since halving a binary number is the same as dropping its least-significant bit, one can also say that the running time is linear in the bit length of n. In other words, if you double the length of a number (rather than its size), you just double the number of steps required.

When using the multiplicative notation, this algorithm is known as exponentiation by squaring or square-and-multiply. When using the additive notation, this becomes multiplication by doubling or double-and-add. The algorithm exploits the fact that the group operation is associative: Instead of evaluating the expression from left to right, the operations are grouped in such a way that as many intermediate results as possible can be reused. Reusing intermediate results instead of recomputing them is called common subexpression elimination in compiler design. It's also the core idea of dynamic programming. Let's look at an example:

Multiplicative | Additive | Both

Multiplicative

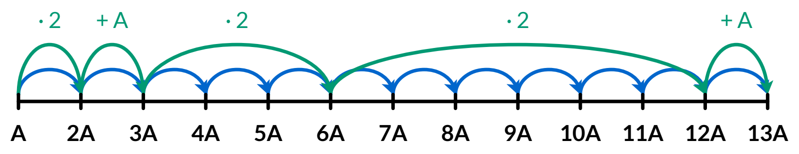
$$\begin{aligned} A^{13} &= A \cdot A \cdot A \cdot A \cdot A \cdot A \cdot A \cdot A \cdot A \cdot A \cdot A \cdot A \cdot A \\ &= \Big(\big(((A \cdot A) \cdot A) \cdot ((A \cdot A) \cdot A)\big) \cdot \big(((A \cdot A) \cdot A) \cdot ((A \cdot A) \cdot A)\big)\Big) \cdot A \\ &= \Big(\big((I \cdot \underset{\substack{\uparrow \\ 1}}{A})^2 \cdot \underset{\substack{\uparrow \\ 1}}{A}\big)^2 \cdot \underset{\substack{\uparrow \\ 0}}{I}\Big)^2 \cdot \underset{\substack{\uparrow \\ 1}}{A} \end{aligned}$$



We can compute A^{13} in 5 steps (in green) instead of 12 steps (in blue) by squaring intermediate results.

Additive

$$\begin{aligned} 13A &= A + A + A + A + A + A + A + A + A + A + A + A + A \\ &= \Big(\big(((A + A) + A) + ((A + A) + A)\big) + \big(((A + A) + A) + ((A + A) + A)\big)\Big) + A \\ &= 2\Big(2\big(2(O + \underset{\substack{\uparrow \\ 1}}{A}) + \underset{\substack{\uparrow \\ 1}}{A}\big) + \underset{\substack{\uparrow \\ 0}}{O}\Big) + \underset{\substack{\uparrow \\ 1}}{A} \end{aligned}$$



We can compute 13A in 5 steps (in green) instead of 12 steps (in blue) by doubling intermediate results.

Instead of 12 group operations, we performed only 5 (ignoring the combinations with the identity element). 13 written as a binary number is 1101 (8 + 4 + 1), which corresponds to the pattern of when you have to combine the intermediate result with the element before doubling/squaring it again. (If you look at the recursive formula, the least-significant bit determines whether the current number is odd or even and thus whether you combine the recursive invocation with the element or not before dropping this bit.)

Crucially, this technique works only if you know how many repetitions you have to perform. If you try to determine the number of repetitions from the result, you cannot work backwards and halve the number every other step, which is why this algorithm can be used only in one direction. I cover the best known algorithms for determining the number of repetitions in a separate chapter.

▼ Non-recursive algorithm

The above recursive algorithm can easily be turned into a non-recursive algorithm:

Multiplicative | Additive | Both

Multiplicative

```
function $repeat(A, n)$ {
  if $(n < 0)$ {
    $A := A^{-1}$
    $n := -n$
  }
  let $B := I$
  while $(n > 0)$ {
    if $(n \% 2 = 1)$ {
      $B := B \cdot A$
      $n := n - 1$
    }
    $A := A \cdot A$
    $n := n/2$
  }
  return $B$
}
```

Additive

```
function $repeat(A, n)$ {
  if $(n < 0)$ {
    $A := -A$
    $n := -n$
  }
  let $B := O$
  while $(n > 0)$ {
    if $(n \% 2 = 1)$ {
      $B := B + A$
      $n := n - 1$
    }
    $A := A + A$
    $n := n/2$
  }
  return $B$
}
```

This function returns the correct result because $B \cdot A^n$ (respectively $B + nA$) equals the desired result before and after every loop iteration and we repeat the loop until $n = 0$. Unlike the variant on Wikipedia, the above algorithm wastes one group operation in the last iteration when $n = 1$. On the other hand, we don't have to handle the case where $n = 0$ separately. Since computers represent numbers in binary, you can check whether n is odd with a bitwise and (typically written as `n & 1 === 1`) and divide n by 2 without having to subtract the least-significant bit first by shifting the bits one to the right (typically written as `n >> 1`). Since I haven't yet introduced any groups of interest, I will provide an interactive tool for fast repetitions later on.

Group order

A group is finite if its set contains only a finite number of elements. The number of elements is then called the order of the group. If $\mathbb{G}$ is the set of the group, its order is usually written as $|\mathbb{G}|$, which is the same notation as used to denote the cardinality of a set.

▼ **Notation for sets and elements**

A set is a collection of distinct elements. When a variable is used to denote a set, it is often written in blackboard bold, such as $\mathbb{Z}$ for the set of integers, $\mathbb{R}$ for the set of real numbers, and $\mathbb{C}$ for the set of complex numbers. The symbol $\in$ is used to denote that A "is" an element of the set $\mathbb{G}$, written as $A \in \mathbb{G}$. I put "is" in quotation marks because the same notation is used when A is not an element of the set but rather a variable which represents an element of the set. Sets are usually defined by enumerating all its elements within curly brackets, such as $\{1, 2, 3\}$, or by specifying a criteria for its elements after a vertical bar, such as $\{x \in \mathbb{Z} \mid x > 2\}$ for the set of integers greater than two.

▼ **Group order example**

As we saw earlier, the hours of an analog clock form a finite group under addition. Using the above notations, we write the group as $\mathbb{G} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ and its order as $|\mathbb{G}| = 12$.

Element order

If you keep repeating an element of a finite group, you will reach an earlier result again at some point because you run out of fresh elements. Let A be the element that we repeat and B be the first element that we reach twice, which we can depict as follows:

Generic | Multiplicative | Additive | All

Generic

$$E \circ \overbrace{\underbrace{A \circ A \circ \ldots \circ A}_{m \text{ times} = B} \circ \underbrace{A \circ \ldots \circ A}_{n-m \text{ times} = E}}^{n \text{ times} = B}$$

Multiplicative

$$I \cdot \overbrace{\underbrace{A \cdot A \cdot \ldots \cdot A}_{m \text{ times} = B} \cdot \underbrace{A \cdot \ldots \cdot A}_{n-m \text{ times} = I}}^{n \text{ times} = B}$$

Additive

$$O + \overbrace{\underbrace{A + A + \ldots + A}_{m \text{ times} = B} + \underbrace{A + \ldots + A}_{n-m \text{ times} = O}}^{n \text{ times} = B}$$

If you reach B for the first time after repeating A m times and again after repeating A n times, then A repeated $n - m$ times has to equal the identity element because the identity element is the unique solution to $B \circ X = B$. If we start with zero repetitions and thus the identity element instead of A, the identity element has to be the first element which we encounter again because as soon as $m > 0$, $n - m$ is smaller than n, which means that you reach the identity element before you reach B for the second time. The smallest $n > 0$ which results in the identity element when repeating A n times is called the order of the element A, written as $|A|$:

Generic | Multiplicative | Additive | All

Generic

$$\underbrace{A \circ \ldots \circ A}_{|A| \text{ times}} = E$$

Multiplicative

$$A^{|A|} = I$$

Additive

$$(|A|)A = O$$

In a finite group, every element has a finite order. Once you've reached the identity element, the elements repeat in the same order:

Generic | Multiplicative | Additive | All

Generic

$$\overbrace{\underset{\underset{A}{\downarrow}}{A} \circ \underset{\underset{B}{\downarrow}}{A} \circ \underset{\underset{C}{\downarrow}}{A} \circ \ldots \circ \underset{\underset{E}{\downarrow}}{A}}^{|A| \text{ times}} \circ \underset{\underset{A}{\downarrow}}{A} \circ \underset{\underset{B}{\downarrow}}{A} \circ \underset{\underset{C}{\downarrow}}{A} \circ \ldots$$

Multiplicative

$$\overbrace{\underset{\underset{A}{\downarrow}}{A} \cdot \underset{\underset{B}{\downarrow}}{A} \cdot \underset{\underset{C}{\downarrow}}{A} \cdot \ldots \cdot \underset{\underset{I}{\downarrow}}{A}}^{|A| \text{ times}} \cdot \underset{\underset{A}{\downarrow}}{A} \cdot \underset{\underset{B}{\downarrow}}{A} \cdot \underset{\underset{C}{\downarrow}}{A} \cdot \ldots$$

Additive

$$\overbrace{\underset{\underset{A}{\downarrow}}{A} + \underset{\underset{B}{\downarrow}}{A} + \underset{\underset{C}{\downarrow}}{A} + \ldots + \underset{\underset{O}{\downarrow}}{A}}^{|A| \text{ times}} + \underset{\underset{A}{\downarrow}}{A} + \underset{\underset{B}{\downarrow}}{A} + \underset{\underset{C}{\downarrow}}{A} + \ldots$$

If A repeated $|A|$ times results in the identity element, then A repeated $|A| - 1$ times has to result in the inverse element of A:

Generic | Multiplicative | Additive | All

Generic

$$A \circ \underbrace{A \circ \ldots \circ A}_{|A|-1 \text{ times}} = A \circ \overline{A} = \underbrace{A \circ \ldots \circ A}_{|A|-1 \text{ times}} \circ A = \overline{A} \circ A = E$$

Multiplicative

$$A \cdot \underbrace{A \cdot \ldots \cdot A}_{|A|-1 \text{ times}} = A \cdot A^{-1} = \underbrace{A \cdot \ldots \cdot A}_{|A|-1 \text{ times}} \cdot A = A^{-1} \cdot A = I$$

Additive

$$A + \underbrace{A + \ldots + A}_{|A|-1 \text{ times}} = A + (-A) = \underbrace{A + \ldots + A}_{|A|-1 \text{ times}} + A = (-A) + A = O$$

This is the most intuitive way to understand why a right inverse is also always a left inverse and vice versa.

▼ **Element order examples**

We can determine the order of every element in the group which corresponds to an analog clock by repeating each element:

1A	2A	3A	4A	5A	6A	7A	8A	9A	10A	11A	12A	Order |A|
0												|0| = 1
1	2	3	4	5	6	7	8	9	10	11	0	|1| = 12
2	4	6	8	10	0							|2| = 6
3	6	9	0									|3| = 4
4	8	0										|4| = 3
5	10	3	8	1	6	11	4	9	2	7	0	|5| = 12
6	0											|6| = 2
7	2	9	4	11	6	1	8	3	10	5	0	|7| = 12
8	4	0										|8| = 3
9	6	3	0									|9| = 4
10	8	6	4	2	0							|10| = 6
11	10	9	8	7	6	5	4	3	2	1	0	|11| = 12

The order of each element in the additive group of integers modulo 12.
(You can set the modulus to an arbitrary value in a tool below.)

▼ **Why do we need invertibility as an axiom?**

Since you get the inverse of any element A in any finite group simply by repeating A, you may wonder why we require each element to have an inverse in the group as one of our axioms. The reason for this is that without invertibility, $X \circ A = B$ can have several solutions, which means that you can get stuck in a loop which doesn't involve the identity element. (To see this happening, set the modulus to a composite number in the repetition table of multiplicative groups below.) Instead of requiring that each element has an inverse, we can require that $X \circ A = B$ and $A \circ Y = B$ have a unique solution for any elements A and B of the group. This is an alternative definition of a group and makes even the identity axiom redundant, as we'll see later.

Group generators

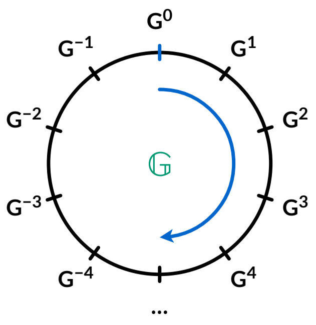
If an element reaches every element of the group when it is repeated, the element is said to generate the group. Such an element is called a generator and usually denoted as G. A group can have several generators or no generator. By definition, the order of each generator equals the order of the group: $|G| = |\mathbb{G}|$. The set of elements generated by an element A is usually written with angle brackets as $\langle A \rangle$. Also by definition, the set generated by a generator is the whole group: $\mathbb{G} = \langle G \rangle$.

▼ **Group generators example**

As shown in a previous box, the elements 1, 5, 7, and 11 generate the additive group modulo 12: $\mathbb{G} = \langle 1 \rangle = \langle 5 \rangle = \langle 7 \rangle = \langle 11 \rangle$.

Cyclic groups

A group with a generator is called cyclic because all its elements can be reached in a single cycle when repeating the generator G:



Repeating the generator **G** using the multiplicative notation.

Since every element can be written as a repetition of G, the group operation and element inversion can be performed as follows:

Multiplicative | Additive | Both

Multiplicative

Let $A = G^a$ and $B = G^b$ be arbitrary elements, then $A \cdot B = G^a \cdot G^b = G^{a+b}$ and $A^{-1} = (G^a)^{-1} = (G^{-1})^a = G^{-a}$.

Additive

Let $A = aG$ and $B = bG$ be arbitrary elements, then $A + B = aG + bG = (a + b)G$ and $-A = -(aG) = a(-G) = (-a)G$.

Therefore, cyclic groups are commutative because the addition of the integers in the repetition of the generator is commutative:

Multiplicative | Additive | Both

Multiplicative

Let $A = G^a$ and $B = G^b$ be arbitrary elements, then $A \cdot B = G^a \cdot G^b = G^{a+b} = G^{b+a} = G^b \cdot G^a = B \cdot A$.

Additive

Let $A = aG$ and $B = bG$ be arbitrary elements, then $A + B = aG + bG = (a + b)G = (b + a)G = bG + aG = B + A$.

▼ **Even number of generators**

Repeating the inverse of G generates the group in the opposite direction because the inverse undoes one repetition at a time:

Multiplicative | Additive | Both

Multiplicative

$$G^i = G^{|G|-(|G|-i)} = \underbrace{G^{|G|}}_{=I} \cdot (G^{-1})^{|G|-i} = (G^{-1})^{|G|-i}$$

Additive

$$iG = (|G| - (|G| - i))G = \underbrace{(|G|)G}_{=O} + (|G| - i)(-G) = (|G| - i)(-G)$$

As a consequence, every cyclic group with more than two elements has an even number of generators because the inverse of every generator is also a generator. (Since each inverse is unique and the inverse of each inverse is the original element again, two different generators cannot have the same inverse — and a generator can equal its inverse only for groups of order 2.)

▼ **Cyclic group example**

The group defined by an analog clock is cyclic and has an even number of generators, namely 1, 5, $7 =_{12} -5$, and $11 =_{12} -1$.

Subgroups

A group $\mathbb{H}$ is a subgroup of another group $\mathbb{G}$ if every element of $\mathbb{H}$ is also an element of $\mathbb{G}$ and they have the same operation. Since a group must have an identity element and the identity element is defined by the operation, the subgroup $\mathbb{H}$ has to include the same identity element as the supergroup $\mathbb{G}$. Given that the identity element results in itself when it is combined with itself, every group has a trivial subgroup, which contains only the identity element. Since associativity is a property of the operation and not of the elements, a subgroup satisfies this axiom simply by using the same operation as its supergroup. What has to be verified, though, is that the elements of the subgroup are closed under the group operation. If this is the case and the subgroup is non-empty and finite, the identity element and all inverses are guaranteed to be included as you reach them by repeating the corresponding element. Any element A of $\mathbb{G}$ generates a cyclic subgroup, which is denoted as $\langle A \rangle$. Generated subsets are always closed because combining repetitions of A results in another repetition of A. If A does not generate all elements of $\mathbb{G}$, then $\langle A \rangle$ is a proper subgroup of $\mathbb{G}$.

▼ **Subgroup examples**

The group defined by an analog clock, which consists of the elements $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$, has the following subgroups: $\{0\}$, $\{0, 6\}$, $\{0, 4, 8\}$, $\{0, 3, 6, 9\}$, $\{0, 2, 4, 6, 8, 10\}$, and $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$. Any other subset of $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$, such as $\{0, 1, 2, 3\}$, is not closed. (As we'll see later, all subgroups of cyclic groups are cyclic.)

▼ **Notation for the subgroup relation**

As you may have noticed, I began to denote a group by its set in this section. The subgroup relation is usually written as $\mathbb{H} \leq \mathbb{G}$ (or as $\mathbb{H} < \mathbb{G}$ if $\mathbb{H} \neq \mathbb{G}$) instead of $\mathbb{H} \subseteq \mathbb{G}$ because not every subset of elements forms a subgroup.

▼ **Intersection of subgroups is a subgroup**

Given two subgroups $\mathbb{H}_1$ and $\mathbb{H}_2$ of a group $\mathbb{G}$, their intersection $\mathbb{H} = \mathbb{H}_1 \cap \mathbb{H}_2$ is also a subgroup of $\mathbb{G}$ because:

- $\mathbb{H}$ is not empty because the identity element E of $\mathbb{G}$ is included in both $\mathbb{H}_1$ and $\mathbb{H}_2$. Therefore, $E \in \mathbb{H}$.
- $\mathbb{H}$ is closed because for any elements A and B in $\mathbb{H}$, A and B belong to both $\mathbb{H}_1$ and $\mathbb{H}_2$.
  Since $\mathbb{H}_1$ and $\mathbb{H}_2$ are closed, $A \circ B \in \mathbb{H}_1$ and $A \circ B \in \mathbb{H}_2$. Therefore, $A \circ B \in \mathbb{H}$.

On the other hand, the union of two subgroups is generally not a subgroup.

Subgroup cosets

If you combine each element of a subgroup $\mathbb{H}$ with a fixed element A of the supergroup $\mathbb{G}$, you get a coset of $\mathbb{H}$, which is written as:

Generic | Multiplicative | Additive | All

Generic

- **Right coset**: $\mathbb{H} \circ A = \{H \circ A \mid H \in \mathbb{H}\}$,
- **Left coset**: $A \circ \mathbb{H} = \{A \circ H \mid H \in \mathbb{H}\}$.

Multiplicative

- **Right coset**: $\mathbb{H} \cdot A = \{H \cdot A \mid H \in \mathbb{H}\}$,
- **Left coset**: $A \cdot \mathbb{H} = \{A \cdot H \mid H \in \mathbb{H}\}$.

Additive

- **Right coset**: $\mathbb{H} + A = \{H + A \mid H \in \mathbb{H}\}$,
- **Left coset**: $A + \mathbb{H} = \{A + H \mid H \in \mathbb{H}\}$.

If the group operation is commutative, the right coset and the left coset of a subgroup $\mathbb{H}$ and an element $A \in \mathbb{G}$ are the same. If the element A belongs to the subgroup $\mathbb{H}$, the right coset and the left coset equal the subgroup itself due to closure:

Generic | Multiplicative | Additive | All

Generic

$$\mathbb{H} \circ A = \mathbb{H} = A \circ \mathbb{H}$$

Multiplicative

$$\mathbb{H} \cdot A = \mathbb{H} = A \cdot \mathbb{H}$$

Additive

$$\mathbb{H} + A = \mathbb{H} = A + \mathbb{H}$$

Any two right cosets and any two left cosets are either equal or disjoint. Given arbitrary elements $A, B \in \mathbb{G}$, there are two cases:

Generic | Multiplicative | Additive | All

Generic

1. If $B \in \mathbb{H} \circ A$, then $\mathbb{H} \circ A = \mathbb{H} \circ B$
   because $B = H_B \circ A$ for some $H_B \in \mathbb{H}$, and thus for every element $C \in \mathbb{H} \circ B$, there's an $H_C \in \mathbb{H}$ so that $C = H_C \circ B = H_C \circ$

$(H_B \circ A) = (H_C \circ H_B) \circ A$, where $H_C \circ H_B \in \mathbb{H}$ due to closure, and thus $C \in \mathbb{H} \circ A$. So far, we have just shown that $\mathbb{H} \circ B \subseteq \mathbb{H} \circ A$. Since $B = H_B \circ A$, we have that $A = \overline{H_B} \circ B$, and thus $A \in \mathbb{H} \circ B$. This implies that $\mathbb{H} \circ A \subseteq \mathbb{H} \circ B$ for the same reason as before. Therefore, $\mathbb{H} \circ A = \mathbb{H} \circ B$.

2. If $B \notin \mathbb{H} \circ A$, then $\mathbb{H} \circ A \cap \mathbb{H} \circ B = \varnothing$ (the intersection of the two cosets results in the empty set)
because an overlap would mean that for some element C, there are $H_A, H_B \in \mathbb{H}$ so that $C = H_A \circ A = H_B \circ B$. But in this case, $B = \overline{H_B} \circ (H_A \circ A) = (\overline{H_B} \circ H_A) \circ A \in \mathbb{H} \circ A$, which contradicts the premise of the second case.

Multiplicative

1. If $B \in \mathbb{H} \cdot A$, then $\mathbb{H} \cdot A = \mathbb{H} \cdot B$
because $B = H_B \cdot A$ for some $H_B \in \mathbb{H}$, and thus for every element $C \in \mathbb{H} \cdot B$, there's an $H_C \in \mathbb{H}$ so that $C = H_C \cdot B = H_C \cdot (H_B \cdot A) = (H_C \cdot H_B) \cdot A$, where $H_C \cdot H_B \in \mathbb{H}$ due to closure, and thus $C \in \mathbb{H} \cdot A$. So far, we have just shown that $\mathbb{H} \cdot B \subseteq \mathbb{H} \cdot A$. Since $B = H_B \cdot A$, we have that $A = H_B^{-1} \cdot B$, and thus $A \in \mathbb{H} \cdot B$. This implies that $\mathbb{H} \cdot A \subseteq \mathbb{H} \cdot B$ for the same reason as before. Therefore, $\mathbb{H} \cdot A = \mathbb{H} \cdot B$.
2. If $B \notin \mathbb{H} \cdot A$, then $\mathbb{H} \cdot A \cap \mathbb{H} \cdot B = \varnothing$ (the intersection of the two cosets results in the empty set)
because an overlap would mean that for some element C, there are $H_A, H_B \in \mathbb{H}$ so that $C = H_A \cdot A = H_B \cdot B$. But in this case, $B = H_B^{-1} \cdot (H_A \cdot A) = (H_B^{-1} \cdot H_A) \cdot A \in \mathbb{H} \cdot A$, which contradicts the premise of the second case.

Additive

1. If $B \in \mathbb{H} + A$, then $\mathbb{H} + A = \mathbb{H} + B$
because $B = H_B + A$ for some $H_B \in \mathbb{H}$, and thus for every element $C \in \mathbb{H} + B$, there's an $H_C \in \mathbb{H}$ so that $C = H_C + B = H_C + (H_B + A) = (H_C + H_B) + A$, where $H_C + H_B \in \mathbb{H}$ due to closure, and thus $C \in \mathbb{H} + A$. So far, we have just shown that $\mathbb{H} + B \subseteq \mathbb{H} + A$. Since $B = H_B + A$, we have that $A = (-H_B) + B$, and thus $A \in \mathbb{H} + B$. This implies that $\mathbb{H} + A \subseteq \mathbb{H} + B$ for the same reason as before. Therefore, $\mathbb{H} + A = \mathbb{H} + B$.
2. If $B \notin \mathbb{H} + A$, then $\mathbb{H} + A \cap \mathbb{H} + B = \varnothing$ (the intersection of the two cosets results in the empty set)
because an overlap would mean that for some element C, there are $H_A, H_B \in \mathbb{H}$ so that $C = H_A + A = H_B + B$. But in this case, $B = (-H_B) + (H_A + A) = ((-H_B) + H_A) + A \in \mathbb{H} + A$, which contradicts the premise of the second case.

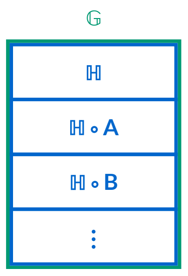
As different $H \in \mathbb{H}$ are mapped to different elements in every coset, all cosets contain the same number of elements, namely $|\mathbb{H}|$.

▼ **Subgroup cosets example**

Given the group $\mathbb{G} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ with addition modulo 12, which corresponds to an analog clock, the subgroup $\mathbb{H} = \{0, 3, 6, 9\}$ has the following cosets: $\mathbb{H} + 0 = \{0, 3, 6, 9\}$, $\mathbb{H} + 1 = \{1, 4, 7, 10\}$, $\mathbb{H} + 2 = \{2, 5, 8, 11\}$, $\mathbb{H} + 3 = \{3, 6, 9, 0\}$, and so on. (You can generate all cosets of a cyclic subgroup below.) It's easy to verify that these subgroup cosets are either equal or disjoint as proven above. For example, $\mathbb{H} + 0 = \mathbb{H} + 3$, whereas $\mathbb{H} + 1 \cap \mathbb{H} + 2 = \varnothing$.

Lagrange's theorem

Lagrange's theorem states that for any finite group $\mathbb{G}$, the order of every subgroup $\mathbb{H}$ of $\mathbb{G}$ divides the order of $\mathbb{G}$. In other words, if $\mathbb{H}$ is a subgroup of $\mathbb{G}$, $|\mathbb{G}|$ is a multiple of $|\mathbb{H}|$. Since each element $A \in \mathbb{G}$ belongs to a coset of $\mathbb{H}$ ($A \in \mathbb{H} \circ A$) and all cosets of $\mathbb{H}$ are either equal or disjoint and contain the same number of elements, the cosets of $\mathbb{H}$ split the group $\mathbb{G}$ into subsets of equal size:



The cosets of $\mathbb{H}$ form a partition of the supergroup $\mathbb{G}$
(i.e. every element of $\mathbb{G}$ is included in exactly one coset).

Since $\mathbb{H}$ is a coset of $\mathbb{H}$, it follows that $|\mathbb{H}|$ divides $|\mathbb{G}|$. This theorem is named after Joseph-Louis Lagrange (1736 – 1813).

Lagrange's theorem has several important consequences:

1. The order of every element A in $\mathbb{G}$ divides the order of $\mathbb{G}$ because A generates the subgroup $\langle A \rangle$, where $|\langle A \rangle| = |A|$.
2. Any element A in $\mathbb{G}$ repeated $|\mathbb{G}|$ times equals the identity element because with $|\mathbb{G}| = |A| \cdot n$ for some integer n:

Multiplicative | Additive | Both

Multiplicative

$$A^{|\mathbb{G}|} = A^{|A| \cdot n} = (A^{|A|})^n = I^n = I$$

Additive

$$(|\mathbb{G}|)A = (|A| \cdot n)A = n((|A|)A) = nO = O$$

3. Every group of prime order is cyclic, and every element except the identity element is a generator. (Given a group $\mathbb{G}$ where $|\mathbb{G}|$ is prime, the order of every $A \in \mathbb{G}$ has to be 1 or $|\mathbb{G}|$. If its order is 1, A is the identity element. If its order is $|\mathbb{G}|$, A is a generator.)

▼ **Lagrange's theorem example**

We determined the order of every element in the group which corresponds to an analog clock earlier. It's easy to verify that the order of each element divides the group's order. Since these generated subgroups are the only possible subgroups as explained later on, the order of every subgroup divides the group's order. And since $12A$ is a multiple of 12 for any element A, we have that $12A =_{12} 0$ for all elements of the group as implied by the second consequence of Lagrange's theorem.

▼ **Index and cofactor**

Given a group $\mathbb{G}$ and a subgroup $\mathbb{H}$, the ratio $\frac{|\mathbb{G}|}{|\mathbb{H}|}$ is called the index of the subgroup or the cofactor in the case of elliptic curves.

▼ **Proof without cosets for commutative groups**

If we restrict our attention to commutative groups, there's a much simpler proof for why you get the identity element when you repeat an element of such a group as many times as there are elements in the group. (I found this proof on page 34 of Victor Shoup's book.) Since you get different results when you combine an element A with two different elements B_1 and B_2, $f(X) = A \circ X$ is an invertible function. This means that $f(X)$ is a permutation, and thus $f(\mathbb{G}) = \{f(B) \mid B \in \mathbb{G}\} = \mathbb{G}$. Using Greek letters to iterate over all the elements of a commutative group, we can observe the following:

Multiplicative | Additive | Both

Multiplicative

$$\prod_{B \in \mathbb{G}} B = \prod_{B \in \mathbb{G}} f(B) = \prod_{B \in \mathbb{G}} A \cdot B = A^{|\mathbb{G}|} \cdot \Big(\prod_{B \in \mathbb{G}} B\Big)$$

After canceling $\prod_{B \in \mathbb{G}} B$ on both sides, we have that $I = A^{|\mathbb{G}|}$.

Additive

$$\sum_{B \in \mathbb{G}} B = \sum_{B \in \mathbb{G}} f(B) = \sum_{B \in \mathbb{G}} A + B = (|\mathbb{G}|)A + \Big(\sum_{B \in \mathbb{G}} B\Big)$$

After canceling $\sum_{B \in \mathbb{G}} B$ on both sides, we have that $O = (|\mathbb{G}|)A$.

You can replace B with $f(B)$ and aggregate all A outside the loop only if the operation of the group is commutative. The order of every element has to divide the order of the group because it cannot be larger and you wouldn't get the identity element if $|\mathbb{G}|$ was not a multiple of $|A|$. To make similar statements about non-commutative groups and non-cyclic subgroups, you still need Lagrange's theorem with its cosets. This is probably why this proof is not more popular; but it is enough for our purposes.

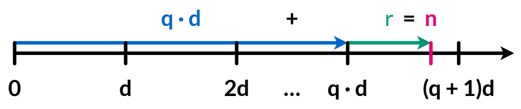
Modular arithmetic

Before we can look at some examples of finite groups, we have to learn about modular arithmetic first.

Euclidean division

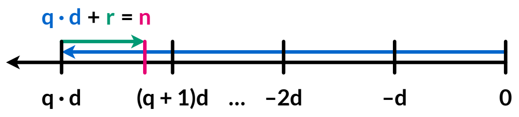
Integers aren't closed under division, but you can divide any two integers with a remainder, which is known as Euclidean division. Given a positive integer $d > 0$, every integer n can be written as $n = q \cdot d + r$, where q and r are integers and $0 \leq r < d$. In this equation, n is called the dividend (the quantity to be divided), d is called the divisor (the quantity which divides), q is called the quotient (how many times the divisor is included in the dividend), and r is called the remainder (the quantity which is left over).

Since the divisor d splits the number line into equal sections, the quotient q and the remainder r are unique:



The divisor d divides the number line for positive integers.

Personally, I'm satisfied with this geometric argument. You find a more formal proof for the existence and the uniqueness of q and r on Wikipedia. Since we are interested in positive divisors only, I didn't bother to define Euclidean division for a negative d. What we do care about, though, is that the remainder r is non-negative (i.e. greater than or equal to zero) even if the integer n is negative:



The divisor d also divides the number line for negative integers.

▼ Divisor

An integer d which divides another integer n without a remainder is a co-called divisor of n. This is usually written as $d \mid n$, which means that there is an integer c so that $c \cdot d = n$. In this context, the term "divisor" is used differently than in the Euclidean division above.

Modulo operation

The modulo operation returns the remainder of Euclidean division. The divisor of the division is also called the modulus (a small measure or interval) of the operation. The modulo operation is typically written as $n \bmod m$ or $n \mathbin{\%} m$, where n is the dividend and m is the modulus. For example, $8 \mathbin{\%} 3 = 2$. Many calculators, including Google Search and Apple Spotlight, support the modulo operation in both notations but struggle if you make the numbers large enough (because some use floating-point arithmetic). Since we want to perform calculations with very large integers in cryptography, it's time for the first interactive tool of this article:

Integer: 15 Modulus: 12

Integer:	15
Modulus:	12
Remainder:	3

▼ Modulo in programming languages

In many programming languages, including C, C++, Go, Java, and JavaScript, the `%` operator does not compute the remainder as presented here. Instead, these languages round the quotient towards zero and allow the remainder to be negative, which is listed as truncated division on Wikipedia. For example, `-7 % 2 == -1`, which means that you cannot test whether a potentially negative integer is odd by using `n % 2 == 1`. For the quotient, this behavior is probably desirable, which is why signed division is implemented like this in the `IDIV` instruction of the x86 instruction set. The situation becomes even more complicated if you allow the divisor/modulus to be negative.

Equivalence relation

When two numbers have the same remainder, we say that they are equivalent up to a multiple of the modulus, which means that their difference is a multiple of the modulus. Given the numbers a and b, this is usually written in one of the following two ways:

$$a \equiv b \pmod{m}$$
$$\text{or} \quad a \equiv_m b$$

I prefer the latter to the former because it makes it clearer that the aspect which is affected by the modulus is the comparison of the two numbers. The second notation is also more flexible as it allows us to use the modular comparison and the ordinary equality in a single line. Since the third dash increases the load on our eyes (similar to visual pollution) without any benefits, I will simply write:

$$a =_m b$$

The reason for not dropping the modulus in the subscript of the equals sign as well is that the modulus is a useful reminder of whether we perform a computation with simple numbers in the modular group or the repetition ring. If a computation is performed with something else, such as points or polynomials, we will leave the meaning of the equals sign to the context in which it is used.

We have now five different ways to express the same relation between two integers a and b:

$$a \mathbin{\%} m = b \mathbin{\%} m \iff a =_m b \iff (a - b) \mathbin{\%} m = 0 \iff m \mid (a - b) \iff a = b + c \cdot m \text{ for some integer } c$$

Equality of remainders is an equivalence relation, which is defined by the following three properties given any numbers a, b, and c:

- **Reflexivity**: $a =_m a$.
- **Symmetry**: If $a =_m b$, then $b =_m a$.
- **Transitivity**: If $a =_m b$ and $b =_m c$, then $a =_m c$.

These properties are satisfied when considering only the remainder of numbers because the remainder is unique for every number.

Congruence relation

The above equivalence relation is compatible with addition and multiplication in the sense that equivalent inputs yield equivalent outputs: For any integers a_1, a_2, b_1, and b_2 so that $a_1 =_m a_2$ and $b_1 =_m b_2$ (i.e. $a_1 = a_2 + c_a \cdot m$ and $b_1 = b_2 + c_b \cdot m$ for some integers c_a and c_b), we have that

- $a_1 + b_1 =_m a_2 + b_2$ because $a_1 + b_1 = (a_2 + c_a \cdot m) + (b_2 + c_b \cdot m) = (a_2 + b_2) + (c_a + c_b) \cdot m$, and
- $a_1 \cdot b_1 =_m a_2 \cdot b_2$ because $a_1 \cdot b_1 = (a_2 + c_a \cdot m) \cdot (b_2 + c_b \cdot m) = (a_2 \cdot b_2) + (a_2 \cdot c_b + c_a \cdot b_2 + c_a \cdot c_b \cdot m) \cdot m$.

An equivalence relation which is compatible with the operations of interest is a congruence relation. When calculating modulo m, we can eliminate multiples of m without affecting the equivalence relation. Since only the remainders are relevant, we can – and will – represent all numbers by their unique remainder. Instead of reinterpreting what it means to be equal, we could just as well overload the operators so that $a +_m b = (a + b) \mathbin{\%} m$ and $a \cdot_m b = (a \cdot b) \mathbin{\%} m$. This is an adequate way to think about modular arithmetic and also how I implemented the following tools. Regarding notation, it's better to keep the modulus next to the equals sign, though, because it makes longer expressions easier to read. Moreover, some operations, such as modular exponentiation, are written without an operator to which the modulus could be attached.

Additive groups

After having gone through quite a bit of theory, the time has finally come to see some finite groups in action. The integers modulo some integer m form a commutative group under addition. As explained in the previous chapter, you can either define the set of the group to contain all the integers and redefine what it means to be equal, or restrict the set of elements to $\{0, 1, \dots, m - 1\}$ and redefine addition as $A +_m B = (A + B) \mathbin{\%} m$. The latter is usually preferred because it makes the representation of elements unique. The number of elements is given by the modulus m. It's easy to see why addition modulo m satisfies the four group axioms:

- **Closure**: Every integer has a remainder between 0 and $m - 1$ (both inclusive). It's not possible to leave the set of elements.
- **Associativity**: Since reducing intermediate results to their remainder doesn't change the result, addition remains associative.
- **Identity**: The number 0 is the only identity element. For any number A, we have that $A + 0 = 0 + A = A$.
- **Invertibility**: The inverse of any number A is $m - A$ because $A + (m - A) =_m 0$.

The combinations of elements can be displayed in an operation table and the repetitions of each element in a repetition table.

Operation table

An operation table is a mathematical table which lists the results of combining any two elements with a binary operation. If the table is exhaustive, it defines the operation by enumeration. You likely had to learn the multiplication table up to a factor of 10 by heart in primary school, and you might also be familiar with truth tables, which are used to define truth functions. In the case of groups, an operation table is also called a group table, a composition table, or a Cayley table, named after Arthur Cayley (1821 – 1895). Just to be clear, an operation table has nothing to do with an operating table.

The following tool allows you to generate the operation table for all additive groups up to a modulus of 100:

Modulus: 10 Next prime Previous prime Coprime: Composite:

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2

+	0	1	2	3	4	5	6	7	8	9
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

All the elements which are coprime with the modulus are highlighted with a green background in the first row and the first column. We'll discuss the properties of operation tables when we look at multiplicative groups, where the tables are a bit more interesting.

Repetition table

Most of the introduction to group theory was about combining the same element with itself repeatedly. We learned that the identity element is reached before another element occurs twice and that this element order divides the group order. You can observe both properties for moduli up to 100 in the tool below, where each element is repeated in a separate row until it reaches the identity element. Such tables aren't commonly used to visualize Lagrange's theorem and other properties, such as the number of generators. I call them repetition tables, but feel free to refer to them as Etter tables. 😉

Modulus: 10 Next prime Previous prime Coprime: Repeat: Order: Totient:

1A	2A	3A	4A	5A	6A	7A	8A	9A	10A
0									
1	2	3	4	5	6	7	8	9	0
2	4	6	8	0					
3	6	9	2	5	8	1	4	7	0
4	8	2	6	0					
5	0								
6	2	8	4	0					
7	4	1	8	5	2	9	6	3	0
8	6	4	2	0					
9	8	7	6	5	4	3	2	1	0

The first row indicates how many times the element in the first column was repeated. (Remember that repeated addition is written as multiplication.) The tool highlights all the repetitions which divide the number of elements in the group with a blue background. These are the columns in which the identity element can be reached in the repetitions below. You can increase and decrease the modulus by pressing the up and down arrow keys of your keyboard when the cursor is in the input field.

In the case of additive groups, the following properties are easy to understand even if you don't know anything about group theory:

- **Group order**: A multiple of the modulus is obviously congruent to zero, i.e. $mA =_m 0$ for any number A. This was also one of the consequences of Lagrange's theorem. You can observe it in the repetition table above by enabling "Repeat".
- **Prime modulus**: If the modulus m is prime, you won't get a multiple of m until you repeat any element except zero m times. Since you cannot hit some element twice before reaching the identity element, all elements except zero generate the whole group.
- **Cyclic groups**: Additive groups modulo some number are always cyclic because the number 1 always generates the whole group.
- **Gray element**: If the modulus m is even, the element $\frac{m}{2}$ has an order of two. An order of two means that $\frac{m}{2}$ is its own inverse. All other elements except zero have an order greater than two because for all elements $A < \frac{m}{2}$, we have that $2A < m$, and for all elements $A > \frac{m}{2}$, we have that $m < 2A < 2m$. Thus, whenever an element has an even order (which can be the case only if m is even), you reach $\frac{m}{2}$ halfway to the identity element. To make it easier to see this, the tool marks $\frac{m}{2}$ with a gray background.

You can ignore the additional options of the tool for now; we'll make use of them later on.

Subgroup cosets

As we learned above, the order of every subgroup divides the order of the group because the cosets of a subgroup are either equal or disjoint and all elements of the group belong to a coset of a particular subgroup. The following tool demonstrates this for additive groups modulo some number. The first row contains the elements of the subgroup which is generated by the given element. It corresponds to the row of the given element in the group's repetition table with the difference that the identity element comes first instead of last. The second row contains the elements of the first row incremented by one, the third row contains the elements of the first row incremented by two, and so on. The reason why I put the identity element into the first column is to make it easy to recognize which coset is represented by a particular row.

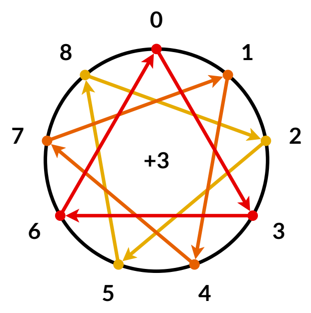
Modulus: 9 Next prime Previous prime Element: 3 Unique: Delay: 0.5

0	**3**	**6**
1	4	7
2	5	8
3	6	0
4	7	1
5	8	2
6	0	3
7	1	4
8	2	5

For example, setting the modulus to 9 and the element to 3, the first row is $\langle 3 \rangle = \{0, 3, 6\}$, the second row is $\langle 3 \rangle + 1 = \{0 + 1 = 1, 3 + 1 = 4, 6 + 1 = 7\}$, and so on. If the element in the first column appears in another row, then the elements in these two rows are identical, they just appear in a different order. If the element in the first column does not appear in another row, then the two rows have no elements in common. You can filter all the rows whose elements already appeared in the table by enabling "Unique". In order to make it easier to spot that each element appears exactly once when "Unique" is enabled, the elements of the group are highlighted with a blue background one after the other. You can disable this animation by setting the delay to 0.

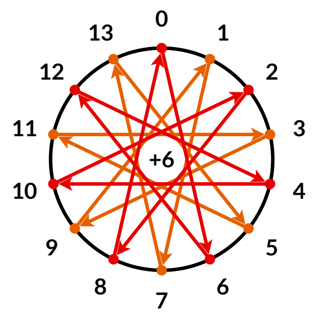
▼ **Visualization of cosets**

For additive groups modulo some number, even the fact that a group is partitioned by subgroup cosets is somewhat intuitive:



The subgroup generated by the element 3 in red
with the two disjoint cosets in orange and yellow
in the additive group modulo 9.

Admittedly, this is the simplest possible case. What happens if you don't hit 0 in the first pass and have to do more loops? If you repeat an element from the left side of the circle, then you just reach the elements in the opposite order (e.g. $(6 + 6)\ \%\ 9 = 3$ and $(3 + 6)\ \%\ 9 = 0$). For a more sophisticated example, let's look at the additive group modulo 14 with the element 6:



The subgroup generated by the element 6 in red
with the only disjoint coset in orange.

As you can see in the above graphic, the distance between adjacent elements of the subgroup is always the same. This has to be the case because repeating an element generates a cyclic subgroup, which is closed. As a consequence, the difference between the two closest elements is itself an element. (If we denote the closest elements as A and B, then $B + (-A)$ is also an element of the subgroup due to closure.) This difference can then be added and subtracted from all other elements. If this leads to an even smaller gap, you repeat this procedure until the subgroup is closed. If the original element does not generate the whole group, you get a disjoint coset when you rotate the generated subgroup by one. As we'll see later, the difference between adjacent elements of the generated subgroup equals the greatest common divisor of the generating element and the modulus, and the first time you hit zero again corresponds to the least common multiple of these two numbers.

Group notations

The additive group of integers modulo some number m is usually denoted as $(\mathbb{Z}/m\mathbb{Z})^+$ or $\mathbb{Z}_m^+$. (By convention, $\mathbb{Z}$ stands for the set of all integers. The letter comes from the German word for numbers, which is Zahlen.) The former notation refers to the additive group of the quotient ring $\mathbb{Z}/m\mathbb{Z}$. Since the second notation is also used for a different mathematical concept, mathematicians typically prefer the former, whereas cryptographers tend to settle for the latter.

Multiplicative groups

The integers modulo some integer m can also be combined using multiplication, which results in the so-called multiplicative group of integers modulo m. Since equivalence up to a multiple of m is preserved also under multiplication, it's easy to see why modular multiplication is still associative. It's also rather obvious that the number 1 is the identity element because for any number A, $A \cdot 1 = 1 \cdot A = A$. Since there is no number B for which $0 \cdot B =_m 1$, 0 has to be excluded from the set of elements. What remains to be seen is which other numbers below the modulus have an inverse and why the group is closed if the numbers without an inverse are excluded. The short answer is that integers have a multiplicative inverse if and only if the only factor in common with the modulus is 1. We will see in the next chapter why this is the case. For now, just be aware that the following tools do not always display groups. If the modulus is not prime, you have to enable the "coprime filter" to get a group. This is the reason why prime numbers play such an important role in number theory and cryptography.

▼ **If and only if**

Given two statements P and Q, "P if and only if Q" means that either both statements are true or neither of them is. In other words, P is necessary and sufficient for Q (and the other way around), which means that each statement implies the other. In formulas, this so-called material equivalence is usually written as $\iff$, which is a binary truth function with the following truth table (using $\top$ for true and $\bot$ for false):

P	Q	$P \iff Q$
$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\bot$
$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$

The definition of $\iff$.

Operation table

The following tool allows you to generate the operation table for all multiplicative groups up to a modulus of 100:

Modulus: 11 Next prime Previous prime Coprime: Zero: Composite:

·	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	1	3	5	7	9
3	3	6	9	1	4	7	10	2	5	8
4	4	8	1	5	9	2	6	10	3	7
5	5	10	4	9	3	8	2	7	1	6
6	6	1	7	2	8	3	9	4	10	5
7	7	3	10	6	2	9	5	1	8	4
8	8	5	2	10	7	4	1	9	6	3
9	9	7	5	3	1	10	8	6	4	2
10	10	9	8	7	6	5	4	3	2	1

Operation tables of groups, which are also known as Cayley tables, have some interesting properties. I limited the modulus to 100 not because these properties no longer hold for larger moduli but because the operation table becomes unwieldy way before that.

Symmetries

The table is symmetric along the diagonal axis from the upper left corner to the lower right corner if and only if the operation is commutative and the elements are listed in the same order in both dimensions. All the groups that we are interested in are commutative. If the operation is not commutative, the convention is that the first element determines the row and the second element determines the column of the table. If you play with the above tool, you notice that the table is also symmetric along the diagonal axis from the lower left corner to the upper right corner. The reason for this is that the product of two negative numbers is the same as the product of the corresponding positive numbers: $A \cdot B = B \cdot A = (-B) \cdot (-A) =_m (m - B) \cdot (m - A)$. This second symmetry is specific to multiplicative groups, which you can verify by going back to the additive groups.

Permutations

The group axioms imply that $A \circ X = B$ and $Y \circ A = B$ have unique solutions for any elements A and B of the group. This means that each row and column of a group's operation table may contain each element only once. Otherwise, the solution is not unique:

∘	…	X_1	…	X_2	…
⋮		⋮		⋮	
A	…	B	…	B	…
⋮		⋮		⋮	

A row (or column) may not contain the same element twice as $A \circ X = B$ would have more than one solution otherwise.

Due to closure, each combination of two elements results in another element. Therefore, none of the cells may be empty and each row and column has to contain each element exactly once. In other words, the rows and columns are permutations of the group's elements and the operation table forms a so-called Latin square, where each symbol occurs exactly once in each row and column.

Identity row and column

Since groups have an identity element, one of the rows has to match the column headers and one of the columns has to match the row headers. Put differently, the identity permutation, which maps every element of the group to itself, has to be included in the permutations of the rows and the columns. If the elements are listed in the same order along the vertical and the horizontal axis of the table, the identity row intersects the identity column on the diagonal from the upper left corner to the lower right corner because the identity element equals itself when it is combined with itself. (This property is called idempotence.)

▼ **Latin square + associativity = group**

Being a Latin square is not a sufficient condition to form a group, which can be demonstrated with the following example:

∘	A	B	C
A	B	A	C
B	A	C	B
C	C	B	A

This operation does not form a group.

This table lacks an identity row and column, but more importantly, the operation is not associative. For example, $(A \circ B) \circ C = A \circ C = C$, whereas $A \circ (B \circ C) = A \circ B = B$. Since associativity makes a statement about three elements and not just two, it is difficult to spot whether an operation is associative based on its table. If an operation defined by a Latin square is associative, it does form a group, though. In order to understand why this is the case, we need to understand why an associative operation with unique solutions implies that the identity element is the same for all elements.

Since every column of the operation table is a permutation of the group's elements, every element has to appear in its own column. This means that for every element D, there exists an element E so that $E \circ D = D$. By applying this element on both sides from the left, we get $E \circ (E \circ D) = E \circ D$. Due to associativity, $(E \circ E) \circ D = E \circ D$. Since $X \circ D = D$ has a unique solution, $E \circ E = E$. In other words, every identity for a single element is idempotent. What remains to be seen is that an idempotent element is an identity for all elements.

For any element F, $E \circ X = F$ has a unique solution. Since $E \circ E = E$, $(E \circ E) \circ X = F$ and thus $E \circ (E \circ X) = F$. Using the fact that $E \circ X = F$, it follows that F is the unique solution for $E \circ X = F$. This is why an identity for one element is an identity for all elements. The identity element is unique because $X \circ F = F$ would have two solutions otherwise. The same argument can be used to show that the right identity is also unique. The left identity E_L and the right identity E_R are the same because they are identities for each other: $E_L = E_L \circ E_R = E_R$. Since you can get from any element to any other element in a Latin square, you can get to the unique identity from any element, which means that each element has an inverse.

We have shown that any associative operation which generates a Latin square forms a group. I cover an alternative proof for these alternative group axioms in the last chapter.

▼ **Fixing the group of order 3**

Before we continue, let's fix the above operation table by starting with the necessary identity row and column:

∘	A	B	C
A	A	B	C
B	B	?	?
C	C	?	?

A skeleton for a group of order 3.

We may try to complete the operation table by setting $B \circ B$ to A, but this would lead to a conflict in the third row and column:

∘	A	B	C
A	A	B	C
B	B	A	!
C	C	!	?

This cannot be completed to a Latin square.

Given that C has to occur in the middle row and column and cannot occur a second time in the last row and column, it has to be that $B \circ B = C$. Another reason for this is that $B \circ B = A$ makes the order of B 2 since A is the identity element; but this violates Lagrange's theorem, which implies that the order of any element divides the order of the group. Therefore:

∘	A	B	C
A	A	B	C
B	B	C	A
C	C	A	B

The only group of order 3.

Since there was no other way to complete the operation table, this is the only group of order 3. In fact, all groups of prime order are unique up to a relabeling of the elements because as we saw earlier, groups of prime order are cyclic, and as we will see later, all cyclic groups of the same order behave identically. This particular group corresponds to the additive group modulo 3.

If you enjoy solving Sudokus, you might also enjoy solving the operation table for groups of higher order. For example, there are two solutions for groups of order 4. Can you find them? And can you prove that these are the only two groups of order 4? 🤓

Repetition table

The following tool repeats the elements of the multiplicative group modulo the given modulus similar to the additive groups above:

Modulus: 11 | Next prime | Previous prime | Coprime: | Repeat: | Order: | Totient:

11 is prime

A^1	A^2	A^3	A^4	A^5	A^6	A^7	A^8	A^9	A^{10}
1									
2	4	8	5	10	9	7	3	6	1
3	9	5	4	1					
4	5	9	3	1					
5	3	4	9	1					
6	3	7	9	10	5	8	4	2	1
7	5	2	3	10	4	6	9	8	1
8	9	6	4	10	3	2	5	7	1
9	4	3	5	1					
10	1								

As noted earlier, you have to enable "Coprime" to exclude the numbers which don't have an inverse and therefore don't belong to the group if the modulus isn't prime. Multiplicative groups are different from additive groups in several regards:

- **Non-cyclic groups**: Not all multiplicative groups are cyclic. For example, the multiplicative groups modulo 8, 12, and 15 have no generator. As we'll discuss later, all multiplicative groups modulo a prime number are cyclic, though.

- **Square root of one**: If a multiplicative group is cyclic and the modulus m is greater than two, then $m - 1$ is the only element with an order of two. ($m - 1$ is congruent to -1, and $(-1)^2 = 1$.) The element $m - 1$ is marked with a gray background. Whenever the order of an element in a cyclic group is even, you encounter $m - 1$ at half its order. No other element can have an order of two as the group wouldn't be cyclic otherwise. As mentioned in the previous point, not all multiplicative groups are cyclic. When calculating modulo 24, for example, all elements except the identity element have an order of two and are thus their own inverse.
- **Subgroups of prime order**: You cannot construct a group with a prime order greater than two with the above tool because the number of coprime elements below any modulus is always even. When a prime order is desirable for cryptographic applications, you have to resort to subgroups. For example, many elements generate a subgroup of order 11 when you set the modulus to 23.
- **Symmetry of even columns**: Since $(-A)^e = A^e$ whenever the exponent e is even, even columns are palindromes (i.e. ↓ = ↑).

▼ All subgroups of cyclic groups are cyclic

Given a generator G which generates the cyclic group $\mathbb{G} = \langle G \rangle$ of order $n = |\mathbb{G}| = |G|$, there exists a single subgroup of order d for every divisor d of n. We prove this statement by showing that such a subgroup exists and that it is unique:

Multiplicative | Additive | Both

Multiplicative

- **Existence**: Let $c = n/d$, then $\langle G^c \rangle = \{G^c, (G^c)^2, \ldots, (G^c)^d = I\}$ is a cyclic subgroup of order d.
- **Uniqueness**: Given a subgroup $\mathbb{H} = \{A_1, \ldots, A_d\}$ of order d, we show that there is an integer b for each $A \in \mathbb{H}$ so that $A = (G^c)^b$, which implies that $\mathbb{H} = \langle G^c \rangle$. Since G generates all elements of $\mathbb{G}$, there exists an integer a so that $A = G^a$. Since a subgroup is a group, the order of A must divide d according to Lagrange's theorem. Therefore, $A^d = (G^a)^d = I$. Since the order of G is n, $a \cdot d$ must be a multiple of n, which means that there exists an integer b so that $a \cdot d = b \cdot n = b \cdot (c \cdot d)$. It follows that $a = b \cdot c$ and $A = G^a = (G^c)^b$.

Additive

- **Existence**: Let $c = n/d$, then $\langle cG \rangle = \{cG, 2(cG), \ldots, d(cG) = O\}$ is a cyclic subgroup of order d.
- **Uniqueness**: Given a subgroup $\mathbb{H} = \{A_1, \ldots, A_d\}$ of order d, we show that there is an integer b for each $A \in \mathbb{H}$ so that $A = b(cG)$, which implies that $\mathbb{H} = \langle cG \rangle$. Since G generates all elements of $\mathbb{G}$, there exists an integer a so that $A = aG$. Since a subgroup is a group, the order of A must divide d according to Lagrange's theorem. Therefore, $dA = d(aG) = O$. Since the order of G is n, $a \cdot d$ must be a multiple of n, which means that there exists an integer b so that $a \cdot d = b \cdot n = b \cdot (c \cdot d)$. It follows that $a = b \cdot c$ and $A = aG = b(cG)$.

For example, 2 generates the multiplicative group modulo 13 of order 12. A subgroup of order 6 is generated by $2^{12/6} =_{13} 4$. Another subgroup of order 6 may contain $9 =_{13} 2^8$. In this case, $b = \frac{a \cdot d}{n} = \frac{8 \cdot 6}{12} = 4$. It follows that $4^4 =_{13} 9$, which is true.

To get a better feeling for where an element with a certain order may or may not appear, you can enable the "Order" option.

Subgroup cosets

The following tool shows the cosets of the subgroup generated by the given element (see the additive groups for explanations):

Modulus: 11 | Next prime | Previous prime | Element: 3 | Unique: | Delay: 0.5

1	**3**	**9**	**5**	**4**
2	6	7	10	8
3	9	5	4	1
4	1	3	9	5
5	4	1	3	9
6	7	10	8	2
7	10	8	2	6
8	2	6	7	10
9	5	4	1	3

1	**3**	**9**	**5**	**4**
10	8	2	6	7

▼ **Non-cyclic subgroups of non-cyclic groups**

Subgroups don't have to be generated by a single element. For example, $\{1, 4, 11, 14\}$ form a multiplicative group modulo 15:

$$4 \cdot 4 =_{15} 1$$
$$4 \cdot 11 =_{15} 14$$
$$4 \cdot 14 =_{15} 11$$
$$11 \cdot 11 =_{15} 1$$
$$11 \cdot 14 =_{15} 4$$
$$14 \cdot 14 =_{15} 1$$

Since each element is its own inverse, none of the elements generates the whole subgroup. Its cosets still partition the non-cyclic multiplicative group of coprime elements modulo 15:

1	**4**	**11**	**14**
2	8	7	13

The unique cosets of {1, 4, 11, 14} modulo 15.

The above tool just doesn't support this case because we're usually interested only in repetitions of a single element.

Group notations

The multiplicative group of integers modulo some number m is usually denoted as $(\mathbb{Z}/m\mathbb{Z})^\times$ or $\mathbb{Z}_m^\times$. As mentioned previously, the former notation is derived from the quotient ring $\mathbb{Z}/m\mathbb{Z}$. Both notations imply that the integers below the modulus are restricted to those which have an inverse. Cryptographers often use the latter notation with an asterisk as the multiplication sign, i.e. $\mathbb{Z}_m^*$.

Fermat's little theorem

Fermat's little theorem states that for any integers a and p, if p is prime and a is not a multiple of p, then $a^{p-1} =_p 1$. This theorem is an instantiation of the second consequence of Lagrange's theorem, which states that if you repeat an element of a group as many times as there are elements in the group, you get the identity element of the group. The theorem is named after Pierre de Fermat (1607 − 1665). There is no Fermat's big theorem, but several theorems are named after him. We will generalize Fermat's little theorem to non-prime integers in the next chapter.

Modular multiplicative inverse

As discussed earlier, you can compute the inverse of an element by repeating it one time less than its order. Using Fermat's little theorem, we can compute the multiplicative inverse modulo a prime number p for any element A as $A^{-1} =_p A^{p-2}$ because $A \cdot A^{p-2} =_p A^{p-1} =_p 1$. As we'll see in the next chapter, there's a faster way to compute the multiplicative inverse of an element. In cryptography, speed is not the only consideration, though; you also want to avoid side-channel attacks. If you want to compute the multiplicative inverse of an element in constant time, you might still prefer to compute A^{-1} as A^{p-2}.

Discrete-logarithm problem (DLP)

A lot of modern cryptography is built on the assumption that computing the discrete logarithm in multiplicative groups is infeasible on classical computers if you choose the modulus m carefully. In other words, while computing $K =_m G^k$ given a generator G and an exponent k is easy (we have seen that the running time is proportional to the number of bits in k), computing $k = \log_G(K)$ as the logarithm of K to the base G is presumably hard. If this assumption is indeed true, modular exponentiation is a linear one-way function. As mentioned earlier, it is widely believed that this is the case, but we still lack a proof for the hardness of the discrete-logarithm problem. We'll study the best known algorithms for solving the discrete-logarithm problem in the second to last chapter.

▼ **Quantum computers**

Quantum computers perform calculations using the physical properties of quantum mechanics, such as superposition. The discrete-logarithm problem can be solved efficiently on a sufficiently powerful quantum computer with Shor's algorithm, named after Peter Shor (born in 1959). The quantum computers which have been built until now are by far not powerful enough to break modern cryptography, but since this might change in the future, cryptography which cannot be broken by quantum computers – so-called post-quantum cryptography – is an active area of research with increasing interest.

Prime numbers

We've talked quite a bit about prime numbers already. The goal of this chapter is to give you a better understanding about primality and related concepts such as factorization. From here to the end of this article, the topics get more and more advanced. None of the remaining topics are required to get a basic understanding of modern cryptosystems. This chapter fills some gaps from previous chapters, such as why only coprime integers have a multiplicative inverse, generalizes some results, such as Fermat's little theorem to Euler's theorem, and discusses some algorithmic aspects, such as how to compute the multiplicative inverse and how to test whether an integer is prime. The chapters afterwards build towards elliptic curves, which are another popular way to construct a linear one-way function.

▼ **Formatting preferences**

All tools in this chapter can handle arbitrarily large integers, which are easier to read when their digits are grouped with a delimiter. I prefer the apostrophe as a thousand separator, but other people have different preferences. From time to time, you may also want to copy numbers or expressions to other calculators or to source code, in which case it is best if the number or expression isn't formatted at all. (Many programming languages actually allow underscores in integer literals.) For these reasons, I decided to make the outputs of my tools fully configurable, including the signs of various operations:

Decimal separator:	' (apostrophe, U+0027)	Division sign:	/ (slash, U+002F)
Hexadecimal separator:	_ (space, U+0020)	Modulo sign:	% (percent sign, U+0025)
Minus sign:	− (minus sign, U+2212)	Exponentiation sign:	(none, raise exponent)
Multiplication sign:	· (middle dot, U+00B7)		

Decimal integer:	1'234'567'890'987'654'321
Hexadecimal integer:	0x112210F4 B16C1CB1
Equation:	$1 + 2 - 3 \cdot 4 / 5 \% 6^7$

(The hexadecimal characters following the `U+` in the parentheses indicate the Unicode code point of the corresponding symbol.)

Prime factorization

Every positive integer is a multiple of 1 and itself. Many positive integers are also a multiple of other positive integers, which means that they can be divided by an integer between 1 and themselves without a remainder. Such integers can therefore be written as a product of two factors, which are both smaller than their product. Positive integers which can be split into a product of two smaller positive integers are called composite. All other positive integers except 1 are called prime. The process of splitting a composite number into a product of smaller integers is called integer factorization. As long as one of the factors is still composite, you can continue the factorization until all factors are prime, which is known as prime factorization. Since the factors are smaller than the integer which is factorized but cannot get smaller than 2, the factorization into primes terminates after a finite number of steps for every integer. To find the factors of an integer, you can simply try all possible factors until the square root of the given integer. This algorithm is called trial division and proves constructively that every positive integer can be written as a product of primes. We'll prove later that the factorization into primes is unique (up to the order of the primes). Unfortunately, the number of integers between 2 and the square root of the given integer scales exponentially in the number of bits, and hence trial division isn't very useful in practice. There are faster factoring algorithms (we'll study one of them in the second to last chapter), but similar to the discrete-logarithm problem, no algorithm is known for factoring (large) integers efficiently on classical computers. The following tool factorizes integers into primes using trial division with the configured delay between attempts.

Integer: 231 Totients: Delay: 0.20 Factorize

6 steps in 2.65 s

231 = 3 · 7 · 11

Clear

▼ **The integer 1**

By convention, 1 is not included in the set of prime numbers in order to make statements involving prime numbers simpler.

▼ **Euclid's theorem**

Euclid's theorem states that there are infinitely many prime numbers. It is named after Euclid, who proved it around 300 BCE. If there were only a finite number of primes, then the product of all primes plus 1 cannot be divided without remainder by any of the prime numbers. The new number is therefore either prime or has prime factors which were not accounted for. Since this is a contradiction, there has to be an infinite number of primes.

▼ **Smooth numbers**

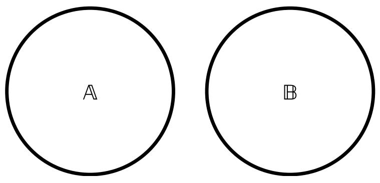
An integer is called n-smooth if none of its prime factors are greater than n. As you can see when playing with the above tool, factoring large numbers is easy if most of their factors are sufficiently small. Since this is often undesirable in cryptographic applications, you have to choose the delicate parameters carefully.

Multiset of prime factors

Since each number can be written as a product of prime factors, we can represent each number by the multiset of its prime factors. Unlike a set, a multiset can contain an element multiple times. The number of times that an element is included in a multiset is called the multiplicity of the element in the multiset. We can extend the following operations and relations from set theory to multisets:

- **Union**: The union $\mathbb{C}$ of the multisets $\mathbb{A}$ and $\mathbb{B}$ contains each element contained in $\mathbb{A}$ or $\mathbb{B}$ with the higher of the two multiplicities. Using the notation from sets, we write this as $\mathbb{C} = \mathbb{A} \cup \mathbb{B}$. For example, $\{2, 2, 3\} \cup \{2, 5\} = \{2, 2, 3, 5\}$.
- **Intersection**: The intersection $\mathbb{C}$ of the multisets $\mathbb{A}$ and $\mathbb{B}$ contains each element contained in both $\mathbb{A}$ and $\mathbb{B}$ with the lower of the two multiplicities. Using the notation from sets, we write this as $\mathbb{C} = \mathbb{A} \cap \mathbb{B}$. For example, $\{2, 2, 3\} \cap \{2, 5\} = \{2\}$.
- **Inclusion**: The multiset $\mathbb{B}$ includes the multiset $\mathbb{A}$ if every element contained in $\mathbb{A}$ is also contained in $\mathbb{B}$ with the same or a higher multiplicity. We write this as $\mathbb{A} \subseteq \mathbb{B}$ or, if at least one element has a higher multiplicity in $\mathbb{B}$ than in $\mathbb{A}$, as $\mathbb{A} \subset \mathbb{B}$. For example, $\{2, 2\} \subset \{2, 2, 3\}$.

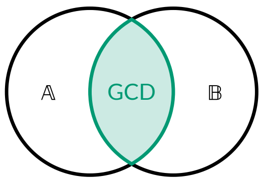
Unlike tuples and lists, the order of the elements in multisets matters neither for comparison nor the above operations. We say that an integer a is coprime with an integer b (or, equivalently, that a is relatively prime to b) if a and b have no prime factor in common. Using the name of an integer in blackboard bold to denote the multiset of its prime factors, we can write this as $\mathbb{A} \cap \mathbb{B} = \varnothing$, where $\varnothing$ denotes the empty multiset. Coprimality can be visualized with a Venn diagram, named after John Venn (1834 – 1923), like this:



The prime factors of two coprime integers are disjoint.

Greatest common divisor (GCD)

The greatest common divisor of two integers is the largest positive integer which divides both integers without a remainder. Since the factorization into primes is unique, as we will prove soon, the prime factors of any divisor of an integer a are included in the prime factors of a. (If this wasn't the case, the factorization of a wouldn't be unique.) The greatest common divisor of any two integers thus corresponds to the intersection of their prime factors because any larger integer no longer divides both integers:

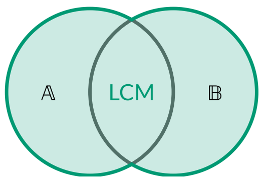


GCD = intersection of prime factors

We write the greatest common divisor of the integers a and b as $\gcd(a, b)$. Any integer which divides both a and b also divides $\gcd(a, b)$. The only and thus the greatest common divisor of two coprime integers is 1. As we will see in a moment, the greatest common divisor of two integers can be computed efficiently without having to determine the prime factors of both integers first.

Least common multiple (LCM)

The least common multiple of two positive integers is the smallest positive integer which is a multiple of both integers. The product of all the numbers in the union of the prime factors of both integers is the smallest number which is a multiple of both integers:



LCM = union of prime factors

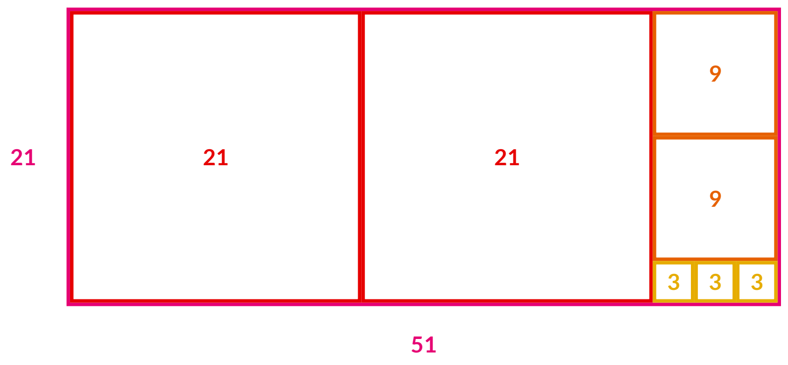
We write the least common multiple of the integers a and b as $\operatorname{lcm}(a, b)$. Since the product of a and b contains the prime factors in the intersection twice, you get the least common multiple when you divide the product by the greatest common divisor of a and b:

$$\operatorname{lcm}(a, b) = \frac{a \cdot b}{\gcd(a, b)}$$

If a and b are coprime, then $\gcd(a, b) = 1$ and $\operatorname{lcm}(a, b) = a \cdot b$. (Unlike Wikipedia, we don't care about negative inputs here.)

Euclidean algorithm

The greatest common divisor can also be interpreted geometrically: Given a rectangle, what is the largest square which tiles the rectangle in both dimensions without remainders? If you want to tile a rectangular bathroom which is 51 units wide and 21 units deep, for example, which square tiles should you order so that you don't have to break any tiles and have as little work as possible? In order to find the answer, we can make the problem smaller and smaller. Any square which divides some length in one dimension also divides the same length in the other dimension. This means that we can subtract the smaller side from the larger side without affecting the solution. Given that our room is 21 units deep, we know that there has to be a boundary between tiles 21 units in from the left. This reduces the problem to a rectangle which is 30 units wide and 21 units deep. We can repeat this procedure of splitting off squares from our rectangle until we're left with a square. The side of this square corresponds to the greatest common divisor of the two numbers that we started with. In our example, a square whose sides are 3 units is the largest square which tiles the room:



From the pink room, we split off the red squares and the orange squares to get the yellow squares.

This procedure for determining the greatest common divisor of two numbers is called the Euclidean algorithm. It is named after the Greek mathematician Euclid, who described it around 300 BCE. A recursive implementation of the algorithm looks as follows:

$$\gcd(a, b) = \begin{cases} a & \text{if } a = b, \\ \gcd(a - b, b) & \text{if } a > b, \\ \gcd(a, b - a) & \text{if } a < b. \end{cases}$$

The algorithm is guaranteed to terminate because one of the two parameters gets smaller in every iteration and the parameters never get smaller than 1 since 1 divides all integers. Instead of subtracting the smaller number from the larger number one by one, we can use the modulo operation to subtract the smaller number from the larger number as many times as it takes to make the larger number smaller than the smaller number in one go. The Euclidean algorithm is typically implemented as follows:

```
function gcd(a, b) {
    while (b != 0) {
        let t = b;
        b = a % b;
        a = t;
    }
    return a;
}
```

Since this is valid JavaScript, you can copy the code to the developer tools of your browser and verify that `gcd(51, 21)` is indeed `3`. `t` is just a temporary variable to swap the values of `a` and `b`. If `b` is greater than `a` initially, the first round results only in the swapping of `a` and `b`. Unlike integer factorization, the Euclidean algorithm is efficient and can thus be computed even for very large integers.

Extended Euclidean algorithm

When calculating the greatest common divisor of any positive integers a and b with the Euclidean algorithm, we can keep track of how we combined these two numbers to get the current intermediate value. Instead of subtracting numbers from one another, we subtract equations from one another so that we end up with coefficients for a and b which result in the greatest common divisor:

$$\begin{aligned} a &= 1 \cdot a + 0 \cdot b \\ b &= 0 \cdot a + 1 \cdot b \\ &\vdots \\ \gcd(a, b) &= c \cdot a + d \cdot b \end{aligned}$$

This procedure is known as the extended Euclidean algorithm, and the equation we end up with is known as Bézout's identity. The following tool visualizes the extended Euclidean algorithm with a table. The first two rows after the column titles contain the initialization of the algorithm as in the equations above this paragraph, where the larger number is taken first. The "remainder" column contains the values of the Euclidean algorithm; the two columns after that contain the coefficients of the larger and the smaller input. The "quotient" column indicates how many times the current row is subtracted from the row above. The last remainder before reaching zero is the greatest common divisor of the two inputs. We'll study the multiplicative inverse below.

Integer a: 51   Integer b: 21

Step	Quotient	Remainder	=	… · 51	+	… · 21
		51		1		0
	– 2 ·	21		0		1
1	– 2 ·	9		1		–2
2	– 3 ·	3		–2		5
3		0		7		–17

Greatest common divisor: gcd(51, 21) = 3

Least common multiple: lcm(51, 21) = 51 · 21 / 3 = 357

Multiplicative inverse: [does not exist]

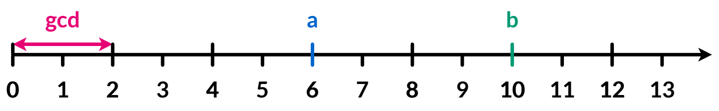
Bézout's identity: 3 = (–2) · 51 + 5 · 21

Bézout's identity

Bézout's identity, named after Étienne Bézout (1730 – 1783), makes the following two statements:

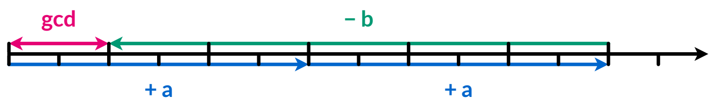
- **Existence**: For any two positive integers a and b, there exist some integers c and d so that $\gcd(a, b) = c \cdot a + d \cdot b$. The extended Euclidean algorithm proves constructively that such coefficients c and d exist. These coefficients are not unique, though. Since the coefficients in the last row of the tool above add up to 0 when multiplied by the corresponding input, you can add multiples of the last row to the second to last row without affecting the left side of the equation. For example, as $3 = (-2) \cdot 51 + 5 \cdot 21$ and $0 = 7 \cdot 51 + (-17) \cdot 21$, we also have that $3 = (-2 + 7) \cdot 51 + (5 - 17) \cdot 21 = 5 \cdot 51 + (-12) \cdot 21$.
- **Multiples**: Every linear combination of a and b is a multiple of $\gcd(a, b)$, and every multiple of $\gcd(a, b)$ can be expressed as a linear combination of a and b. Denoting $\gcd(a, b)$ as g, we can write this as $\{e \cdot a + f \cdot b \mid e, f \in \mathbb{Z}\} = \{h \cdot g \mid h \in \mathbb{Z}\}$. The second half of this statement is an immediate consequence of the previous point: Since $g = c \cdot a + d \cdot b$ for some integers c and d, any multiple $h \cdot g = (h \cdot c) \cdot a + (h \cdot d) \cdot b$ due to distributivity, which is a linear combination of a and b. We prove the first half of the statement with the following two lemmas:
  - Let $\mathbb{I} = \{e \cdot a + f \cdot b \mid e, f \in \mathbb{Z}\}$ and i be the smallest positive element in $\mathbb{I}$. Claim: All elements in $\mathbb{I}$ are a multiple of i. Proof: Using Euclidean division, every element j in $\mathbb{I}$ can be written as $j = q \cdot i + r$ for some integers q and r with $0 \leq r < i$. Since i and j are in $\mathbb{I}$, which means that $i = i_a \cdot a + i_b \cdot b$ and $j = j_a \cdot a + j_b \cdot b$ for some $i_a, i_b, j_a, j_b \in \mathbb{Z}$, so is $r = j - q \cdot i = (j_a \cdot a + j_b \cdot b) - q \cdot (i_a \cdot a + i_b \cdot b) = (j_a - q \cdot i_a) \cdot a + (j_b - q \cdot i_b) \cdot b \in \mathbb{I}$. Since $0 \leq r < i$ and i is by assumption the smallest positive element in $\mathbb{I}$, r must be 0. Therefore, every $j \in \mathbb{I}$ is indeed a multiple of i. (As we will see later, $\mathbb{I}$ is a so-called ideal.)
  - Claim: The smallest positive element $i \in \mathbb{I}$ is the greatest common divisor of a and b. Proof: Since all linear combinations of a and b are multiples of i according to the first lemma, both a and b are multiples of i, which means that i is a common divisor of both a and b. Any integer which divides both a and b also divides every integer of the form $e \cdot a + f \cdot b$ due to distributivity. Since i is of this form, all common divisors divide i. Therefore, i is the greatest common divisor of a and b.

The second part of Bézout's identity matches our earlier observation that all modular multiples of a number are spaced equally. You can also think of the greatest common divisor of a and b as the biggest unit in which a and b can be expressed without fractions. Just as you can't get fractions when adding and subtracting integers, you can't get fractions of this unit when you add and subtract only multiples of it. I use distance to visualize a and b, but you can apply this idea to any quantity, including mass, volume, and time.



You cannot get an odd number when adding and subtracting two even numbers.

The first part of Bézout's identity means that we can measure this unit when given only prototypes for the quantities a and b:



The extended Euclidean algorithm tells us that gcd(10, 6) = 2 = 2 · 6 – 1 · 10.

Euclid's lemma

Euclid's lemma states that if a prime p divides the product $a \cdot b$ of some integers a and b, then p divides a or b (or both). Clearly, if p divides a, the statement is true. So all we need to show is that if p does not divide a, then p divides b. Since p is prime and a is not a multiple of p, p and a are coprime. Using Bézout's identity, we know that there are some integers c and d so that $c \cdot p + d \cdot a = 1$. When we multiply this equation by b, we get $c \cdot p \cdot b + d \cdot a \cdot b = b$. Since the previous statement is an implication, we can assume the so-called antecedent (that p divides $a \cdot b$) when proving the so-called consequent (that p divides b). Therefore, b is a multiple of p because both $c \cdot p \cdot b$ and $d \cdot a \cdot b$ are multiples of p. (If you add multiples of a number, you get another multiple due to distributivity: $e \cdot p + f \cdot p = (e + f) \cdot p$ for any e and f.)

▼ **Unique factorization theorem**

The unique factorization theorem, which is also known as the fundamental theorem of arithmetic, states that every integer greater than 1 can be written uniquely – up to the order of the factors – as a product of prime numbers. It makes two claims:

- **Existence**: All integers greater than 1 can be factorized into primes because each of them is either composite or prime.
- **Uniqueness**: Suppose that there are integers greater than 1 which have distinct prime factorizations. Let n be the smallest such integer so that $n = p_1 \cdot p_2 \cdot \ldots \cdot p_k = q_1 \cdot q_2 \cdot \ldots \cdot q_l$, where each p_i and each q_j is prime. Since p_1 divides $n = q_1 \cdot q_2 \cdot \ldots \cdot q_l$, it has to divide one of the q_js according to (a recursive application of) Euclid's lemma. Since we don't care about the order of the q_js, let's assume that p_1 divides q_1. Since both p_1 and q_1 are prime and greater than 1, it follows that $p_1 = q_1$. When we cancel these factors on both sides of the equation, we get $m = p_2 \cdot \ldots \cdot p_k = q_2 \cdot \ldots \cdot q_l$. Given that the two factorizations of n were distinct, these two factorizations of m have to be distinct as well. Since $m < n$, n cannot be the smallest integer with distinct prime factorizations, which is a contradiction. Thus, each positive integer can be represented by a unique multiset of prime factors.

Multiplicative inverse revisited

Coming back to multiplicative groups modulo some integer m for a moment: Why is it that only those integers have a multiplicative inverse which are coprime with the group's modulus? Coprimality is necessary and sufficient for an integer to have an inverse since:

- **Sufficient**: If an integer a is coprime with the modulus m, Bézout's identity tells us that there are two integers b and n so that $b \cdot a + n \cdot m = 1$. This equation means that $b \cdot a$ equals 1 up to a multiple of m, that is $b \cdot a =_m 1$. b is therefore the multiplicative inverse of a modulo m. The extended Euclidean algorithm can be used to determine b. This approach is usually much faster than using exponentiation. The above tool colors the coefficient of the smaller number blue whenever the greatest common divisor is 1. If you use the extended Euclidean algorithm only to determine the multiplicative inverse of a, you don't need to keep track of the coefficient of m (i.e. you don't have to implement the second to last column).
- **Necessary**: The second part of Bézout's identity tells us that the closest you can get to a multiple of m without reaching it is $\gcd(a, m)$. If an integer a is not coprime with the modulus m, $\gcd(a, m) > 1$. Consequently, there is no integer b so that $b \cdot a =_m 1$. For example, if both a and m are even, which means that their greatest common divisor is a multiple of 2, you cannot reach an odd number, including 1, by adding a repeatedly to itself.

There is another important question to be answered: Why are multiplicative groups closed if you limit the elements to the integers which are coprime with the modulus? The best way to think about this is with multisets of the prime factors. When you multiply two integers, you combine the multisets of their prime factors by adding the multiplicities of each element. For example, $6 \cdot 15 = \{2, 3\} + \{3, 5\} = \{2, 3, 3, 5\} = 90$. If neither a nor b has prime factors in common with some integer m, then neither has $a \cdot b$. Calculating the product modulo m doesn't change this fact because the congruence property of modular multiplication means that you can reduce integers to their remainder, not that you have to. After all, the Euclidean algorithm showed us that subtracting the smaller number m repeatedly from the larger number $a \cdot b$ does not change the greatest common divisor of the two numbers.

Since the order of a group plays a crucial role in cryptography, our next goal is to find a formula which counts the number of coprime integers smaller than m. On our way there, we'll study another famous result from number theory: the Chinese remainder theorem.

▼ **Least common multiple and 0**

There is another way to see why elements which share a prime factor with the modulus are undesirable. If the integer a is not coprime with the modulus m, then $\gcd(a, m) > 1$ and $\operatorname{lcm}(a, m) < a \cdot m$. This means that there is some integer $b < m$ for which $b \cdot a = \operatorname{lcm}(a, m)$. Since $\operatorname{lcm}(a, m)$ is also a multiple of m, $b \cdot a =_m 0$. As I explained when I introduced multiplicative groups, 0 has to be excluded from the set of elements. Since a group has to be closed, we have to exclude a or b. This problem can be observed in the operation table of multiplicative groups. If we set the modulus to 12, for example, we see that all the elements which are not coprime with the modulus (the tool marks them with a blue background) reach 0 at some point. After that, the elements repeat because $(b + 1) \cdot a =_m b \cdot a + 1 \cdot a =_m 0 + a =_m a$. The first factor for which we reach 0 is $b = \frac{\operatorname{lcm}(a,m)}{a} = \frac{m}{\gcd(a,m)}$. This means that there are $\gcd(a, m)$ repetitions in the rows and columns of non-coprime elements. It also implies that the equation $a \cdot x =_m c$ has $d = \gcd(a, m)$ solutions in $\mathbb{Z}_m$ if c is a multiple of d and no solution otherwise.

▼ **Uniqueness of the multiplicative inverse**

As required by the group axioms, the multiplicative inverse of a coprime element is unique. Suppose that the coprime element a has two multiplicative inverses: $b_1 \cdot a =_m 1$ and $b_2 \cdot a =_m 1$. It follows that $b_1 \cdot a =_m b_2 \cdot a$ and hence that $b_1 \cdot a - b_2 \cdot a =_m 0$. This means that $b_1 \cdot a - b_2 \cdot a = (b_1 - b_2) \cdot a$ is a multiple of m. Since $\gcd(a, m) = 1$, m has to divide $b_1 - b_2$ according to a generalized version of Euclid's lemma. (Its proof required only that p and a are coprime, which means that p has to be only relatively prime to one of the factors, not "absolutely" prime.) Therefore, b_1 has to equal b_2 up to a multiple of m.

Chinese remainder theorem (CRT)

The Chinese remainder theorem states that if the remainders of an integer are known relative to several moduli which are coprime with one another, the integer is unique up to multiples of the product of the moduli. More formally, the system of congruences

$$\begin{gathered} x =_{m_1} r_1 \\ x =_{m_2} r_2 \\ \vdots \\ x =_{m_l} r_l \end{gathered}$$

has a unique solution x with $0 \leq x < M$ if $\gcd(m_i, m_j) = 1$ for all $i \neq j$, where $M = \prod_{i=1}^{l} m_i$. The problem was first stated in a Chinese text around the 4th century CE. The solution can be determined as follows: Let $M_i = M / m_i$, which is the product of all moduli except m_i. Given that m_i has no prime factor in common with any of the other moduli, M_i and m_i are coprime. Using the extended Euclidean algorithm, we find an N_i so that $N_i \cdot M_i =_{m_i} 1$. As M_i is a multiple of all other moduli, $N_i \cdot M_i =_{m_j} 0$ for all $j \neq i$. Therefore, $x =_M \sum_{i=1}^{l} r_i \cdot N_i \cdot M_i$ satisfies all congruences. The solution is unique because for any two solutions, $x_1 - x_2 =_{m_i} 0$ for all i and thus also $x_1 - x_2 =_M 0$ because any multiple of coprime factors is also a multiple of their product.

The following tool implements this procedure for $l = 2$. If you want to solve a larger system of congruences, you can repeatedly replace any two equations with their solution until only a single congruence is left, which is the solution to the whole system. The case of $l = 2$ is special because $M_1 = m_2$ and $M_2 = m_1$, which means that a single Bézout's identity is enough to find the solution:

Modulus m1:	5	Remainder r1:	3
Modulus m2:	7	Remainder r2:	4

Problem: $x =_{m_1} r_1 =_5 3$
$x =_{m_2} r_2 =_7 4$

Bézout's identity: $1 = N_1 \cdot m_2 + N_2 \cdot m_1$
$= (-2) \cdot 7 + 3 \cdot 5$

Solution: $x =_{m_1 \cdot m_2} r_1 \cdot N_1 \cdot m_2 + r_2 \cdot N_2 \cdot m_1$
$=_{35} 3 \cdot (-2) \cdot 7 + 4 \cdot 3 \cdot 5$
$=_{35} 18$

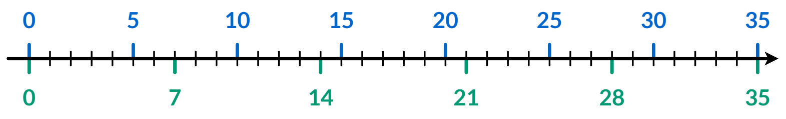
▼ **Sum and product of similar terms**

I used the following two shortcuts in the description of the Chinese remainder theorem, which you might not be familiar with:

- **Capital-sigma notation**: $\sum_{i=1}^{l} r_i \cdot N_i \cdot M_i = r_1 \cdot N_1 \cdot M_1 + \ldots + r_l \cdot N_l \cdot M_l$, and
- **Capital-pi notation**: $\prod_{i=1}^{l} m_i = m_1 \cdot \ldots \cdot m_l$, where i is the index which is incremented.

▼ **Signals of different frequencies**

The Chinese remainder theorem is more intuitive than it may seem. Given two pulses with coprime repetition frequencies, it's clear that the phase of one pulse shifts relative to the phase of the other pulse until they re-align at the least common multiple:



Visualizing shifting repetitions on a number line with $m_1 = 5$ and $m_2 = 7$.

The offset between the two pulses has to be different throughout this cycle. If the same offset occurred twice, the difference between the two occurrences would be a multiple of both frequencies smaller than the least common multiple, which is a contradiction. (This argument is similar to the one we used to establish that you reach the identity element before any of the group's elements can appear again when repeating an element.) This is also why all (and only) the coprime elements generate the additive group. In the above example, there are 5 offsets to cover and 5 green ticks before the cycle repeats. Since no two offsets may be the same, all 5 offsets are covered due to the pigeonhole principle. (On the other side of the number line, there are 7 offsets to cover with 7 blue ticks.) The difference between the two remainders in the tool above determines into which repetition the solution falls. For example, if the first remainder is one smaller than the second remainder, the solution is $15 + r_1$.

Group isomorphisms

A group isomorphism is an invertible function which maps the elements of one group to distinct elements of another group so that the result of combining any two elements in one group matches the result when combining the paired elements in the other group. More formally, given two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with corresponding operations $\circ_1$ and $\circ_2$, an invertible function $f\colon \mathbb{G}_1 \to \mathbb{G}_2$ defines a group isomorphism if and only if $f(A \circ_1 B) = f(A) \circ_2 f(B)$ for all $A, B \in \mathbb{G}_1$. A group isomorphism can be visualized as follows:

$$\begin{tikzcd}
\mathbb{G}_1\colon & \boxed{A_1} \arrow[d, leftrightarrow] & \circ_1 & \boxed{B_1} \arrow[d, leftrightarrow] & = & \boxed{C_1} \arrow[d, leftrightarrow] \\
\mathbb{G}_2\colon & \boxed{A_2} & \circ_2 & \boxed{B_2} & = & \boxed{C_2}
\end{tikzcd}$$

An isomorphism between two groups.

You may have realized that this looks similar to our definition of a linear one-way function. The main differences are that the inputs to an isomorphism are the elements of a group, which don't have to be integers, and that the operation doesn't have to resemble addition. Moreover, an isomorphism may be easy to invert, thus neither "linear" nor "one-way" are appropriate adjectives.

If an isomorphism exists between the groups $\mathbb{G}_1$ and $\mathbb{G}_2$, the groups are said to be isomorphic, which is usually written as $\mathbb{G}_1 \cong \mathbb{G}_2$. For a function to be invertible, it has to map a single element of $\mathbb{G}_1$ to every element of $\mathbb{G}_2$. As a consequence, isomorphic groups have the same order ($|\mathbb{G}_1| = |\mathbb{G}_2|$), and the inverse of an isomorphism is also an isomorphism. Furthermore, an isomorphism maps the identity element of $\mathbb{G}_1$ to the identity element of $\mathbb{G}_2$ because $f(A) = f(A \circ_1 E) = f(A) \circ_2 f(E)$ for all $A \in \mathbb{G}_1$. It also maps the inverse $\overline{A}$ of each element $A \in \mathbb{G}_1$ to the inverse of $f(A)$ because $f(A) \circ_2 f(\overline{A}) = f(A \circ_1 \overline{A}) = f(E)$. Since you reach the identity element in one group only when you reach it in the other group, the order of mapped elements is the same: $|A| = |f(A)|$. Last but not least, either both groups are commutative or neither of them is. In conclusion, isomorphic groups represent the same structure, they just use different labels for the elements.

▼ Isomorphism of cyclic groups

All cyclic groups of the same order are isomorphic. As we saw earlier, additive groups are always cyclic. This means that all cyclic groups of order m are isomorphic to $\mathbb{Z}_m^+$ and thus commutative. Given a generator G, $\langle G \rangle \cong \mathbb{Z}_m^+$ if and only if $|G| = m$.

Multiplicative | Additive | Both

Multiplicative

The function $f(x) = G^x$ is a group isomorphism because:

$$\begin{tikzcd}
\mathbb{Z}_m^+\colon & \boxed{a} \arrow[d, leftrightarrow] & + & \boxed{b} \arrow[d, leftrightarrow] & = & \boxed{a + b} \arrow[d, leftrightarrow] \\
\langle G \rangle\colon & \boxed{G^a} & \cdot & \boxed{G^b} & = & \boxed{G^{a + b}}
\end{tikzcd}$$

An isomorphism with $\mathbb{Z}_m^+$ using multiplicative notation.

Additive

The function $f(x) = xG$ is a group isomorphism because:

$$\begin{tikzcd}
\mathbb{Z}_m^+\colon & \boxed{a} \arrow[d, leftrightarrow] & + & \boxed{b} \arrow[d, leftrightarrow] & = & \boxed{a + b} \arrow[d, leftrightarrow] \\
\langle G \rangle\colon & \boxed{aG} & + & \boxed{bG} & = & \boxed{(a + b)G}
\end{tikzcd}$$

An isomorphism with $\mathbb{Z}_m^+$ using additive notation.

In some groups, the inverse isomorphism is difficult to compute, giving us the linear one-way functions we were looking for.

Direct product

The Cartesian product of the l sets $\mathbb{G}_1$ to $\mathbb{G}_l$ is the set of l-tuples where the i-th component is an element of $\mathbb{G}_i$. More formally:

$$\mathbb{G}_1 \times \ldots \times \mathbb{G}_l = \{(A_1, \ldots, A_l) \mid A_1 \in \mathbb{G}_1, \ldots, A_l \in \mathbb{G}_l\}$$

The product is named after René Descartes (1596 – 1650). The product of l groups forms a group with the following operation:

$$(A_1, \ldots, A_l) \circ (B_1, \ldots, B_l) = (A_1 \circ_1 B_1, \ldots, A_l \circ_l B_l)$$

This so-called direct product of groups with such a component-wise operation fulfills all group axioms: It is closed and associative because the operation in each of the groups is closed and associative, the tuple consisting of each group's identity element is the identity element of the direct product, and each element of the direct product has a unique inverse consisting of the component-wise inverses. Being a Cartesian product, the order of the direct product equals the product of each group's order:

$$|\mathbb{G}_1 \times \ldots \times \mathbb{G}_l| = |\mathbb{G}_1| \cdot \ldots \cdot |\mathbb{G}_l|$$

The order of each element in the direct product is the least common multiple of the order of each component in its group:

$$|(A_1, \ldots, A_l)| = \mathrm{lcm}(|A_1|, \ldots, |A_l|)$$

Hence, the direct product is cyclic if and only if the individual groups are all cyclic and their orders are coprime with one another.

▼ **Internal direct product of commutative subgroups**

Given two subgroups $\mathbb{H}_1$ and $\mathbb{H}_2$ of a commutative group $\mathbb{G}$ which have no element in common other than the identity element E (i.e. $\mathbb{H}_1 \cap \mathbb{H}_2 = \{E\}$), the so-called internal direct product $\mathbb{H}_1 \circ \mathbb{H}_2 = \{A_1 \circ A_2 \mid A_1 \in \mathbb{H}_1, A_2 \in \mathbb{H}_2\}$ is isomorphic to the direct product $\mathbb{H}_1 \times \mathbb{H}_2$ (i.e. $\mathbb{H}_1 \times \mathbb{H}_2 \cong \mathbb{H}_1 \circ \mathbb{H}_2$). The isomorphism is given by the function $f\big((A_1, A_2)\big) = A_1 \circ A_2$, which maps elements of $\mathbb{H}_1 \times \mathbb{H}_2$ to elements of $\mathbb{H}_1 \circ \mathbb{H}_2$. In order to show that this is indeed an isomorphism, we need to show that f is compatible with the group operations (i.e. $f(A \circ_1 B) = f(A) \circ_2 f(B)$) and invertible:

- **Compatibility**: $f\big((A_1, A_2) \circ (B_1, B_2)\big) = f\big((A_1 \circ B_1, A_2 \circ B_2)\big) = (A_1 \circ B_1) \circ (A_2 \circ B_2) = (A_1 \circ A_2) \circ (B_1 \circ B_2) = f\big((A_1, A_2)\big) \circ f\big((B_1, B_2)\big)$ due to associativity and commutativity.
- **Invertibility** (a so-called bijection): The function f is invertible if the following two criteria are fulfilled:
  - **Distinct elements are mapped to distinct elements** (injectivity): This is the case if we obtain equal outputs only for equal inputs. In other words, we need to show that $f\big((A_1, A_2)\big) = f\big((B_1, B_2)\big)$ implies $(A_1, A_2) = (B_1, B_2)$. As $A_1 \circ A_2 = B_1 \circ B_2$, we have that $\overline{B_1} \circ A_1 = B_2 \circ \overline{A_2}$. Clearly, $\overline{B_1} \circ A_1 \in \mathbb{H}_1$ and $B_2 \circ \overline{A_2} \in \mathbb{H}_2$ due to closure. Since the identity element is the only element which is in both $\mathbb{H}_1$ and $\mathbb{H}_2$, $\overline{B_1} \circ A_1 = E$ and $B_2 \circ \overline{A_2} = E$. Thus, $A_1 = B_1$ and $A_2 = B_2$.
  - **Every element of the target group is reached** (surjectivity): Since every element in $\mathbb{H}_1 \circ \mathbb{H}_2$ is of the form $A_1 \circ A_2$ by definition, there's an input for every element in $\mathbb{H}_1 \circ \mathbb{H}_2$, namely (A_1, A_2) because $f\big((A_1, A_2)\big) = A_1 \circ A_2$.

▼ **Order when combining two elements of coprime orders**

Given a finite commutative group $\mathbb{G}$ with two elements A_1 and A_2 so that $\gcd(|A_1|, |A_2|) = 1$, the order of $A_1 \circ A_2$ is $|A_1| \cdot |A_2|$. Since A_1 generates the subgroup $\langle A_1 \rangle$ and A_2 generates the subgroup $\langle A_2 \rangle$, their intersection is a subgroup of both $\langle A_1 \rangle$ and $\langle A_2 \rangle$. By Lagrange's theorem, $|\langle A_1 \rangle \cap \langle A_2 \rangle|$ divides both $|\langle A_1 \rangle|$ and $|\langle A_2 \rangle|$. Since $\gcd(|\langle A_1 \rangle|, |\langle A_2 \rangle|) = 1$, $|\langle A_1 \rangle \cap \langle A_2 \rangle|$ has to equal 1. Since the identity element E is part of every subgroup, $\langle A_1 \rangle \cap \langle A_2 \rangle = \{E\}$. According to the previous box, $\langle A_1 \rangle \circ \langle A_2 \rangle$ is thus isomorphic to $\langle A_1 \rangle \times \langle A_2 \rangle$. Therefore, $|A_1 \circ A_2| = |(A_1, A_2)| = \mathrm{lcm}(|A_1|, |A_2|) = |A_1| \cdot |A_2|$.

Composite groups

Given the l coprime moduli $m_1, \ldots, m_l$ and their product $M = \prod_{i=1}^{l} m_i$, the additive group $\mathbb{Z}_M^+$ is isomorphic to the direct product of $\mathbb{Z}_{m_1}^+, \ldots, \mathbb{Z}_{m_l}^+$, written as follows:

$$\mathbb{Z}_M^+ \cong \mathbb{Z}_{m_1}^+ \times \ldots \times \mathbb{Z}_{m_l}^+$$

Let us verify this step by step. Since the order of additive groups is given by the modulus, both groups contain the same number of elements: $|\mathbb{Z}_M^+| = |\mathbb{Z}_{m_1}^+| \cdot \ldots \cdot |\mathbb{Z}_{m_l}^+| = M$. According to the Chinese remainder theorem, there's a unique element in $\mathbb{Z}_M^+$ for every element of $\mathbb{Z}_{m_1}^+ \times \ldots \times \mathbb{Z}_{m_l}^+$. As a consequence, the function $f(x) = (x \mathbin{\%} m_1, \ldots, x \mathbin{\%} m_l)$ is invertible. $f(x)$ is an isomorphism because modular equivalence is a congruence relation for any modulus: $f(A + B) = ((A + B) \mathbin{\%} m_1, \ldots, (A + B) \mathbin{\%} m_l) = (A \mathbin{\%} m_1, \ldots, A \mathbin{\%} m_l) + (B \mathbin{\%} m_1, \ldots, B \mathbin{\%} m_l) = f(A) + f(B)$ for any $A, B \in \mathbb{Z}_M^+$.

The same function is also an isomorphism for multiplicative groups because $(A \cdot B) \mathbin{\%} m_i =_{m_i} (A \mathbin{\%} m_i) \cdot (B \mathbin{\%} m_i)$:

$$\mathbb{Z}_M^\times \cong \mathbb{Z}_{m_1}^\times \times \ldots \times \mathbb{Z}_{m_l}^\times$$

Both groups contain the same number of elements because an integer is coprime with M if and only if it is coprime with all moduli $m_1, \ldots, m_l$. This gives us an easy way to determine the number of elements in multiplicative groups modulo a composite integer, as we shall see in the next section.

You can display the decomposition of each element modulo the prime factors of the modulus in the operation tables of additive groups and multiplicative groups by enabling the "Product" option. For example, $\mathbb{Z}_{35}^\times \cong \mathbb{Z}_5^\times \times \mathbb{Z}_7^\times$. Instead of multiplying two elements in $\mathbb{Z}_{35}^\times$, you can multiply their components in $\mathbb{Z}_5^\times$ and $\mathbb{Z}_7^\times$. For example, $[11 \leftrightarrow (1, 4)] \cdot [8 \leftrightarrow (3, 1)] = [18 \leftrightarrow (3, 4)]$. Decomposing a composite group into a direct product of its factors is another way to see why only coprime elements have an inverse because multiples of the prime factors are mapped to 0 in the corresponding group and 0 has no multiplicative inverse.

Euler’s totient function φ

As explained earlier, multiplicative groups contain all the integers between 0 and m which are coprime with the modulus m. So how many coprime integers below m are there? We answer this question by starting with the simplest case and generalizing from there:

- $m = p$ for some prime p: All integers strictly between 0 and p (i.e. $0 < x < p$) are coprime with p. Using the Greek letter phi to denote the function which returns the number of coprime integers smaller than its input, $\varphi(p) = p - 1$.
- $m = p^e$ for some prime p and a positive integer e: All integers between 0 and p^e except the multiples of p are coprime with p^e. Since $0 < c \cdot p < p^e$ for $0 < c < p^{e-1}$, there are $p^{e-1} - 1$ multiples of p which have to be excluded. Therefore, $\varphi(p^e) = (p^e - 1) - (p^{e-1} - 1) = p^e - p^{e-1} = p^{e-1} \cdot (p - 1)$.
- $m = p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}$ for l distinct primes with positive exponents: Since powers of different primes are coprime with one another, $\mathbb{Z}_m^\times \cong \mathbb{Z}_{p_1^{e_1}}^\times \times \ldots \times \mathbb{Z}_{p_l^{e_l}}^\times$ according to the previous section. Therefore, $\varphi(p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}) = \varphi(p_1^{e_1}) \cdot \ldots \cdot \varphi(p_l^{e_l})$.

Given the prime factorization of $m = p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}$, the number of coprime integers below m is given by the following formula:

$$\varphi(m) = \prod_{i=1}^{l} \varphi(p_i^{e_i}) = \prod_{i=1}^{l} p_i^{e_i} - p_i^{e_i - 1} = \prod_{i=1}^{l} p_i^{e_i - 1} \cdot (p_i - 1)$$

This is known as Euler’s totient function, named after Leonhard Euler (1707 – 1783). (Totient comes from the Latin “tot”, which means “that many”.) Since the computation of Euler’s totient function requires the prime factorization of the given input, I’ve included it in the prime factorization tool above; you just have to activate the “Totients” toggle.

▼ **Euler’s theorem**

Euler’s theorem is a generalization of Fermat’s little theorem: For any coprime integers a and m, we have that $a^{\varphi(m)} =_m 1$.

▼ **Number of generators in cyclic groups**

We saw that all cyclic groups are isomorphic to the additive group of the same order. Any element A that is coprime with the modulus m generates the additive group because mA is the first time you get a multiple of m. (If there was a smaller number n such that $nA =_m 0$, then the least common multiple of A and m would be smaller than mA, which means that their greatest common divisor would be larger than 1.) As a consequence, the number of generators of any cyclic group $\mathbb{G}$ is given by $\varphi(|\mathbb{G}|)$.

▼ **Sum of Euler’s totient function over divisors**

As shown earlier, all subgroups of cyclic groups are cyclic, and a unique subgroup exists for every divisor d of the group’s order n. Since subgroups are groups, the cyclic subgroup of order d has $\varphi(d)$ generators. This means that in a cyclic group of order n, there are exactly $\varphi(d)$ elements of order d for every divisor d of n. According to Lagrange’s theorem, each of the n elements has an order which divides n. This implies that the sum of the Euler’s totient function of every divisor d of n equals n:

$$\sum_{d \mid n} \varphi(d) = n, \text{where } \varphi(1) = 1.$$

If you enable the “Totient” option in the repetition tables above, you see the output of the Euler’s totient function for every divisor of the group’s order in a separate row. The sum of all the values in this row plus 1 for the identity element, whose count is covered by the label $\varphi(i)$, equals the number of elements in the group. Please note that the value in the totient row matches the number of elements with the given order only if the group is cyclic.

Carmichael’s totient function λ

As mentioned earlier, not all multiplicative groups are cyclic. We use the Greek letter lambda to denote the function which returns the largest order among the elements of the group: $\lambda(m) = \max_{A \in \mathbb{Z}_m^\times} |A|$. This is known as Carmichael’s (totient) function, named after Robert Daniel Carmichael (1879 – 1967). We shall see shortly that for prime powers, its value can be computed as follows:

$$\lambda(p^e) = \begin{cases} \frac{1}{2}\varphi(p^e) = 2^{e-2} & \text{if } p = 2 \text{ and } e > 2, \\ \varphi(p^e) = p^{e-1}(p-1) & \text{otherwise.} \end{cases}$$

For other integers, $\mathbb{Z}_m^\times$ is a composite group. Since the order of each element in $\mathbb{Z}_{p^e}^\times$ divides $\lambda(p^e)$ as we’ll see below, the largest element order in a direct product is the least common multiple of the largest element order in each of the composing groups:

$$\lambda(p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}) = \operatorname{lcm}(\lambda(p_1^{e_1}), \ldots, \lambda(p_l^{e_l}))$$

Given my definition of Carmichael's totient function as $\lambda(m) = \max_{A \in \mathbb{Z}_m^\times} |A|$, which is different from but equivalent to what you encounter elsewhere, and the definition of Euler's totient function as $\varphi(m) = |\mathbb{Z}_m^\times|$, the multiplicative group $\mathbb{Z}_m^\times$ is cyclic if and only if $\lambda(m) = \varphi(m)$. According to the first formula, $\lambda(p^e) = \varphi(p^e)$ if and only if $p \neq 2$ or $e \leq 2$. And since $\varphi(p^e) = p^{e-1}(p-1)$ is even for any prime power other than 2^1, $\mathrm{gcd}(\varphi(p_1^{e_1}), \varphi(p_2^{e_2})) \geq 2$ and thus $\mathrm{lcm}(\varphi(p_1^{e_1}), \varphi(p_2^{e_2})) < \varphi(p_1^{e_1}) \cdot \varphi(p_2^{e_2})$ if neither $p_1^{e_1}$ nor $p_2^{e_2}$ equals 2^1. Therefore, multiplicative groups are cyclic if and only if the modulus m equals $2, 4, p^e$, or $2p^e$ for a prime $p \neq 2$ and a positive integer e. You can verify this for moduli up to 100 with the repetition table tool. (This is why the tool displays the prime factorization of the modulus above the table.) You can compute Carmichael's totient function of an integer by enabling the "Totients" option of the factorization tool.

The math required to understand the formula for $\lambda(p^e)$ is a bit advanced, which is why I cover it in several information boxes below. In fact, it was rather difficult to find a satisfying explanation online. I adapted the following proof from Victor Shoup's book starting on the pages 160 and 203. You find an alternative approach here.

▼ Order of prime power

Multiplicative | Additive | Both

Multiplicative

For any element A of a group $\mathbb{G}$, if $A^{p^e} = I$ and $A^{p^{e-1}} \neq I$ for some prime p and a positive integer e, then $|A| = p^e$. The reason for this is that whatever the order of A is, it has to divide p^e given that $A^{p^e} = I$. Only powers of p divide p^e, but the order of A cannot have a smaller exponent than e because $A^{p^{e-1}} \neq I$.

Additive

For any element A of a group $\mathbb{G}$, if $p^e A = O$ and $p^{e-1} A \neq O$ for some prime p and a positive integer e, then $|A| = p^e$. The reason for this is that whatever the order of A is, it has to divide p^e given that $p^e A = O$. Only powers of p divide p^e, but the order of A cannot have a smaller exponent than e because $p^{e-1} A \neq O$.

▼ Exponent of a multiplicative group

The so-called exponent of the multiplicative group $\mathbb{Z}_m^\times$ is the smallest positive integer n so that $A^n =_m 1$ for all elements A of the group. Clearly, n divides the order of the group and is a multiple of the order of each element, including the largest one. In this box, we show that an element of order n exists, which implies that $\lambda(m) = n$ and $A^{\lambda(m)} =_m 1$ for any number A which is coprime with the modulus m. This is in fact how Carmichael's totient function is usually defined. Since $\lambda(m) \leq \varphi(m)$ for any positive integer m, Carmichael's totient function gives a tighter exponent than Euler's totient function in Euler's theorem.

Let $\prod_{i=1}^{l} p_i^{e_i}$ be the prime factorization of the exponent n. For each i between (and including) 1 and l, there exists an element $A_i \in \mathbb{Z}_m^\times$ so that $A_i^{n/p_i} \neq_m 1$. If no such A_i could be found for some i, n/p_i would be an exponent smaller than n which results in 1 for all elements of the group, which would contradict the minimality of n. Given such an element A_i for each i, the element $A_i^{n/p_i^{e_i}}$ has order $p_i^{e_i}$ because $(A_i^{n/p_i^{e_i}})^{p_i^{e_i}} =_m A_i^n =_m 1$ and $(A_i^{n/p_i^{e_i}})^{p_i^{e_i - 1}} =_m A_i^{n/p_i} \neq_m 1$ (see the previous box). As all the prime powers of a prime factorization are coprime with one another, the order of $\prod_{i=1}^{l} A_i^{n/p_i^{e_i}}$ is n according to an earlier theorem.

▼ Why multiplicative groups modulo a prime are cyclic

Let's start with the simplest case: $\mathbb{Z}_p^\times$ is cyclic if p is prime. In order to prove this, we need math which we haven't covered yet:

1. $\mathbb{Z}_p = \{0, \dots, p-1\}$ is a finite field over which polynomials can be defined. We'll discuss finite fields later in this article.
2. A non-constant single-variable polynomial of degree d over any field evaluates to 0 for at most d distinct inputs. In other words, $f(x) = \sum_{i=0}^{d} c_i x^i$ with some coefficients $c_0, \dots, c_d \in \mathbb{Z}_p$ where $d > 0$ and at least $c_d \neq 0$ has at most d roots. Confusingly enough, this statement is also known as Lagrange's theorem. When formulated over the field of complex numbers, it is also known as the fundamental theorem of algebra. We'll prove it in the article about coding theory.

We learned in the previous box that $A^{\lambda(p)} =_p 1$ for all elements A of $\mathbb{Z}_p^\times$. This means that the polynomial $f(X) =_p X^{\lambda(p)} - 1$ evaluates to 0 for all $\varphi(p) = p - 1$ elements of $\mathbb{Z}_p^\times$. Since a polynomial of degree $\lambda(p)$ over a field can have at most $\lambda(p)$ roots, $\lambda(p)$ cannot be smaller than $\varphi(p)$. Therefore, $\lambda(p) = \varphi(p)$, which implies that $\mathbb{Z}_p^\times$ is cyclic. Please note that this argument works only for fields (i.e. if the modulus is prime). In $\mathbb{Z}_{24}$, for example, $X^2 - 1 =_{24} 0$ has 8 solutions.

Even though we know that $\mathbb{Z}_p^\times$ is cyclic for any prime p and that it has $\varphi(p-1)$ generators, no formula is known for finding a generator without searching. We'll discuss how to find a generator for a multiplicative group in the last section of this chapter.

Since $\mathbb{Z}_{p^e}$ is not a field for any integer $e > 1$, we need four additional facts to prove that $\mathbb{Z}_{p^e}^\times$ is cyclic for any odd prime p.

▼ Binomial coefficients and the binomial theorem

Since multiplication distributes over addition, we have that $(a+b)(c+d) = a(c+d) + b(c+d) = ac + ad + bc + bd$. More generally, when evaluating the product of two sums, you add up all possible products of a term from the first sum and a term from the second sum. If the two sums are the same, you get some products several times: $(a+b)^2 = (a+b)(a+b) = aa + ab + ba + bb = a^2 + 2ab + b^2$. The same is true for powers greater than 2. In the case of $(a+b)^n$, each product consists of n instead of 2 terms. How many times do we obtain the product $a^i b^{n-i}$ for some integer i between 0 and n when we expand $(a+b)^n$? The first a can be chosen from any of the n sums, the second a from any of the remaining $n-1$ sums, and so on until you have $n-i+1$ options left for the i^{th} a. Since we don't care about the order in which we picked the as in a specific product of as and bs, we have to divide the integer that we obtained so far by the number of ways in which i as can be ordered. The first a can be any of the i as, the second a any of the remaining $i-1$ as, and so on. $i \cdot (i-1) \cdot \ldots \cdot 2 \cdot 1$ is the so-called factorial of i, which is usually written with an exclamation mark as $i!$. The number of ways in which you can pick i as out of n sums is:

$$\binom{n}{i} = \frac{n(n-1)(n-2)\ldots(n-i+1)}{i(i-1)(i-2)\ldots 1} = \frac{n!}{i!\,(n-i)!} \quad \text{for } 0 \leq i \leq n$$

This is known as the binomial coefficient, which is often written as the two numbers above each other in parentheses. Since no choice is left for the $n-i$ bs, we don't need to account for them. Instead of choosing the i as, we could choose the $n-i$ bs, though, which would give us the same result as $\binom{n}{i} = \binom{n}{n-i}$. Putting everything together, we get the binomial theorem:

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i}$$

▼ Binomial coefficients of a prime are multiples of the prime

$\binom{p}{i} =_p 0$ for any prime p and any integer i strictly between 0 and p (i.e. $0 < i < p$). Proof: Since $\binom{p}{i} = \frac{p(p-1)(p-2)\ldots(p-i+1)}{i!}$ must be an integer given what it counts, $i!$ divides $p(p-1)(p-2)\ldots(p-i+1)$. Since p is prime and all factors in $i!$ are smaller than p, $\text{gcd}(p, i!) = 1$. By Euclid's lemma, $i!$ then has to divide $(p-1)(p-2)\ldots(p-i+1)$. Therefore, $\binom{p}{i}$ is a multiple of p.

▼ Congruence after exponentiation modulo a prime power

Given a prime p and a positive integer e, $a =_{p^e} b$ implies $a^p =_{p^{e+1}} b^p$ for any integers a and b.

Proof: $a =_{p^e} b$ means that $a = b + cp^e$ for some integer c. Thus, $a^p = (b + cp^e)^p = b^p + pb^{p-1}cp^e + dp^{2e}$ for some integer d. (See the binomial theorem if you struggle with this expansion.) As a consequence, a^p equals b^p up to some multiple of p^{e+1}.

▼ Congruence of 1 plus prime power after exponentiation

Given a prime p and a positive integer e with $p^e > 2$, $a =_{p^{e+1}} 1 + p^e$ implies $a^p =_{p^{e+2}} 1 + p^{e+1}$ for any integers a and b.

Proof: According to the previous box, $a =_{p^{e+1}} 1 + p^e$ implies $a^p =_{p^{e+2}} (1 + p^e)^p$. Using the binomial theorem, we get:

$$(1+p^e)^p = \sum_{i=0}^{p} \binom{p}{i} (p^e)^i = 1 + p \cdot p^e + \Big(\sum_{i=2}^{p-1} \binom{p}{i} p^{ei} \Big) + p^{ep}$$

Since $\binom{p}{i}$ is divisible by p for $0 < i < p$, we conclude that $\binom{p}{i} p^{ei}$ is divisible by p^{1+2e} for $2 \leq i < p$. Since $1 + 2e \geq e + 2$ for all $e \geq 1$, we can ignore the terms of the second sum when computing modulo p^{e+2}. By the requirement that $p^e > 2$, either $p \geq 3$ or $e \geq 2$. In both cases, $ep \geq e + 2$. Therefore, we can ignore p^{ep} as well, which leaves us with $(1 + p^e)^p =_{p^{e+2}} 1 + p^{e+1}$.

▼ Why multiplicative groups modulo a power of an odd prime are cyclic

In this box, we want to show that $\mathbb{Z}_{p^e}^\times$ is cyclic for any odd prime p and any integer $e > 1$. We learned earlier that $\mathbb{Z}_p^\times$ is cyclic. Let G be a generator of $\mathbb{Z}_p^\times$ and n be the order of G in $\mathbb{Z}_{p^e}^\times$. Since $G^n =_{p^e} 1$, $G^n =_p 1$. (If $G^n - 1$ is a multiple of p^e, it is also a multiple of p because p^e is a multiple of p.) Therefore, n has to be a multiple of $p - 1$, which is the order of $\mathbb{Z}_p^\times$. Since n is the smallest positive integer so that $G^n =_{p^e} 1$, $G^{n/(p-1)}$ has an order of $p - 1$ in $\mathbb{Z}_{p^e}^\times$. If we find an element H with an order of p^{e-1} in $\mathbb{Z}_{p^e}^\times$, then $H \cdot G^{n/(p-1)}$ has an order of $p^{e-1}(p-1) = \varphi(p^e)$ according to an earlier theorem.

We now show that $H = 1 + p$ has an order of p^{e-1} in $\mathbb{Z}_{p^e}^\times$. According to the previous box, $H =_{p^2} 1 + p$ implies $H^p =_{p^3} 1 + p^2$. If $e > 2$, p^e is a multiple of p^3, and thus $H^p \neq_{p^e} 1$. When we raise H^p to the power of p again, we get $H^{p^2} =_{p^4} 1 + p^3$. This continues until $H^{p^{e-1}} =_{p^{e+1}} 1 + p^e =_{p^e} 1$. Since this is the first time that we reach the identity element, the order of H is p^{e-1}.

▼ Why multiplicative groups modulo a power of 2 greater than 4 are not cyclic

You can use the repetition table to verify that $\mathbb{Z}_2^\times$ and $\mathbb{Z}_4^\times$ are cyclic and that $\mathbb{Z}_8^\times$ has no generator. In this box, we first show that every element in $\mathbb{Z}_{2^e}^\times$ for an integer $e > 2$ has an order of at most 2^{e-2}. This is an upper bound for $\lambda(2^e)$. By showing that the element 5 has this order, we establish the same value as a lower bound, which implies that $\lambda(2^e) = 2^{e-2}$ for $e > 2$.

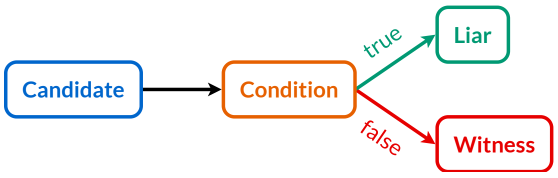
We first observe that all even integers have the factor 2 in common with powers of 2 and that all odd integers are coprime with powers of 2. Therefore, any element $A \in \mathbb{Z}_{2^e}^\times$ can be written as $A = 1 + 2b$ for some integer b. Using our algebraic skills, $A^2 = (1 + 2b)^2 = 1 + 4b + 4b^2 = 1 + 4b(1 + b)$. Since either b or $1 + b$ is even, $c = \frac{b(1+b)}{2}$ is an integer. Therefore, $A^2 = 1 + 8c$. For $e = 3$, we have seen that $A^{2^{e-2}} = 1 + 2^e c$, which means that $A^{2^{e-2}} =_{2^e} 1$. By induction, we show that this is also true for any $e > 3$: $(A^{2^{e-2}})^2 = A^{2^{e-1}} = (1 + 2^e c)^2 = 1 + 2 \cdot 2^e c + 2^{2e} c^2 = 1 + 2^{e+1}(c + 2^{e-1} c^2)$, which means that $A^{2^{e-1}} =_{2^{e+1}} 1$.

With the same argument as in the previous box, we prove that the element 5 has an order of 2^{e-2} in $\mathbb{Z}_{2^e}^\times$. This time, though, we start with $5 =_{2^3} 1 + 2^2$, which means that we have $H =_{p^3} 1 + p^2$ for $p = 2$ before raising H to the power of p for the first time. As a consequence, we lag one exponentiation behind when compared to the previous box, which is why the order of 5 is 2^{e-2} instead of 2^{e-1}. Ignoring this and working it out again, we get $5^{2^i} =_{2^{3+i}} 1 + 2^{2+i}$ when squaring 5 i times according to this box. When $i = e - 2$, we have $5^{2^{e-2}} =_{2^{e+1}} 1 + 2^e =_{2^e} 1$. When $i = e - 3$, $5^{2^{e-3}} =_{2^e} 1 + 2^{e-1} \neq_{2^e} 1$. Therefore, $|5| = 2^{e-2}$.

Probabilistic primality tests

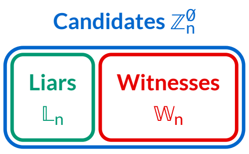
Let's continue with something different: How do we know whether an integer is prime? For small integers, we can use trial division: Given an integer n, you can simply try to divide n by all possible factors up to $\sqrt{n}$. If you find a factor, n is composite. Otherwise, n is prime. For large integers, trial division becomes infeasible because it scales with the square root of n. Even though a deterministic algorithm is known to determine in polynomial time whether an integer is prime, probabilistic primality tests are used in practice because they are much faster and much simpler.

Probabilistic primality tests are based on some condition which is true for all integers strictly between 0 and n if n is prime. If n is composite, the condition still holds for some but not all of these integers. By evaluating the condition/test repeatedly for random candidates from this set of integers, we will eventually find a candidate for which the condition is false if we keep searching for long enough. Since this cannot happen if n is prime, such a candidate is a so-called witness for the compositeness of n. A candidate for which the condition is true even though n is composite is a so-called liar as it lies about the true nature of n.



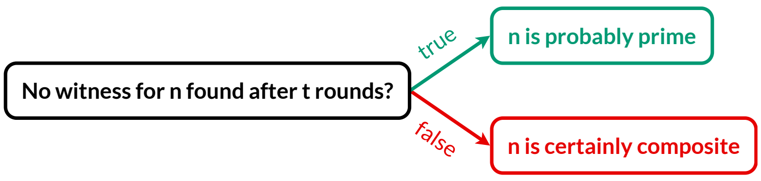
The condition classifies candidates into liars and witnesses.

If we denote the set of integers strictly between 0 and n as $\mathbb{Z}_n^\varnothing$, the set of liars for n as $\mathbb{L}_n$, and the set of witnesses for n as $\mathbb{W}_n$, we get the following Venn diagram, where $\mathbb{L}_n \cup \mathbb{W}_n = \mathbb{Z}_n^\varnothing$ and $\mathbb{L}_n \cap \mathbb{W}_n = \varnothing$:



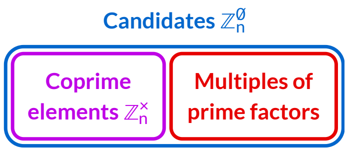
The classification visualized as sets.

If n is prime, the set of witnesses is empty and the liars aren't lying. If n is composite, the probability that a random candidate $A \in \mathbb{Z}_n^\varnothing$ is a liar is $|\mathbb{L}_n|/|\mathbb{Z}_n^\varnothing|$. When we repeat the probabilistic primality test t times, the probability that we select t liars and therefore think that n is prime even though it isn't is $(|\mathbb{L}_n|/|\mathbb{Z}_n^\varnothing|)^t$. Since $|\mathbb{L}_n|/|\mathbb{Z}_n^\varnothing| < 1$ if n is composite, we can lower the probability that we err when we declare n to be prime to an arbitrarily small number. Unless we test all possible candidates, which is infeasible when n is sufficiently large, we cannot be certain, though. For this reason, we call an integer which passes many rounds of a probabilistic primality test only a probable prime, which is in contrast to a provable prime, whose primality has been established with certainty.



The two possible outcomes of a probabilistic primality test.

In the following, we'll study two probabilistic primality tests: The Fermat primality test and the superior and thus preferable Miller-Rabin primality test. When analyzing a probabilistic primality test, we want to know whether the ratio of liars to all candidates is smaller than some bound for any composite n. Since it's difficult to reason about all the integers between 0 and n, we'll consider only the integers which are coprime with n as they form a multiplicative group, which allows us to use everything we know about groups in our analyses. Since $\mathbb{Z}_n^\times \subsetneq \mathbb{Z}_n^\varnothing$ if n is composite, it follows that $|\mathbb{L}_n|/|\mathbb{Z}_n^\varnothing| \lneq |\mathbb{L}_n|/|\mathbb{Z}_n^\times|$, which means that any upper bound which we prove for $|\mathbb{L}_n|/|\mathbb{Z}_n^\times|$ also holds for $|\mathbb{L}_n|/|\mathbb{Z}_n^\varnothing|$. As we'll see, all the candidates which aren't coprime with n are witnesses for both the Fermat and the Miller-Rabin primality test. If n is the product of large prime numbers, the multiples of these prime factors are so rare that we cannot rely on finding one of them, though.


Candidates $\mathbb{Z}_n^\varnothing$
Coprime elements $\mathbb{Z}_n^\times$
Multiples of prime factors

A different classification of the candidates, where $|\mathbb{Z}_n^\times| \leq |\mathbb{Z}_n^\varnothing| < n$ and $\mathbb{L}_n \subseteq \mathbb{Z}_n^\times$. ($|\mathbb{Z}_n^\times| = |\mathbb{Z}_n^\varnothing|$ if and only if n is prime.)

Fermat primality test

Fermat's little theorem states that given a prime n, $A^{n-1} =_n 1$ for all $A \in \mathbb{Z}_n^\times$. This can be used as the condition for a probabilistic primality test, which is known as the Fermat primality test. The problem with this test is that there are infinitely many composite integers for which the theorem holds as well. These integers are called Carmichael numbers. While Carmichael numbers are rare, they are common enough that you cannot ignore them, especially in the adversarial context of cryptography. If a composite integer n is not a Carmichael number, then at least half of all the elements in $\mathbb{Z}_n^\times$ are witnesses, which means that $|\mathbb{L}_n^F| \leq \frac{1}{2}|\mathbb{Z}_n^\times| < \frac{n}{2}$. (The F in $\mathbb{L}_n^F$ indicates that we are talking about the liars of the Fermat primality test.) There are two ways to see why this is the case:

- **Pairing**: There is at least one witness $W \in \mathbb{W}_n^F$ as n would be a Carmichael number otherwise. Since $\mathbb{Z}_n^\times$ is a group, W maps each liar $L \in \mathbb{L}_n^F$ to a distinct witness: $(L \cdot W)^{n-1} =_n L^{n-1} \cdot W^{n-1} =_n W^{n-1} \neq_n 1$ ($L^{n-1} =_n 1$ by definition). Thus, $|\mathbb{L}_n^F| \leq |\mathbb{W}_n^F|$.
- **Subgroup**: $\mathbb{L}_n^F$ is a subgroup of $\mathbb{Z}_n^\times$ because $\mathbb{L}_n^F$ is not empty as $1 \in \mathbb{L}_n^F$ and $\mathbb{L}_n^F$ is closed as $(L_1 \cdot L_2)^{n-1} =_n L_1^{n-1} \cdot L_2^{n-1} =_n 1$ for all $L_1, L_2 \in \mathbb{L}_n^F$. By Lagrange's theorem, $|\mathbb{L}_n^F| = \frac{1}{c}|\mathbb{Z}_n^\times|$ for a $c \in \mathbb{Z}_{>0}$. As n isn't a Carmichael number, $|\mathbb{L}_n^F| < |\mathbb{Z}_n^\times|$ and $c > 1$.

The following tool implements the Fermat primality test. You can enter a comma-separated list of candidates that you want to test the input n with and specify the number of rounds which shall be performed with randomly chosen candidates. See the boxes below for inputs on which the test fails with a high probability, which is why you should not use the Fermat primality test alone in practice.

Input n:	12345678910987654321	Rounds:	0
Candidates:	2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37	Abort:	
Seed:	0 Increment		

12'345'678'910'987'654'321 is probably prime.

Candidate A:	2	3	5	7	11	13	17	19	23	29	31	37
A^{n-1} % n:	1	1	1	1	1	1	1	1	1	1	1	1

▼ Remarks on this tool

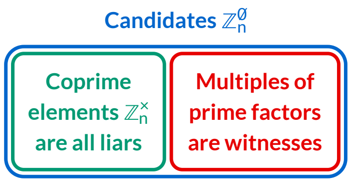
- **Input n**: The input n has to be odd because the inputs are shared with the Miller-Rabin primality test, which cannot handle even inputs. (For even inputs, the candidate 2 is always a witness as we'll see in the next box.)
- **Seed**: As many candidates as indicated by the "Rounds" slider are generated pseudo-randomly from the given seed. This allows you to revisit earlier candidates and share interesting outputs with others. Another benefit is that you get a new set of candidates simply by changing the seed, such as by clicking on the "Increment" button. If you implement a probabilistic primality test yourself, there's no reason to use a pseudo-random number generator (PRNG) like I did. In particular, using a known seed is dangerous in an adversarial setting. (The

design choice of generating candidates deterministically also made it easy to use the same candidates in the tool of the Miller-Rabin primality test below.)

- **Abort**: If you want to find liars for a particular input, you don't want the tool to stop after encountering a witness. This is why the tool allows you to disable the "Abort", which you wouldn't do if you care only about whether the input is probably prime.

▼ Carmichael numbers

A composite integer n is a Carmichael number if and only if Carmichael's totient function $\lambda(n)$ divides $n - 1$. If this is the case, then $A^{n-1} =_n A^{\lambda(n) \cdot c} =_n 1^c =_n 1$ for all A which are coprime with n, and some integer c. These numbers are also named after Robert Daniel Carmichael (1879 – 1967). The Fermat primality test fails to detect Carmichael numbers as composite only if all the chosen candidates are coprime with the number which is being tested. If a candidate A is not coprime with the integer n, A has no multiplicative inverse, and the closest you can get to a multiple of n is $\text{gcd}(A, n)$. In other words, if $\text{gcd}(A, n) > 1$, then $A^{n-1} \neq_n 1$. Thus, $\mathbb{L}_n^F = \mathbb{Z}_n^\times$. If you use the first dozen prime numbers as candidates, which the tool above does by default, the Fermat primality test fails for only 2 out of the first 33 Carmichael numbers as given by this list from the On-Line Encyclopedia of Integer Sequences (OEIS): 252'601 = 41 · 61 · 101 and 410'041 = 41 · 73 · 137. All the other numbers on the list have a factor smaller than 41. As we will see in the next box, there's a formula to obtain Carmichael numbers which don't have a small factor.


Candidates $\mathbb{Z}_n^\varnothing$
Coprime elements $\mathbb{Z}_n^\times$ are all liars
Multiples of prime factors are witnesses

How the Fermat primality test classifies the candidates of a Carmichael number.

▼ Chernick's Carmichael numbers

If you find an integer k so that $6k + 1$, $12k + 1$, and $18k + 1$ are prime, then $n = (6k + 1)(12k + 1)(18k + 1)$ is a Carmichael number since $\lambda(n) = \text{lcm}(6k, 12k, 18k) = 36k$ divides $n - 1 = 1296k^3 + 396k^2 + 36k$. ($1296/36 = 36$ and $396/36 = 11$.) The small subset of Carmichael numbers which are of this form are called Chernick's Carmichael numbers, named after Jack Chernick (1911 – 1971), who lacks an entry on Wikipedia. As we learned in the previous box, the Fermat primality test detects the compositeness of a Carmichael number only if one of the tested candidates is a multiple of one of its prime factors. The probability that a random candidate is a liar, and we thus fail to detect the compositeness of the Carmichael number n in any particular round, is $\frac{|\mathbb{L}_n^F|}{|\mathbb{Z}_n^\varnothing|} = \frac{\varphi(n)}{n-1}$. Since each round is independent from the others, the failure rate after 100 rounds is $\left(\frac{\varphi(n)}{n-1}\right)^{100}$.

k	Carmichael number	Prime factorization	Failure rate per round	Failure rate after 100 rounds
1	1'729	7 · 13 · 19	0.7500	0.0000
6	294'409	37 · 73 · 109	0.9508	0.0065
35	56'052'361	211 · 421 · 631	0.9913	0.4183
45	118'901'521	271 · 541 · 811	0.9932	0.5076
51	172'947'529	307 · 613 · 919	0.9940	0.5497
55	216'821'881	331 · 661 · 991	0.9945	0.5741
56	228'842'209	337 · 673 · 1'009	0.9946	0.5798
100	1'299'963'601	601 · 1'201 · 1'801	0.9970	0.7369
…	…	…	…	…
511	173'032'371'289	3'067 · 6'133 · 9'199	0.9994	0.9420
710	464'052'305'161	4'261 · 8'521 · 12'781	0.9996	0.9579

The first Chernick's Carmichael numbers according to this list from the On-Line Encyclopedia of Integer Sequences (OEIS).

▼ Carmichael numbers are a product of at least three distinct primes

It is no coincidence that Chernick's formula for Carmichael numbers is a product of three prime numbers. In this box we'll show that every Carmichael number is a product of at least three distinct primes. Let $n = p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}$ be the prime factorization of an arbitrary Carmichael number. This factorization has the following four properties (see page 308 in Victor Shoup's book):

1. n is odd, i.e. $p_i \neq 2$ for all $i \in \{1, \ldots, l\}$: If n was even, $n - 1$ would be odd and the coprime element $-1 =_n n - 1$ would be a witness because $-1^d =_n -1 \neq_n 1$ for any odd integer d. Since n is a Carmichael number, this cannot be the case.
2. $e_i = 1$ for every $i \in \{1, \ldots, l\}$: The composite group $\mathbb{Z}_n^\times$ is isomorphic to the direct product of $\mathbb{Z}_{p_1^{e_1}}^\times, \ldots, \mathbb{Z}_{p_l^{e_l}}^\times$. Since n is a Carmichael number, $A^{n-1} =_n 1$ for every $A \in \mathbb{Z}_n^\times$. Since the isomorphism maps $1 \in \mathbb{Z}_n^\times$ to $(1, \ldots, 1) \in \mathbb{Z}_{p_1^{e_1}}^\times \times \ldots \times \mathbb{Z}_{p_l^{e_l}}^\times$, it has to be the case that $A^{n-1} =_{p_i^{e_i}} 1$ for every $A \in \mathbb{Z}_{p_i^{e_i}}^\times$. Since p_i is odd according to the previous point, $\mathbb{Z}_{p_i^{e_i}}^\times$ is cyclic, and thus $n - 1$ has to be a multiple of $|\mathbb{Z}_{p_i^{e_i}}^\times| = \varphi(p_i^{e_i}) = p_i^{e_i - 1}(p_i - 1)$. With $e_i > 1$, $p_i^{e_i-1}(p_i - 1)$ would be a multiple of p_i, and thus $n - 1$ would also be a multiple of p_i. However, given that p_i is a factor of n, e_i has to be 1 because n and $n - 1$ cannot both be multiples of p_i.
3. $p_i - 1$ divides $n - 1$ for all $i \in \{1, \ldots, l\}$: I explained in the previous point why $n - 1$ has to be a multiple of $p_i^{e_i-1}(p_i - 1)$. Now that we know that $e_i = 1$, it follows that $n - 1$ is a multiple of $p_i - 1$.
4. $l \geq 3$: By definition, n has to be composite and thus $l > 1$. Suppose $l = 2$, which means that n is the product of two primes. According to the previous point, $p_1 - 1$ divides $n - 1$. Given that $n - 1 = p_1 p_2 - 1 = (p_1 - 1)p_2 + (p_2 - 1)$, $p_1 - 1$ has to divide $p_2 - 1$. ($a \cdot b + c$ can be a multiple of a only if c is a multiple of a.) By a symmetric argument, $p_2 - 1$ has to divide $p_1 - 1$. However, two integers can be multiples of each other only if they are the same. Since our notation for the prime factorization requires the factors to be distinct, p_1 cannot equal p_2. Hence, n is the product of at least three distinct primes.

The second and the third point are sufficient for n to be a Carmichael number because they ensure that $\lambda(n)$ divides $n - 1$ as required by our definition. This is known as Korselt's criterion, named after Alwin Reinhold Korselt (1864 – 1947).

▼ Density of Carmichael numbers and prime numbers

For a sufficiently large integer x, there are at least $x^{1/3}$ Carmichael numbers smaller than x. On the other hand, the number of primes smaller than x is approximately $\frac{x}{\log_e(x)}$, where $\log_e(x)$ is the natural logarithm of x. To give you an idea for how much rarer Carmichael numbers are than prime numbers: There are 2'220'819'602'560'918'840 ≈ 2 · 10^{18} prime numbers and only 8'220'777 ≈ 8 · 10^6 Carmichael numbers below 100'000'000'000'000'000'000 = 10^{20}.

▼ Integers for which half of all coprime elements are Fermat liars

As we saw above, the Fermat liars $\mathbb{L}_n^F$ form a subgroup of the coprime elements $\mathbb{Z}_n^\times$ for all integers n greater than 1. Due to Lagrange's theorem, $|\mathbb{Z}_n^\times|/|\mathbb{L}_n^F|$ is an integer. This ratio is known as the index of the subgroup. If the index of $\mathbb{L}_n^F$ is 1 and n is composite, n is a Carmichael number. If a composite n is not a Carmichael number, the index of $\mathbb{L}_n^F$ has to be at least 2. There are integers for which half of all coprime elements are liars in the Fermat primality test:

			Ratio of liars
n	**Prime factorization**	**Number of liars**	**to all candidates**
15	3 · 5	4	0.2857
91	7 · 13	36	0.4000
703	19 · 37	324	0.4615
1'891	31 · 61	900	0.4762
2'701	37 · 73	1'296	0.4800
11'305	5 · 7 · 17 · 19	3'456	0.3057
12'403	79 · 157	6'084	0.4906
13'981	11 · 31 · 41	6'000	0.4292
…	…	…	…

The odd integers for which $\frac{|\mathbb{L}_n^F|}{|\mathbb{Z}_n^\times|} = \frac{1}{2}$ according to this list from the On-Line Encyclopedia of Integer Sequences (OEIS).

If we want the ratio of Fermat liars to all candidates (i.e. $|\mathbb{L}_n^F|/|\mathbb{Z}_n^\varnothing|$) to be as close to 0.5 as possible, we're interested in those integers which are the product of just two primes. Such integers are known as semiprimes. (The more primes there are in the prime factorization of n, the smaller $\frac{\varphi(n)/2}{n-1}$ becomes.) It turns out that the semiprimes in the above list are of the form $n = p \cdot q$, where p and q are prime and $q = 2p - 1$. The beginning of the list is the same, and we approach 0.5 for larger values of n:

n	Prime factorization	Number of liars	Ratio of liars to all candidates
…	…	…	…
2'701	37 · 73	1'296	0.4800
12'403	79 · 157	6'084	0.4906
18'721	97 · 193	9'216	0.4923
38'503	139 · 277	19'044	0.4946
49'141	157 · 313	24'336	0.4952
79'003	199 · 397	39'204	0.4962
…	…	…	…
1'373'653	829 · 1'657	685'584	0.4991
1'537'381	877 · 1'753	767'376	0.4991

The semiprimes $n = p \cdot q$ so that $q = 2p - 1$ according to this list from the On-Line Encyclopedia of Integer Sequences (OEIS).

Therefore, 0.5 is indeed the best bound for the ratio of Fermat liars to all candidates of composite non-Carmichael numbers.

Miller-Rabin primality test

We can improve upon the Fermat primality test by making the condition in our test stricter: When computing $A^{n-1} =_n 1$ with the square-and-multiply algorithm, we can require that whenever you reach 1 by squaring some number, this number must be either 1 or -1. Given our knowledge, we have three ways to see why 1 and -1 are the only possible square roots of 1 in $\mathbb{Z}_n^\times$ if n is prime:

- **Euclid's lemma**: We want to find the possible solutions for x in $x^2 =_n 1$. By taking the 1 to the left side, we get $x^2 - 1 =_n 0$, which we can refactor into $(x-1)(x+1) =_n 0$. Since the product of $x - 1$ and $x + 1$ is a multiple of the prime n, $x - 1$ or $x + 1$ has to be a multiple of n according to Euclid's lemma. If $x - 1$ is a multiple of n, then $x =_n 1$. If not, then x has to equal -1.
- **Cyclic group**: Since n is prime, $\mathbb{Z}_n^\times$ is cyclic. As I argued earlier, there can be only a single element half-way through the cycle.
- **Polynomial**: Even though this is not officially part of our toolkit yet, I mentioned earlier that $\mathbb{Z}_n$ is a finite field if n is prime and that a polynomial over a field can have no more roots than its degree. Therefore, the polynomial $f(x) = x^2 - 1$ can evaluate to 0 for at most two distinct x, which are 1 and -1.

After checking that n is odd, we know that $n - 1$ is even. (If you want to be able to handle the case where $n = 2$, you have to handle it explicitly with a separate conditional.) Since $A^{n-1} =_n 1$ for all $A \in \mathbb{Z}_n^\times$ by Fermat's little theorem, $A^{\frac{n-1}{2}}$ has to equal either 1 or -1 up to a multiple of n if n is prime. If this is not the case, we know for sure that n is composite. If $A^{\frac{n-1}{2}} =_n 1$ and $\frac{n-1}{2}$ is still even, we can continue the process with $A^{\frac{n-1}{4}}$ and so on. Instead of calculating such large exponentiations repeatedly, we write $n - 1$ as $2^c d$ with an integer $c \geq 1$ and an odd integer d. For a candidate A, we calculate $A^d \text{ mod } n$. If the result is 1, we can continue with the next candidate because the result remains 1 when we square it repeatedly until we arrive at $(A^d)^{2^c} =_n A^{n-1} =_n 1$. If the result is -1, we can also continue with the next candidate because we get 1 after squaring the result once. For any other result, we square the result and check what we have then. If the new result is 1, we know that n is composite because the previous result was neither 1 nor -1. If the new result is -1, we can continue with the next candidate because we're certain that A^{n-1} will be 1. If necessary, we square the first result (i.e. A^d) $c - 1$ times. If we still haven't got -1 at this point, we know that n is composite because even if we would get 1 at A^{n-1} when squaring $A^{\frac{n-1}{2}}$ one more time, we would arrive from a number other than -1 or 1.

This algorithm is known as the Miller-Rabin primality test, named after Gary Lee Miller (I couldn't figure out when this gentleman was born) and Michael Oser Rabin (born in 1931). As for any probabilistic primality test, liars still exist. (If there were no liars, the test wouldn't be probabilistic and we wouldn't have to check more than one candidate.) Unlike the Fermat primality test, however, $|\mathbb{L}_n^{MR}|/|\mathbb{Z}_n^\varnothing| \leq \frac{1}{4}$ for any odd composite n as we will prove below. Any Miller-Rabin liar is also a Fermat liar (i.e. $\mathbb{L}_n^{MR} \subseteq \mathbb{L}_n^F$), so for any candidates for which the Miller-Rabin primality test fails, the Fermat primality test fails as well. Given that the Miller-Rabin primality test doesn't struggle with Carmichael numbers, that it has a smaller bound for the ratio of liars to all candidates than the Fermat primality test ($\frac{1}{4}$ instead of $\frac{1}{2}$, which the latter achieves only for non-Carmichael numbers), and that it computes fewer squares per round than the Fermat primality test, the Miller-Rabin primality test is strictly preferable to the Fermat primality test.

The following tool implements the Miller-Rabin primality test. Its input is synchronized with the Fermat primality test above so that you can compare the outputs of the two tests. (This is best done on this separate page, which includes only the tools of this article.) Since 1 and $n-1$ are always liars, this tool samples the random candidates between these two values. For this reason, the smallest integer that you can test is 5. You can use the up and down arrows on your keyboard to step through the odd inputs when the cursor is in the field of the input n. The tool displays -1 only for your convenience. When you calculate modulo n, you get $n-1$, of course.

Input n:	12345678910987654321	Rounds:	0
Candidates:	2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37	Abort:	
Seed:	0 Increment		

12'345'678'910'987'654'321 is probably prime.

n = $2^c \cdot$ d + 1
12'345'678'910'987'654'321 = $2^4 \cdot$ 771'604'931'936'728'395 + 1

Candidate A	$A^{2^0 \cdot d}$ % n	$A^{2^1 \cdot d}$ % n	$A^{2^2 \cdot d}$ % n	$A^{2^3 \cdot d}$ % n
2	1			
3	3'919'199'263'467'263'698	9'967'901'397'488'111'931	−1	
5	1			
7	3'707'150'871'678'934'042	10'183'580'126'972'188'736	2'377'777'513'499'542'390	−1
11	9'967'901'397'488'111'931	−1		
13	477'990'353'888'317'855	2'162'098'784'015'465'585	2'377'777'513'499'542'390	−1
17	10'192'383'726'506'864'521	8'426'479'647'520'390'623	9'967'901'397'488'111'931	−1
19	8'638'528'039'308'720'279	10'183'580'126'972'188'736	2'377'777'513'499'542'390	−1
23	11'867'688'557'099'336'466	2'162'098'784'015'465'585	2'377'777'513'499'542'390	−1
29	2'153'295'184'480'789'800	8'426'479'647'520'390'623	9'967'901'397'488'111'931	−1
31	3'707'150'871'678'934'042	10'183'580'126'972'188'736	2'377'777'513'499'542'390	−1
37	3'707'150'871'678'934'042	10'183'580'126'972'188'736	2'377'777'513'499'542'390	−1

▼ **Small prime numbers as candidates**

It turns out that the first l prime numbers form excellent candidates for the Miller-Rabin primality test:

l	Candidates	Smallest composite integer for which the test fails
1	2	2'047 > 2^{10}
2	2, 3	1'373'653 > 2^{20}
3	2, 3, 5	25'326'001 > 2^{24}
4	2, 3, 5, 7	3'215'031'751 > 2^{31}
5	2, 3, 5, 7, 11	2'152'302'898'747 > 2^{40}
6	2, 3, 5, 7, 11, 13	3'474'749'660'383 > 2^{41}
7	2, 3, 5, 7, 11, 13, 17	341'550'071'728'321 > 2^{48}
8	2, 3, 5, 7, 11, 13, 17, 19	341'550'071'728'321 > 2^{48}
9	2, 3, 5, 7, 11, 13, 17, 19, 23	3'825'123'056'546'413'051 > 2^{61}
10	2, 3, 5, 7, 11, 13, 17, 19, 23, 29	3'825'123'056'546'413'051 > 2^{61}
11	2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31	3'825'123'056'546'413'051 > 2^{61}
12	2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37	318'665'857'834'031'151'167'461 > 2^{78}
13	2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41	3'317'044'064'679'887'385'961'981 > 2^{81}

The smallest integers for which the first l prime numbers are not enough to detect their compositeness according to this list from the On-Line Encyclopedia of Integer Sequences (OEIS).

▼ Power function and its preimage

The function $f_y(X) =_n X^y$ computes the yth power of any element X of the multiplicative group $\mathbb{Z}_n^\times$. For any subset $\mathbb{S} \subseteq \mathbb{Z}_n^\times$, we denote the so-called preimage of $\mathbb{S}$ under f_y as $f_y^{-1}(\mathbb{S}) = \{X \in \mathbb{Z}_n^\times \mid f_y(X) \in \mathbb{S}\}$. (As we will revisit later, $f_y^{-1}(\{1\})$ is the so-called kernel of f_y.) We need the following property in order to prove the Monier-Rabin bound on the number of liars: If $\mathbb{Z}_n^\times$ is cyclic, $|f_y^{-1}(\{1\})| = \gcd(y, \varphi(n))$. Proof: Let G be a generator of the cyclic group, then for any $X \in \mathbb{Z}_n^\times$, there's an integer x so that $X =_n G^x$. We want to determine the number of $X \in \mathbb{Z}_n^\times$ for which $X^y =_n (G^x)^y =_n 1$. Now G^z can equal 1 only if z is a multiple of $|G| = \varphi(n)$. How many $x \in \mathbb{Z}_{\varphi(n)}$ are there so that $x \cdot y =_{\varphi(n)} 0$? As we saw earlier, the answer is $\gcd(y, \varphi(n))$.

▼ Monier-Rabin bound on the number of liars

Louis Monier (born in 1956) and Michael Oser Rabin (born in 1931) proved independently in 1980 that $|\mathbb{L}_n^{MR}|/|\mathbb{Z}_n^\emptyset| \leq \frac{1}{4}$ for any odd composite integer n. As a consequence, the probability that an odd composite integer n is mislabeled as a probable prime after t rounds of the Miller-Rabin primality test with random candidates is at most $(\frac{1}{4})^t$. I adapted the following proof from page 309 of Victor Shoup's book. You find similar approaches here and here in case you cannot follow my explanations.

After checking that n is odd and aborting the Miller-Rabin primality test otherwise, there are two cases to consider:

1. $n = p^e$ for some odd prime p and an integer $e > 1$: Since every Miller-Rabin liar is also a Fermat liar (i.e. $\mathbb{L}_n^{MR} \subseteq \mathbb{L}_n^F$), any upper bound on the latter also applies to the former (i.e. if $|\mathbb{L}_n^F| \leq \frac{n-1}{4}$, then $|\mathbb{L}_n^{MR}| \leq \frac{n-1}{4}$). An element $L \in \mathbb{Z}_n^\times$ is a Fermat liar if and only if $L^{n-1} =_n 1$. Since multiplicative groups modulo a power of an odd prime are cyclic, there are $|f_{n-1}^{-1}(\{1\})| = \gcd(n-1, \varphi(n))$ Fermat liars in $\mathbb{Z}_n^\times$ according to the previous box. Using the definitions of Euler's totient function φ and n, we get $\gcd(p^e - 1, p^{e-1}(p-1))$. Clearly, $p - 1$ divides $p^{e-1}(p-1)$. Since $(p-1)\sum_{i=0}^{e-1} p^i = \big(\sum_{i=1}^{e} p^i\big) - \big(\sum_{i=0}^{e-1} p^i\big) = p^e - 1$, $p-1$ divides also $p^e - 1$. (It's obvious when putting it as $(m+1)^e =_m 1$ for $m = p - 1$.) Since $\sum_{i=0}^{e-1} p^i$ is not a multiple of p (as it is one more than a multiple of p), $p - 1$ is the greatest common divisor of $p^e - 1$ and $p^{e-1}(p-1)$. Finally, $p - 1 = (p^e - 1)/\big(\sum_{i=0}^{e-1} p^i\big) = \frac{n-1}{p^{e-1}+\ldots+1} \leq \frac{n-1}{4}$ because $p^{e-1} + 1 \geq 4$ for any $p \geq 3$ and $e \geq 2$. Thus, $|\mathbb{L}_n^F| \leq |\mathbb{Z}_n^\emptyset|/4$. This bound is reached for $n = 9$ (i.e. $p = 3$ and $e = 2$), where two out of eight elements (namely 1 and 8) are Miller-Rabin liars. This is the reason why I write $|\mathbb{L}_n^{MR}|/|\mathbb{Z}_n^\emptyset| \leq \frac{1}{4}$ instead of $|\mathbb{L}_n^{MR}|/|\mathbb{Z}_n^\times| \leq \frac{1}{4}$ as $\varphi(9) = 6$ and $|\mathbb{L}_9^{MR}|/|\mathbb{Z}_9^\times| = \frac{2}{6} > \frac{1}{4}$.
2. $n = p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}$ for $l > 1$ distinct odd primes p_i and positive integers e_i: This case is more complicated because $\mathbb{L}_n^{MR}$ is generally not closed under multiplication. (For example, 8 and 18 are liars for $n = 65$, but $8 \cdot 18 =_{65} 14$ is a witness for the compositeness of n.) The composite group $\mathbb{Z}_n^\times$ is isomorphic to the direct product of $\mathbb{Z}_{p_1^{e_1}}^\times, \ldots, \mathbb{Z}_{p_l^{e_l}}^\times$. Let $n - 1 = 2^c d$ for an integer $c \geq 1$ and an odd integer d. Similarly, $\varphi(p_i^{e_i}) = 2^{c_i} d_i$, where $1 \leq i \leq l$ and d_i is odd. We define a as the smallest integer in the set $\{c, c_1, \ldots, c_l\}$. Since $\varphi(p_i^{e_i})$ is even for any odd prime p_i, $a \geq 1$. Next, we'll prove a series of statements:
   - $L^{2^a d} =_n 1$ for all $L \in \mathbb{L}_n^{MR}$. If $a = c$, this is the case because every Miller-Rabin liar is also a Fermat liar (i.e. $L^{n-1} =_n 1$). If $a < c$, $a = c_j$ for some index j. If $L^{2^b d} \neq_n 1$ for some $L \in \mathbb{L}_n^{MR}$, we must have $L^{2^b d} =_n -1$ for some b greater than or equal to a and smaller than c because L wouldn't be a Miller-Rabin liar otherwise. If $L^{2^b d} + 1$ is a multiple of n, it is also a multiple of $p_j^{e_j}$. Therefore, $L^{2^b d} =_n -1$ implies that $L^{2^b d} =_{p_j^{e_j}} -1$. Since $(L^d)^{2^b} =_{p_j^{e_j}} -1$ and $(L^d)^{2^{b+1}} =_{p_j^{e_j}} 1$, the order of L^d is 2^{b+1} in $\mathbb{Z}_{p_j^{e_j}}^\times$ according to an earlier theorem. However, the order of $\mathbb{Z}_{p_j^{e_j}}^\times$ is $2^{c_j} d_j$, which means that $|L^d|$ does not divide $|\mathbb{Z}_{p_j^{e_j}}^\times|$ because $b + 1 > c_j$. Since this is a contradiction, there is no liar $L \in \mathbb{L}_n^{MR}$ for which $L^{2^a d} \neq_n 1$.
   - $|\mathbb{L}_n^{MR}| \leq 2 \cdot |f_{2^{a-1}d}^{-1}(\{1\})|$. It follows from the previous point and the definition of a Millar-Rabin liar that $L^{2^{a-1}d} =_n \pm 1$ for all $L \in \mathbb{L}_n^{MR}$. Therefore, $\mathbb{L}_n^{MR} \subseteq f_{2^{a-1}d}^{-1}(\{1\}) \cup f_{2^{a-1}d}^{-1}(\{-1\})$. Let $\{U_1, U_2, U_3, \ldots, U_v\}$ denote all the elements of $f_{2^{a-1}d}^{-1}(\{-1\})$. Then $f_{2^{a-1}d}^{-1}(\{1\})$ cannot be smaller than $f_{2^{a-1}d}^{-1}(\{-1\})$ because it contains at least the v elements $\{U_1 \cdot U_1, U_1 \cdot U_2, U_1 \cdot U_3, \ldots, U_1 \cdot U_v\}$. These elements are distinct because all preimages from U_1 to U_v belong to $\mathbb{Z}_n^\times$ and you get different results when you combine the same element with different elements in a group.
   - $|f_y^{-1}(\{1\})| = \prod_{i=1}^{l} \gcd(y, 2^{c_i} d_i)$ for any positive integer y. Since $\mathbb{Z}_n^\times \cong \mathbb{Z}_{p_1^{e_1}}^\times \times \ldots \times \mathbb{Z}_{p_l^{e_l}}^\times$, the number of elements in $\mathbb{Z}_n^\times$ which map to 1 when being raised to the yth power is given by the product of the number of elements which do the same in each of the groups $\mathbb{Z}_{p_1^{e_1}}^\times$ to $\mathbb{Z}_{p_l^{e_l}}^\times$. Since each $\mathbb{Z}_{p_i^{e_i}}^\times$ is a cyclic group, the formula from the previous box can be used.
   - $|f_{2^a d}^{-1}(\{1\})| = 2^l \cdot |f_{2^{a-1}d}^{-1}(\{1\})|$. According to the previous point, $|f_{2^a d}^{-1}(\{1\})| = \prod_{i=1}^{l} \gcd(2^a d, 2^{c_i} d_i)$. Since $a \geq 1$ and $c_i \geq a$ for $i = 1, \ldots, l$, we have that $|f_{2^a d}^{-1}(\{1\})| = 2^l \cdot \prod_{i=1}^{l} \gcd(2^{a-1} d, 2^{c_i} d_i) = 2^l \cdot |f_{2^{a-1}d}^{-1}(\{1\})|$. (This is why a has to be smaller than or equal to each c_i; you wouldn't be able to extract the factor 2 in each of the groups otherwise.)
   - $|f_{2^a d}^{-1}(\{1\})| \leq |f_{2^c d}^{-1}(\{1\})|$ because for any $X \in \mathbb{Z}_n^\times$ for which $X^{2^a d} =_n 1$, $X^{2^c d} =_n 1$ since $c \geq a$.

   If follows from these statements that $|\mathbb{L}_n^{MR}| \leq 2^{-l+1} \cdot |f_{n-1}^{-1}(\{1\})|$ since $|\mathbb{L}_n^{MR}| \leq 2 \cdot |f_{2^{a-1}d}^{-1}(\{1\})| = 2 \cdot 2^{-l} \cdot |f_{2^a d}^{-1}(\{1\})| \leq 2^{-l+1} \cdot |f_{2^c d}^{-1}(\{1\})| = 2^{-l+1} \cdot |f_{n-1}^{-1}(\{1\})|$ given that $2^c d = n - 1$. There are two cases to consider:

- $l \geq 3$: This implies that $|\mathbb{L}_n^{MR}| \leq |f_{n-1}^{-1}(\{1\})|/4$. Since $|f_{n-1}^{-1}(\{1\})| \leq |\mathbb{Z}_n^\times|$ and $|\mathbb{Z}_n^\times| < |\mathbb{Z}_n^\varnothing|$, we have $|\mathbb{L}_n^{MR}| < |\mathbb{Z}_n^\varnothing|/4$.
- $l = 2$: According to an earlier theorem, n cannot be a Carmichael number. Since in this case at most half of all coprime elements can be Fermat liars, we have that $|f_{n-1}^{-1}(\{1\})| = |\mathbb{L}_n^F| \leq |\mathbb{Z}_n^\times|/2$. Thus, $|\mathbb{L}_n^{MR}| \leq 2^{-1} \cdot |f_{n-1}^{-1}(\{1\})| < |\mathbb{Z}_n^\varnothing|/4$.

Sophie Germain and safe primes

A prime number p is a Sophie Germain prime, named after Marie-Sophie Germain (1776 – 1831), if $q = 2p + 1$ is also prime. A prime number q for which $p = (q - 1)/2$ is also prime is called a safe prime. Safe primes are important in cryptography because the difficulty of solving the discrete-logarithm problem is bounded by the largest prime factor of the group's order as we will see later.

Sophie Germain prime	2	3	5	11	23	29	41	53	83	89	…
Safe prime	5	7	11	23	47	59	83	107	167	179	…

The first ten Sophie Germain primes with their corresponding safe primes.

Search for a probable prime

How do we find large prime numbers that we can use for cryptographic applications? Since prime numbers are sufficiently dense, we can repeatedly check with the Miller-Rabin primality test whether a randomly generated integer of the desired length is a probable prime until we find one. The following tool implements this random search for a probable prime. If you need a safe prime, the tool generates a random number p until both p and $q = 2p + 1$ are probably prime. In order to let you see what the tool is doing, the tool waits for a fraction of a second after each attempt. You should disable this delay when generating large prime numbers.

Bits: 32 Safe prime: Generator: Delay: 0.20 Generate

Attempts:	5
Elapsed time:	1.85 s

p = 2'891'966'059

Clear

▼ Primality test optimizations

Since this algorithm tests many integers for their primality, the performance of the prime number generation is determined by the performance of the employed primality test. Here are some ways to reject composite inputs as quickly as possible:

- **Trial division by small primes**: Since every second number is a multiple of 2, every third number number is a multiple of 3, and so on, you can rule out a substantial fraction of inputs using trial division by small prime numbers. (To check that an input is odd, you have to verify only that the least-significant bit is 1.)
- **Coprimality with product of small primes**: You can check that an input is not a multiple of many small primes "at once" by checking that the input is coprime with their product using the Euclidean algorithm. The product of the first l prime numbers is known as the primorial number $p_l\# = \prod_{i=1}^{l} p_i$, where p_i is the i^{th} prime number. For example, the product of the first 40 prime numbers is 166'589'903'787'325'219'380'851'695'350'896'256'250'980'509'594'874'862'046'961'683'989'710 = 2 · 3 · 5 · 7 · 11 · 13 · 17 · 19 · 23 · 29 · 31 · 37 · 41 · 43 · 47 · 53 · 59 · 61 · 67 · 71 · 73 · 79 · 83 · 89 · 97 · 101 · 103 · 107 · 109 · 113 · 127 · 131 · 137 · 139 · 149 · 151 · 157 · 163 · 167 · 173 (see the last row of this table).
- **Incremental instead of random search**: Instead of choosing the next input randomly, you can increment the previous input by a known value. This has the advantage that you no longer need to perform the trial divisions if you store the remainders from the previous attempt. See the note 4.51 in the Handbook of Applied Cryptography for a more detailed discussion of this approach. (The tool above increments even inputs by 1 to make them odd before performing any primality checks.)

▼ Expected number of attempts

As I mentioned earlier, there are approximately $\frac{x}{\log_e(x)}$ prime numbers smaller than x. Since half of all the integers are odd, the probability that a random odd integer smaller than x is prime is around $\frac{x}{\log_e(x)} / \frac{x}{2} = \frac{2}{\log_e(x)}$. Denoting the number of bits we are interested in as l, we get $\frac{2}{\log_e(2^l)}$. Since $s = \log_e(2^l)$ means that $e^s = 2^l$, we can raise both sides by $1/l$ to get $e^{s/l} = 2$. Thus, $\log_e(2) = s/l$ and $s = l \cdot \log_e(2) = \log_e(2^l)$, which is one of the logarithmic identities. The number of independent trials needed to get a success is given by the geometric distribution. If the probability of success in each attempt is P, it takes on average $\frac{1}{P}$ attempts. As a consequence, the expected number of attempts needed to find a prime number with l bits is $l \cdot \frac{\log_e(2)}{2} = 0.34657 \cdot l \approx \frac{1}{3}l$. The probability of finding a safe prime is P^2, which means that it takes $(l \cdot \frac{\log_e(2)}{2})^2 = 0.12 \cdot l^2$ attempts on average. The following table lists the expected number of attempts for lengths in the range of the tool above.

Number of bits	Expected number of attempts	Squared for safe prime
8	3	8
16	6	31
32	11	123
64	22	492
128	44	1'968
256	89	7'872
512	177	31'487
1'024	355	125'948
2'048	710	503'791

The expected number of attempts to find a (safe) prime of the given length rounded to the next integer.

While I implemented some of the optimizations mentioned in the previous box, the tool is not fast enough to find a safe prime with 2'048 bits in a reasonable amount of time. I still wanted to support this length so that you get a sense for how large the prime number has to be in order to get a reasonable amount of security. (Finding a normal prime of this length is no problem.)

▼ **Required number of Miller-Rabin rounds**

Since the Miller-Rabin primality test is a probabilistic primality test, what is the probability that the random search for a prime number returns a composite number? How many rounds do we have to perform in the Miller-Rabin primality test to push this probability below a certain value, such as $(\frac{1}{2})^{80}$? You may think that this error probability is given by the Monier-Rabin bound on the number of liars, namely $(\frac{1}{4})^t$, where t denotes the number of rounds. However, this bound limits the probability that the Miller-Rabin primality test declares an input to be prime given that the input is composite. Using mathematical notation, this is $P(D_t \mid C) \leq (\frac{1}{4})^t$, where D_t denotes the event that the input is declared prime after t rounds of the Miller-Rabin primality test and C represents the event that the input is composite. ($P(D_t \mid C)$ is a conditional probability, which is defined as $P(D_t \mid C) = \frac{P(D_t \cap C)}{P(C)}$.) What we are interested in here, though, is the probability that an input is composite given that the Miller-Rabin primality test declared it to be prime after t rounds: $P(C \mid D_t)$. Using Bayes' theorem, we have

$$P(C \mid D_t) = P(C) \cdot \frac{P(D_t \mid C)}{P(D_t)} \leq \frac{P(D_t \mid C)}{P(D_t)} \leq \frac{1}{r}(\frac{1}{4})^t,$$

where r is the ratio of the inputs which are prime. Since the Miller-Rabin primality test declares all inputs which are prime to be prime, $P(D_t) \geq r$. If r is small, the probability $P(C \mid D_t)$ can be considerably larger than $(\frac{1}{4})^t$. (For example, the probability of error is 1 if the set of candidates contains only composite numbers.) However, the fraction of Miller-Rabin liars is far smaller than $\frac{1}{4}$ for most numbers. According to the note 4.49 in the Handbook of Applied Cryptography, it has been shown that just 6 rounds of the Miller-Rabin primality test are needed to push the probability that the random search for a 512-bit prime returns a composite number below $(\frac{1}{2})^{80}$. For 1'024 bits, 3 rounds are enough, and for 2'048 bits, it suffices to perform 2 rounds. Since you perform on average only a bit more than one round of the Miller-Rabin primality test for unsuccessful candidates even in the worst case, choosing a bigger t won't affect the performance of the random search by much. The situation is different when searching for safe primes, but a better optimization is to increase your confidence that both p and $q = 2p + 1$ are prime in parallel instead of gaining high certainty that p is prime before even considering q. (I didn't implement this optimization in the above tool because it wouldn't match the visualization, including the counter of how many times p was prime.)

Please note that this discussion applies only to the case where the candidates are sampled randomly. If you test an input which is provided by an untrusted party, you can rely only on the worst-case error bound of $(\frac{1}{4})^t$. Not considering the adversarial setting has been an issue in many cryptography libraries. If you verify that an integer provided by someone else is prime, you should use 64 rounds to push the risk of being exploited below 2^{-128}. Alternatively, you can use the Baillie-PSW primality test, which is named after Robert Baillie, Carl Bernard

Pomerance (born in 1944), John Lewis Selfridge (1927 – 2010), and Samuel Standfield Wagstaff Jr. (born in 1945). This primality test consists of a single round of the Miller-Rabin primality test with the candidate 2 followed by the Lucas probable prime test, which is named after François Édouard Anatole Lucas (1842 – 1891).

▼ **Performance impact of the number of rounds**

Since at most $\frac{1}{4}$ of all candidates of a composite number are liars, the probability that you have to perform a second round of the Miller-Rabin primality test after performing a first round with a random candidate is at most $\frac{1}{4}$ for any composite number. The probability that you need a third round is at most $\frac{1}{16}$, and so on. In the worst case (the fraction of liars is usually much smaller than $\frac{1}{4}$), the expected number of rounds needed to detect that a composite number is composite is $1 + \frac{1}{4} + \frac{1}{16} + \ldots = \frac{4}{3}$. ($s = \sum_{i=0}^{\infty} r^i = 1 + r + r^2 + \ldots$ is a geometric series, which converges to $\frac{1}{1-r}$ if $|r| < 1$ because $s - rs = s(1 - r) = 1$ as all but the first term cancel out.) We get the same result when we interpret this as the geometric distribution, whose expected value is $\frac{1}{P}$, with a success probability of $P = \frac{3}{4}$. Ignoring the optimizations mentioned above, we have to multiply the expected number of attempts to find a probable prime by $\frac{4}{3}$ to get an estimate for the total number of Miller-Rabin primality test rounds before the last attempt, which returns the probable prime after an additional t rounds. This estimate is not really affected by the parameter t. For example, when we search for a 2'048-bit prime, we expect to perform in the order of $710 \cdot \frac{4}{3} = 947$ rounds of the Miller-Rabin primality test before the last attempt. Whether you perform an additional 2 or 64 rounds for the last attempt is only a matter of a few percentages. The above optimizations make the impact of the last attempt on the overall performance significantly bigger, but we're still talking about percentages and not multiples when we change the parameter t.

Search for a group generator

Given a cyclic group $\mathbb{G}$ and the prime factorization of its order $n = p_1^{e_1} p_2^{e_2} \ldots p_l^{e_l}$, $G \in \mathbb{G}$ generates the whole group if and only if

Multiplicative | Additive | Both

Multiplicative

$$G^{\frac{n}{p_i}} \neq I$$

Additive

$$\frac{n}{p_i} G \neq O$$

for i from 1 to l. (According to Lagrange's theorem, the order of G has to divide the group order n, and the above condition ensures that the order of G is not one factor smaller than n.) A simple way to find a generator is to repeatedly choose a random element from $\mathbb{G}$ until this is the case. Since a cyclic group of order n has $\varphi(n)$ generators, the probability that a randomly chosen element generates the group is $\frac{\varphi(n)}{n}$. The number of trials needed to find a generator follows the geometric distribution. This means that we find a generator after $\frac{n}{\varphi(n)}$ trials on average. Since $\varphi(n) > \frac{n}{6 \log_e(\log_e(n))}$ for all $n \geq 5$ (see the fact 2.102 in the Handbook of Applied Cryptography), $\frac{n}{\varphi(n)} < 6 \log_e(\log_e(n))$. We will study a smarter version of this algorithm in a box below.

This algorithm can be used only if the prime factorization of the group's order is known. As mentioned earlier, integer factorization is a hard problem. We can avoid this problem for multiplicative groups modulo a prime number q by choosing the factors of $q - 1$ first and then deriving q from them, as we did when searching for a safe prime. In the case of a safe prime $q = 2p + 1 > 5$, $|\mathbb{Z}_q^\times| = 2p$ and $\varphi(|\mathbb{Z}_q^\times|) = \varphi(2p) = \varphi(2) \cdot \varphi(p) = p - 1$. (If $q = 5$ and thus $p = 2$, $\varphi(4) = \varphi(2^2) = 2^1(2 - 1) = 2 \neq 2 - 1 = 1$.) If we ignore the elements 1 and $q - 1$, as the order of 1 is 1 and the order of $q - 1$ modulo q is 2, then half of all the $2p - 2 = 2(p - 1)$ remaining elements in the group are generators. This means that when we choose an element between 1 and $q - 1$ at random (excluding both ends), we have a 50% chance that this element generates the whole group. As a consequence, we need only two attempts on average to find a generator of the group $\mathbb{Z}_q^\times$, where q is a safe prime. Since $q - 1$ is the only element of order 2 in a cyclic group, $G < q - 1$ is a generator if $G^p \neq_q 1$. You can enable the generation of a generator according to this logic in the tool of the previous section. You can also convince yourself that half of all the elements between 1 and $q - 1$ are generators for all the safe primes greater than 5 and smaller than 100 (namely 7, 11, 23, 47, 59, and 83) using the repetition table of multiplicative groups.

For the sake of simplicity, I use only the multiplicative notation in the text parts of the following information boxes.

▼ **Generator of a subgroup**

Given a generator G of a group $\mathbb{G}$ of order n, the element $H = G^{\frac{n}{d}}$ generates the unique subgroup $\mathbb{H}$ of order d, where d is a divisor of n. (The order of H has to be d because the order of G wouldn't be n otherwise.) If the order of the desired subgroup $\mathbb{H}$ is prime, you don't need to find a generator of the supergroup $\mathbb{G}$ first. To find a generator of the subgroup with the prime order p, you can repeatedly choose a random element $R \in \mathbb{G}$ until $R^{\frac{n}{p}} \neq I$. Once you have found an element R for which this is the case, the element $R^{\frac{n}{p}}$ generates the subgroup of order p. Proof: Clearly, $(R^{\frac{n}{p}})^p = I$ because $A^n = I$ for all $A \in \mathbb{G}$ due to Lagrange's theorem. Since $(R^{\frac{n}{p}})^p = I$ and $(R^{\frac{n}{p}})^1 \neq I$, it follows from an earlier theorem with $e = 1$ that the order of $R^{\frac{n}{p}}$ is p.

▼ Determining the order of an element

Given a finite group $\mathbb{G}$ and the prime factorization of its order $n = p_1^{e_1} p_2^{e_2} \ldots p_l^{e_l}$, we want to determine the order d of an element A. Due to Lagrange's theorem, d divides n. Therefore, the multiset of the prime factors of n includes the multiset of the prime factors of d, which means that $d = p_1^{s_1} p_2^{s_2} \ldots p_l^{s_l}$, where $s_i \leq e_i$ for i from 1 to l. Since d is the order of A, $A^d = I$ and $A^{c \cdot d} = (A^d)^c = I^c = I$ for any integer c. This allows us to determine d by lowering each of the e_i to the lowest value s'_i for which A repeated $d' = \prod_{i=1}^{l} p_i^{s'_i}$ times still equals the identity element. The important insight is that we can tweak each of the exponents independently from the others. As long as $s'_i \geq s_i$ for each of the exponents, d' is still a multiple of d. As soon as one $s'_i < s_i$, d' is no longer a multiple of d, and thus $A^{d'} \neq I$. This gives us the following algorithm for determining the order of A:

Multiplicative | Additive | Both

Multiplicative

```
let $d := n$
for ($i$ from 1 to $l$) {
  $d := d / p_i^{e_i}$
  let $B := A^d$
  while ($B \neq I$) {
    $B := B^{p_i}$
    $d := d \cdot p_i$
  }
}
return $d$
```

Additive

```
let $d := n$
for ($i$ from 1 to $l$) {
  $d := d / p_i^{e_i}$
  let $B := dA$
  while ($B \neq O$) {
    $B := p_i B$
    $d := d \cdot p_i$
  }
}
return $d$
```

The algorithm does not require the group to be cyclic, but it requires that the prime factorization of the group's order is known. The following tool implements the above algorithm for arbitrarily large numbers as long as it can determine and then factorize the order of the group with Pollard's rho factorization algorithm. You can ignore the tab titled elliptic curve for now.

Multiplicative group | Elliptic curve | Both

Multiplicative group

Modulus m: 97 | Next prime | Previous prime | Element A: 56 | Random

Group order n = 96 = $2^5 \cdot 3$

$p_i^{e_i}$	d	B $=_m$ A^d
2^5	$3 = 3$	46
	$6 = 2 \cdot 3$	79
	$12 = 2^2 \cdot 3$	33
	$24 = 2^3 \cdot 3$	22
	$48 = 2^4 \cdot 3$	96
	$96 = 2^5 \cdot 3$	1
3^1	$32 = 2^5$	35

$p_i^{e_i}$	d	B $=_m$ A^d
	96 = $2^5 \cdot 3$	1

Element order d = 96

Elliptic curve

Modulus p:	97 Next prime Previous prime	A x value:	33 Random
Parameter a:	−1	A y even:	✓
Parameter b:	4		

Group order n = 90 = $2 \cdot 3^2 \cdot 5$

$p_i^{e_i}$	d	B = dA
2^1	45 = $3^2 \cdot 5$	(67, 0)
	90 = $2 \cdot 3^2 \cdot 5$	O
3^2	10 = $2 \cdot 5$	(65, 65)
	30 = $2 \cdot 3 \cdot 5$	O
5^1	6 = $2 \cdot 3$	(13, 65)
	30 = $2 \cdot 3 \cdot 5$	O

Element order d = 30

▼ A faster algorithm for finding a generator

Given a cyclic, commutative group $\mathbb{G}$ and the prime factorization of its order $n = p_1^{e_1} p_2^{e_2} \dots p_l^{e_l}$, we get a generator G by doing

Multiplicative | Additive | Both

Multiplicative

```
for ($i$ from 1 to $l$) {
    choose $A \in \mathbb{G}$ at random until $A^{\frac{n}{p_i}} \neq I$
    let $G_i := A^{\frac{n}{p_i^{e_i}}}$
}
return $G := \prod_{i=1}^{l} G_i$
```

Additive

```
for ($i$ from 1 to $l$) {
    choose $A \in \mathbb{G}$ at random until $\frac{n}{p_i}A \neq O$
    let $G_i := \frac{n}{p_i^{e_i}}A$
}
return $G := \sum_{i=1}^{l} G_i$
```

Before we can understand why this algorithm works, we must note that the above construction implies that

Multiplicative | Additive | Both

Multiplicative

$$G_i^{p_i^{e_i}} = (A^{\frac{n}{p_i^{e_i}}})^{p_i^{e_i}} = A^n = I \quad \text{and} \quad G_i^{p_i^{e_i-1}} = (A^{\frac{n}{p_i^{e_i}}})^{p_i^{e_i-1}} = A^{\frac{n}{p_i}} \neq I$$

Additive

$$p_i^{e_i} G_i = p_i^{e_i}(\frac{n}{p_i^{e_i}}A) = nA = O \quad \text{and} \quad p_i^{e_i-1} G_i = p_i^{e_i-1}(\frac{n}{p_i^{e_i}}A) = \frac{n}{p_i}A \neq O$$

It follows from an earlier theorem that the order of G_i is $p_i^{e_i}$. According to another theorem, the order of G is n because $\text{gcd}(|G_i|, |G_j|) = 1$ for all $i \neq j$. This means that if the algorithm terminates, G is indeed a generator of the group $\mathbb{G}$.

This algorithm is quite fast because we need on average at most two attempts to find a suitable A for each p_i. This is because:

1. The image $\mathbb{H} = f_{n/p_i}(\mathbb{G})$ of the power function f_{n/p_i} is a subgroup of $\mathbb{G}$: Since every element in $\mathbb{H}$ is of the form A^{n/p_i} for some $A \in \mathbb{G}$ and the group operation is commutative, $A_1^{n/p_i} \cdot A_2^{n/p_i} = (A_1 \cdot A_2)^{n/p_i} \in \mathbb{H}$ because $A_1 \cdot A_2 \in \mathbb{G}$. Thus, $\mathbb{H}$ is closed. Since every element in $\mathbb{G}$ is mapped to some element in $\mathbb{H}$, $\mathbb{H}$ is not empty. Therefore, $\mathbb{H}$ is indeed a subgroup of $\mathbb{G}$.
2. The order of $\mathbb{H}$ is p_i: Given a generator G of the group $\mathbb{G}$, $G^{n/p_i} \in \mathbb{H}$ has order p_i because $(G^{n/p_i})^{p_i} = G^n = I$. Since the group generated by G^{n/p_i} is included in $\mathbb{H}$ due to closure, $\mathbb{H}$ contains at least p_i elements. Since $\mathbb{G}$ is cyclic, the subgroup $\mathbb{H}$ is cyclic as well. So let H be a generator of $\mathbb{H}$ and $J \in f_{n/p_i}^{-1}(\{H\})$ be any of its preimages. Since the order of J can be at most n due to Lagrange's theorem, the order of $H = J^{n/p_i}$ can be at most p_i. Therefore, $\mathbb{H}$ contains exactly p_i elements.
3. The preimage $\mathbb{K} = f_{n/p_i}^{-1}(\{I\})$ of the identity element I is a subgroup of $\mathbb{G}$: $I \in \mathbb{K}$ because $I^{n/p_i} = I$. For any $A_1, A_2 \in \mathbb{K}$ (i.e. $A_1^{n/p_i} = I$ and $A_2^{n/p_i} = I$), $A_1 \cdot A_2 \in \mathbb{K}$ because $(A_1 \cdot A_2)^{n/p_i} = A_1^{n/p_i} \cdot A_2^{n/p_i} = I$ due to commutativity. Since $\mathbb{K}$ is non-empty and closed, $\mathbb{K}$ is a subgroup of $\mathbb{G}$. (As we will revisit later, $\mathbb{K}$ is the so-called kernel of f_{n/p_i}.)
4. f_{n/p_i} maps two inputs A and B to the same output if and only if they belong to the same coset of $\mathbb{K}$:
$A^{n/p_i} = B^{n/p_i} \iff A^{n/p_i}/B^{n/p_i} = I \iff (A/B)^{n/p_i} = I \iff A/B \in \mathbb{K} \iff A \in \mathbb{K} \cdot B.$
5. A^{n/p_i} is distributed uniformly over $\mathbb{H}$ as A is chosen at random from $\mathbb{G}$ and all cosets contain the same number of elements.
6. $P(A^{n/p_i} = I) = \frac{1}{p_i} \leq \frac{1}{2}$ because of the previous point and $|\mathbb{H}| = p_i$. Therefore, the probability of success when choosing A is $P(A^{n/p_i} \neq I) = 1 - \frac{1}{p_i} \geq \frac{1}{2}$. Since the number of attempts needed to find a suitable A for each p_i follows the geometric distribution, the expected number of attempts is $\frac{1}{1-1/p_i} \leq 2$.

You find a more elaborate analysis of the running time of this algorithm on page 328 of Victor Shoup's book. The following tool implements the above algorithm for cyclic groups. As we saw earlier, a multiplicative group is cyclic if and only if $\lambda(m) = \varphi(m)$, which is the case if and only if the modulus $m = 2, 4, p^e$, or $2p^e$ for an odd prime p and a positive integer e. In the case of elliptic curves, the situation is more complicated, and the tool simply aborts after 128 failed attempts to find a suitable A.

Multiplicative group | Elliptic curve | Both

Multiplicative group

Modulus m: 97 Next prime Previous prime Search

97 is prime

Group order n = 96 = $2^5 \cdot 3$

$p_i^{e_i}$	Random A	B $=_m$ A^{n/p_i}	$G_i =_m A^{n/p_i^{e_i}}$
2^5	46	96	45
3^1	14	61	61

Generator G $=_m$ 29

Elliptic curve

Modulus p: 97 Next prime Previous prime

Parameter b: 4

Parameter a: −1

Search

Group order n = 90 = $2 \cdot 3^2 \cdot 5$

$p_i^{e_i}$	Random A	B = (n / p_i)A	G_i = (n / $p_i^{e_i}$)A
2^1	(38, 18)	(67, 0)	(67, 0)
3^2	(14, 55)	(65, 65)	(36, 89)
5^1	(96, 95)	O	
	(46, 87)	(69, 83)	(69, 83)

Generator G = (48, 29)

Commutative rings

In this article, this and the next chapter build towards elliptic curves, which constitute the second popular way to construct linear one-way functions. However, the concepts presented in the following two chapters are also important in their own right. We will encounter repetition rings when we will discuss cryptosystems and finite fields when we will study coding theory.

Ring axioms

A ring is an algebraic structure which consists of a set $\mathbb{R}$ and two binary operations, called addition (+) and multiplication (·), which satisfy the following axioms (written in a compact notation using universal and existential quantifiers):

- Addition forms a commutative group:
  - **Closure**: $\forall\, a, b \in \mathbb{R}\; a + b \in \mathbb{R}$
  - **Associativity**: $\forall\, a, b, c \in \mathbb{R}\; (a + b) + c = a + (b + c)$
  - **Identity**: $\exists\, 0 \in \mathbb{R}\; \forall\, a \in \mathbb{R}\; a + 0 = a$
  - **Invertibility**: $\forall\, a \in \mathbb{R}\; \exists\, {-a} \in \mathbb{R}\; a + (-a) = 0$
  - **Commutativity**: $\forall\, a, b \in \mathbb{R}\; a + b = b + a$
- Multiplication forms a so-called monoid (a group without invertibility):
  - **Closure**: $\forall\, a, b \in \mathbb{R}\; a \cdot b \in \mathbb{R}$
  - **Associativity**: $\forall\, a, b, c \in \mathbb{R}\; (a \cdot b) \cdot c = a \cdot (b \cdot c)$
  - **Identity**: $\exists\, 1 \in \mathbb{R}\; \forall\, a \in \mathbb{R}\; a \cdot 1 = a = 1 \cdot a$
- Multiplication distributes over addition:
  - **Distributivity**: $\forall\, a, b, c \in \mathbb{R}\; a \cdot (b + c) = a \cdot b + a \cdot c$

As the title of this chapter suggests, we're interested only in commutative rings, whose multiplication operation is commutative:

- **Commutativity**: $\forall\, a, b \in \mathbb{R}\; a \cdot b = b \cdot a$

▼ **Remarks on the notation**

- **Lowercase**: I use lowercase letters to denote the elements of a ring instead of uppercase letters as I did in the case of groups. This makes it easier to tell groups and rings apart: As we will see in a moment, the inputs of our linear one-way functions form a ring, whereas the outputs belong to a group. The inputs of the linear one-way functions scale the outputs, and it's common to write coefficients in lowercase.
- **Identities**: Since the only rings we are interested in for now consist of integers, it's convenient to use the integers 0 and 1 to denote the additive and the multiplicative identity.
- **Precedence**: By convention, multiplication and division are evaluated before addition and subtraction, i.e. $a \cdot b + a \cdot c$ is to be interpreted as $(a \cdot b) + (a \cdot c)$. (Exponentiation and root extraction have an even higher precedence.)
- $\mathbb{R}$: This symbol is usually used to denote the set of real numbers. Since we talk only about integers throughout this article, I decided to use this symbol despite this conflict in meaning.

▼ **Universal and existential quantifiers**

When making a statement about the elements of a set, it is important to indicate whether the statment is true for all elements of the set or just for some elements of the set. In predicate logic, one uses the universal quantifier $\forall$ ("for all") for the former and the existential quantifier $\exists$ ("it exists") for the latter. The most important rule of inference is universal instantiation: If a statement is true for all elements of a set, it is true for any particular element of the set. The order of quantifiers matters: The identity axiom says that there is at least one element which is an identity for all elements of the set, whereas the invertibility axiom says that for all elements of the set at least one element exists which is an inverse for that particular element.

▼ **Multiplicative identity crisis**

I stated the group axioms for addition in the ring axioms above in a reduced form. Ignoring for now that the definition of invertibility is ambiguous, it still follows from these reduced group axioms that the right identity is unique and that the right identity is also a left identity (see the last chapter). As far as I can tell, we have to require that the right identity is also a left identity in the case of a monoid in order to ensure that the identity of the monoid is unique. (If there were two distinct identities E_1 and E_2, then $E_1 = E_1 \circ E_2 = E_2$, which is a contradiction.) Alternatively, we could require that both a left identity and a right identity exist in a monoid. (If there were two distinct left identities E_1 and E_2 and a right identity E, then $E_1 = E_1 \circ E = E$ and $E_2 = E_2 \circ E = E$, which contradicts that $E_1 \neq E_2$.)

Integers modulo m

As we saw earlier, the integers modulo an integer m form an additive group. Such a group is also closed under multiplication. Multiplication is still associative when the result is reduced to its remainder, and the integer 1 is the multiplicative identity. Since multiplication means repeated addition and addition is associative and commutative, multiplication distributes over addition:

$$a \cdot (b + c) =_m \underbrace{(b + c) + \ldots + (b + c)}_{a \text{ times}} =_m \underbrace{b + \ldots + b}_{a \text{ times}} + \underbrace{c + \ldots + c}_{a \text{ times}} =_m a \cdot b + a \cdot c$$

Therefore, the integers modulo m, which we write as $\mathbb{Z}/m\mathbb{Z}$ or $\mathbb{Z}_m$ (see above and below), form a (commutative) ring. You can use the following tool to perform modular addition and multiplication with arbitrarily large numbers, which is handy in cryptography:

Modulus m: 15 Next prime Previous prime

Exponent e: 15

Element a: 4

Element b: 7

a =$_m$ 4

b =$_m$ 7

a + b =$_m$ 4 + 7 =$_{15}$ 11

–b =$_m$ –7 =$_{15}$ 8

a – b =$_m$ 4 – 7 =$_{15}$ 12

a · b =$_m$ 4 · 7 =$_{15}$ 13

b^{-1} =$_m$ 7^{-1} =$_{15}$ 13

a / b =$_m$ 4 / 7 =$_{15}$ 7

a^e =$_m$ 4^{15} =$_{15}$ 4

▼ **Subrings**

A ring $\mathbb{S}$ is a subring of another ring $\mathbb{R}$ if $\mathbb{S}$ is a subset of $\mathbb{R}$ and the ring axioms hold for addition and multiplication restricted to $\mathbb{S}$. Since the modulus m is part of these operations (it defines what it means to be equivalent), the integers modulo m have no subrings other than the whole ring because the multiplicative identity 1 generates the whole additive group.

▼ **Zero ring**

The above axioms don't require that $0 \neq 1$. If the additive identity equals the multiplicative identity, all elements in the ring are equal to these identities:

$$\begin{aligned} \forall\, a \in \mathbb{R} \quad a &= a \cdot 1 && \text{since 1 is the multiplicative identity} \\ &= a \cdot 0 && \text{since } 1 = 0 \\ &= 0 && \text{see the next section} \end{aligned}$$

A ring which contains only a single element is a so-called zero ring. Since all zero rings are isomorphic to one another, we speak of *the* zero ring, denoted as $\{0\}$. The zero ring is not a subring of any non-zero ring because the additive identity is different from the multiplicative identity in non-zero rings, and a subring must have the same identities as its superring in order to satisfy the axioms for the same operations. We're not interested in such technicalities, and we will exclude the case where $0 = 1$ explicitly when defining fields, as it is commonly done.

Derived properties

Since addition forms a group and multiplication forms a monoid, the additive identity and the multiplicative identity of a ring are unique. It also follows that the additive inverse is unique. Other consequences of the ring axioms are:

Multiplication by zero

$$\boxed{\forall\, a \in \mathbb{R}\; a \cdot 0 = 0}$$

$$a \cdot 0 = a \cdot (0 + 0) = a \cdot 0 + a \cdot 0$$

Add $-(a \cdot 0)$ on both sides to get $0 = a \cdot 0$.

Multiplication by minus one

$$\boxed{\forall\, a \in \mathbb{R}\; a \cdot (-1) = -a}$$

$$\begin{aligned}
a \cdot 0 &= 0 && \text{as just proven} \\
a \cdot (1 + (-1)) &= 0 && \text{since } 0 = 1 + (-1) \\
a \cdot 1 + a \cdot (-1) &= 0 && \text{due to distributivity} \\
a + a \cdot (-1) &= 0 && \text{since } a \cdot 1 = a \\
a \cdot (-1) &= -a && \text{add } -a \text{ on both sides}
\end{aligned}$$

Units and zero divisors

An element of a ring may or may not have a multiplicative inverse. An element which has a multiplicative inverse is a so-called unit. An element a is called a zero divisor if there exists a non-zero element x so that $a \cdot x = 0$. (Since we care only about commutative rings, we don't distinguish between left and right zero divisors.) Every element of a finite ring is either a unit or a zero divisor, which can be seen as follows. Consider the function $f_a(x) = a \cdot x$, which multiplies the input by some element a. Since the multiplication of ring elements is closed, this function maps each element of $\mathbb{R}$ to another element of $\mathbb{R}$, which is usually written as $f_a\colon \mathbb{R} \to \mathbb{R}$. (In mathematical terms, $\mathbb{R}$ is both the domain and the codomain of the function.) Depending on a, f_a is either so-called injective or not:

- **Injective**: f_a maps distinct inputs to distinct outputs, which means that $x_1 \neq x_2$ implies that $f_a(x_1) \neq f_a(x_2)$. By contraposition, this also means that $f_a(x_1) = f_a(x_2)$ implies that $x_1 = x_2$. Since the codomain of the function (the set of potential outputs) is just as large as its domain (the set of inputs), there has to be an element x for each $y \in \mathbb{R}$ so that $f_a(x) = y$. (A function whose image covers its codomain is said to be surjective. A function which is both injective and surjective is said to be bijective.) As a consequence, f_a is a permutation. (We've already encountered permutations in the context of groups.) It follows that there is an element a^{-1} for which $f_a(a^{-1}) = a \cdot a^{-1} = 1$ and that there is no element z other than zero for which $f_a(z) = a \cdot z = 0$. (As we saw above, $f_a(0) = a \cdot 0 = 0$.) This means that if f_a is injective for some element a, then a is a unit and not a zero divisor.
- **Non-injective**: If f_a is not injective, there are distinct elements x_1 and x_2 for which $f_a(x_1) = f_a(x_2)$. Since $a \cdot x_1 = a \cdot x_2$ and $x_1 \neq x_2$, a is a zero divisor because $a \cdot x_1 - a \cdot x_2 = a \cdot (x_1 - x_2) = 0$ due to distributivity. (As we saw earlier, $f_a(x_1) = f_a(x_2)$ also implies that $f_a(x_1 + b) = f_a(x_2 + b)$ for any element b because $a \cdot (x_1 + b) = a \cdot x_1 + a \cdot b = a \cdot x_2 + a \cdot b = a \cdot (x_2 + b)$, which means that the outputs repeat after a collision.) Since at least two inputs map to the same output, there has to be at least one element $y \in \mathbb{R}$ for which there exists no element x so that $f_a(x) = a \cdot x = y$. If a had a multiplicative inverse a^{-1}, the element $a^{-1} \cdot y$ (which exists due to closure) would constitute such an input as $f_a(a^{-1} \cdot y) = a \cdot (a^{-1} \cdot y) = (a \cdot a^{-1}) \cdot y = y$ due to associativity. Therefore, zero divisors can't have a multiplicative inverse because the existence of a multiplicative inverse makes the mapping injective, whereas a zero divisor makes the mapping non-injective as $f_a(x) = 0$ for 0 and another element.

The existence of zero divisors is "undesirable" because it destroys the cancellation property: $a \cdot b = a \cdot c$ implies that $b = c$ only if a is not a zero divisor. If a is a zero divisor, then b equals c only up to a multiple of the smallest element z for which $a \cdot z = 0$. You can observe these facts in the operation table of multiplicative groups when you choose a non-prime modulus, such as 15.

▼ Integral domains

The statement that each element is either a unit or a zero divisor doesn't hold for rings with an infinite number of elements. For example, the integers $\mathbb{Z}$ form a ring under ordinary addition and multiplication. The only elements which have a multiplicative inverse in $\mathbb{Z}$ are $+1$ and -1, yet no integer other than 0 divides 0. A non-zero commutative ring where the product of any two non-zero elements is non-zero (i.e. 0 is the only zero divisor) is called an integral domain. Since every element of a finite ring is either a unit or a zero divisor, every finite integral domain is a field. Since we're interested only in finite algebraic structures, we don't care about integral domains. I still want to note that the cancellation property holds also for infinite rings: If a is not a zero divisor, then $a \cdot b = a \cdot c$ implies that $b = c$ because $a \cdot b = a \cdot c$ implies that $a \cdot b - a \cdot c = a \cdot (b - c) = 0$ and $b - c$ has to equal 0 given that a is not a zero divisor, which is the case only if $b = c$.

▼ Uniqueness of the multiplicative inverse in rings

If an element of a ring has a multiplicative inverse, the multiplicative inverse is unique. For finite rings, this fact follows directly from the argument above. (If a is a unit, f_a is a permutation, and thus there is only a single element for which $f_a(x) = 1$.) It's not difficult to show that this fact holds for infinite rings as well: After proving that a right inverse is also a left inverse (or simply requiring that the ring is commutative), any two inverses a_1^{-1} and a_2^{-1} of a are the same because $a_1^{-1} = a_1^{-1} \cdot 1 = a_1^{-1} \cdot (a \cdot a_2^{-1}) = (a_1^{-1} \cdot a) \cdot a_2^{-1} = 1 \cdot a_2^{-1} = a_2^{-1}$.

▼ **Multiplicative group of units**

Given a ring $\mathbb{R}$, the set of all units in $\mathbb{R}$ is usually denoted as $\mathbb{R}^\times$ (or as $\mathbb{R}$*). $\mathbb{R}^\times$ forms a group under the ring's multiplication:

- **Identity**: The multiplicative identity 1 is a unit and thus included in $\mathbb{R}^\times$ because it has a multiplicative inverse, namely itself.
- **Invertibility**: By definition, every unit has a multiplicative inverse. Since a right inverse is also a left inverse and vice versa, every inverse is itself invertible (i.e. if b is the inverse of a, then a is the inverse of b).
- **Closure**: The product of two units is another unit because for any $a, b \in \mathbb{R}^\times$, $(a \cdot b)^{-1} = b^{-1} \cdot a^{-1}$ as $(a \cdot b) \cdot (b^{-1} \cdot a^{-1}) = (a \cdot (b \cdot b^{-1})) \cdot a^{-1} = a \cdot a^{-1} = 1$. ($a^{-1}$ and b^{-1} exist because a and b are units.)
- **Associativity**: Since multiplication is the same operation as in the ring, it is still associative.

The notation for the group of units shouldn't surprise you. I've been using it since I introduced multiplicative groups: $\mathbb{Z}_m \to \mathbb{Z}_m^\times$.

Repetition ring

There are two reasons why we study rings. For one thing, fields, which we'll discuss in the next chapter in order to construct elliptic curves, are simply rings in which all non-zero elements are units. For another thing, we learned that you reach the identity element when you repeat an element A of a finite group often enough. From there on, you will get the same elements in the same order. This means that we can reduce the number of repetitions modulo the element's order $|A|$ without affecting the result:

Multiplicative | Additive | Both

Multiplicative

$$A^n = A^{n \,\%\, |A|} \text{ since } A^{|A|} = I$$

Additive

$$nA = (n \,\%\, |A|)A \text{ since } (|A|)A = O$$

When constructing a linear one-way function to be used in cryptosystems, we typically repeat a generator G. Since the function is linear, we can add and multiply two inputs a and b in the output space while knowing only the input b and the output $A = f(a)$:

Multiplicative | Additive | Both

Multiplicative

- **Addition**: $f(a + b) = G^{a+b} = G^a \cdot G^b = A \cdot G^b$
- **Multiplication**: $f(a \cdot b) = G^{a \cdot b} = (G^a)^b = A^b$

Additive

- **Addition**: $f(a + b) = (a + b)G = aG + bG = A + bG$
- **Multiplication**: $f(a \cdot b) = (a \cdot b)G = b(aG) = bA$

Since we compute the addition and the multiplication of inputs modulo the generator's order $|G|$, the inputs of our linear one-way function form a finite ring. I call it the repetition ring of the element which is repeated, even though I haven't found anyone else who uses this term. If the linear one-way function is constructed with a multiplicative group modulo m, it's very easy to confuse the group operation, which is multiplication modulo m, with multiplication in the repetition ring, which is computed modulo $|\mathbb{Z}_m^\times| = \varphi(m)$. In order to minimize confusion, I use uppercase letters for the elements of the group and lowercase letters for the elements of the ring. As mentioned earlier, this is also why I keep the modulus next to the equals sign. For example, such a linear one-way function transforms the equivalence relation $a \cdot b + c =_{\varphi(m)} d$ into $(G^a)^b \cdot G^c =_m G^d$, i.e. the latter relation holds if and only if the former does. Since the repetition ring is a ring, an element a may or may not have a multiplicative inverse. For example, $(G^a)^x =_m G$ has a solution x if and only if a has a multiplicative inverse modulo $\varphi(m)$, in which case $x =_{\varphi(m)} a^{-1}$.

Advanced concepts

While Wikipedia lists the following topics as basic concepts, they are fairly advanced for our standards. I put all of them in optional information boxes because we don't need any of them. They just provide an outlook into the bigger mathematical context, and having an elementary understanding of them makes it easier to read the sources that I've referenced throughout this article, including Wikipedia. They also explain the notation $\mathbb{Z}/m\mathbb{Z}$. I advise you to skip these boxes if you're not interested in any of this.

▼ **Homomorphisms**

A homomorphism is a function which maps the elements of one algebraic structure to the elements of a similar algebraic structure while preserving the relations between the elements in each structure under the operations of the corresponding structure. Unlike an isomorphism, which has to be invertible (see the group isomorphisms above), a homomorphism can map several elements of its domain to a single element of its codomain and doesn't have to reach all the elements of its codomain. In other words, a homomorphism has to be neither injective) nor surjective, while an isomorphism has to be both in order to be bijective. (Every isomorphism is a homomorphism, but a homomorphism is an isomorphism if and only if it is bijective.)

Given two rings $\mathbb{R}_1$ and $\mathbb{R}_2$, a ring homomorphism is a function $f\colon \mathbb{R}_1 \to \mathbb{R}_2$ which satisfies the following properties:

1. **Preservation of addition**: For all $a, b \in \mathbb{R}_1$, $f(a +_1 b) = f(a) +_2 f(b)$.
2. **Preservation of multiplication**: For all $a, b \in \mathbb{R}_1$, $f(a \cdot_1 b) = f(a) \cdot_2 f(b)$.
3. **Preservation of the multiplicative identity**: $f(1_1) = 1_2$.

The index indicates to which ring an operation or an identity element belongs. The third property prevents that all elements are mapped to 0_2. It follows from these properties that the additive identity and additive and multiplicative inverses are preserved.

▼ **Kernel**

The kernel of a homomorphism $f\colon \mathbb{R}_1 \to \mathbb{R}_2$ is the preimage of the additive identity element (or just the identity element in the case of groups): $\mathbb{K}_f = \{x \in \mathbb{R}_1 \mid f(x) = 0_2\} = f^{-1}(\{0_2\})$ (see these boxes if you don't understand these notations). (The kernel is usually written as $\ker(f)$ or $\ker f$ instead of $\mathbb{K}_f$, but I decided to make use of my artistic license again.)

The kernel $\mathbb{K}_f$ of a homomorphism $f\colon \mathbb{R}_1 \to \mathbb{R}_2$ has some interesting properties:

1. The kernel is a group under addition:
   - **Closure**: For all $a, b \in \mathbb{K}_f$, $a +_1 b \in \mathbb{K}_f$ because $f(a +_1 b) = f(a) +_2 f(b) = 0_2 +_2 0_2 = 0_2$.
   - **Associativity**: Inherited from the associativity of addition in $\mathbb{R}_1$.
   - **Identity**: $0_1 \in \mathbb{K}_f \iff f(0_1) = 0_2$ because you can cancel $f(0_1)$ from $f(0_1) = f(0_1 +_1 0_1) = f(0_1) +_2 f(0_1)$.
   - **Invertibility**: For all $a \in \mathbb{K}_f$, $-a \in \mathbb{K}_f$ because $0_2 = f(0_1) = f(a +_1 (-a)) = f(a) +_2 f(-a) = f(-a)$.
2. A homomorphism maps two inputs a and b to the same output if and only if they belong to the same coset of its kernel: $f(a) = f(b) \iff f(a) -_2 f(b) = 0_2 \iff f(a -_1 b) = 0_2 \iff a -_1 b \in \mathbb{K}_f \iff a \in \mathbb{K}_f +_1 b$. Obviously, $b \in \mathbb{K}_f +_1 b$ because $0_1 \in \mathbb{K}_f$. (You may have noticed that we've already used this fact earlier.)
3. f is injective if and only if $\mathbb{K}_f = \{0_1\}$. One direction of this equivalence is trivial to prove: If f is injective, no two inputs are mapped to the same output, including 0_2. If f is not injective, there are at least two distinct elements $a, b \in \mathbb{R}_1$ so that $f(a) = f(b)$. Therefore, $a -_1 b \in \mathbb{K}_f$ as shown in the previous point. Since $a \neq b$, $a -_1 b \neq 0_1$, which proves the contraposition of the other direction.

▼ **Ideals**

An ideal $\mathbb{I}$ is a subset of the elements of a ring $\mathbb{R}$ so that they form a group under addition and that they absorb all the elements of $\mathbb{R}$ under multiplication, which means that for every $r \in \mathbb{R}$ and every $i \in \mathbb{I}$, $r \cdot i \in \mathbb{I}$. (If the ring is not commutative, one has to distinguish between left ideals and right ideals, whose intersection consists of the two-sided ideals.) Ideals are usually not subrings, as they typically lack the multiplicative identity. For example, given the ring of integers $\mathbb{Z}$, the set of all the multiples of an integer n, which is commonly written as $n\mathbb{Z} = \{n \cdot z \mid z \in \mathbb{Z}\}$, is an ideal. This set is closed under addition (the sum of two multiples of n is another multiple of n), addition remains associative (the order in which you add multiples of n doesn't matter), it has an identity element (0 is a multiple of n because $n \cdot 0 = 0$), and every element has an additive inverse ($(n \cdot z) + (n \cdot (-z)) = n \cdot (z - z) = 0$). Additionally, if you multiply any multiple of n by any integer, you get another multiple of n.

The kernel $\mathbb{K}_f$ of a homomorphism f is an ideal because the kernel forms a group under addition (see the first property in the previous box) and the kernel absorbs all the elements under multiplication since for all $r \in \mathbb{R}_1$ and all $k \in \mathbb{K}_f$, $r \cdot_1 k \in \mathbb{K}_f$ as $f(r \cdot_1 k) = f(r) \cdot_2 f(k) = f(r) \cdot_2 0_2 = 0_2$.

▼ **Principal ideals**

A principal ideal is an ideal in a ring $\mathbb{R}$ which is generated by multiplying a single element a of $\mathbb{R}$ by every element of $\mathbb{R}$. Such a principal ideal is often written as $a\mathbb{R} = \{a \cdot r \mid r \in \mathbb{R}\}$. If the ring is not commutative, $a\mathbb{R}$ can be different from $\mathbb{R}a$. If they are the same, the ideal generated by a is also written as $\langle a \rangle$ or simply as (a). As we saw earlier when we discussed Bézout's identity, every linear combination of any two integers a and b is a multiple of $\gcd(a, b)$. This means that any ideal in the ring of integers $\mathbb{Z}$ can be generated by a single element, which makes $\mathbb{Z}$ a so-called principal ideal domain.

▼ **Quotient rings**

Given a ring $\mathbb{R}$ and a (two-sided) ideal $\mathbb{I}$ in $\mathbb{R}$, we can define an equivalence relation $=_\mathbb{I}$ so that for any $a, b \in \mathbb{R}$, $a =_\mathbb{I} b$ if and only if $a - b \in \mathbb{I}$. For $=_\mathbb{I}$ to be an equivalence relation, it must satisfy the following properties for any elements a, b, and $c \in \mathbb{R}$:

- **Reflexivity**: $a =_\mathbb{I} a$. Since every ideal includes 0, $a - a = 0 \in \mathbb{I}$.
- **Symmetry**: If $a =_\mathbb{I} b$, then $b =_\mathbb{I} a$. Since the elements of an ideal form an additive group, an ideal contains the additive inverse of each of its elements. Therefore, if $a - b \in \mathbb{I}$, then $b - a = -(a - b) \in \mathbb{I}$.
- **Transitivity**: If $a =_\mathbb{I} b$ and $b =_\mathbb{I} c$, then $a =_\mathbb{I} c$. Since $a - b \in \mathbb{I}$ and $b - c \in \mathbb{I}$, $(a - b) + (b - c) = a - c \in \mathbb{I}$ because every ideal is closed under addition (i.e. if $i \in \mathbb{I}$ and $j \in \mathbb{I}$, then $i + j \in \mathbb{I}$, which also means that $i =_\mathbb{I} j$).

A set of equivalent elements is called an equivalence class. Transitivity implies that any two equivalence classes overlap each other completely or not at all but not partially. Since the elements of both the ring and the ideal form a group under addition, the ideal is an additive subgroup of the ring. Since $a - b \in \mathbb{I} \iff a \in \mathbb{I} + b$ and $b \in \mathbb{I} + b$ because $0 \in \mathbb{I}$, two elements are identical if and only if they belong to the same additive cosets of the ideal. As shown earlier, cosets are either equal or disjoint.

Interestingly, $=_\mathbb{I}$ is also a congruence relation, which means that equivalent inputs result in equivalent outputs under addition and multiplication: Given $a_1, a_2, b_1, b_2 \in \mathbb{R}$ where $a_1 =_\mathbb{I} a_2$ and $b_1 =_\mathbb{I} b_2$, then

- $a_1 + b_1 =_\mathbb{I} a_2 + b_2$ because $a_1 - a_2 =_\mathbb{I} b_2 - b_1$ as we required that $a_1 - a_2 \in \mathbb{I}$ and $b_2 - b_1 \in \mathbb{I}$, and
- $a_1 \cdot b_1 =_\mathbb{I} a_2 \cdot b_2$ because $a_1 = a_2 + i_a$ for some $i_a \in \mathbb{I}$ and $b_1 = b_2 + i_b$ for some $i_b \in \mathbb{I}$, and thus $a_1 \cdot b_1 = (a_2 + i_a) \cdot (b_2 + i_b) = a_2 \cdot b_2 + a_2 \cdot i_b + i_a \cdot b_2 + i_a \cdot i_b$. Now $a_2 \cdot b_2 + a_2 \cdot i_b + i_a \cdot b_2 + i_a \cdot i_b =_\mathbb{I} a_2 \cdot b_2$ because $a_2 \cdot b_2 + a_2 \cdot i_b + i_a \cdot b_2 + i_a \cdot i_b - a_2 \cdot b_2 = a_2 \cdot i_b + i_a \cdot b_2 + i_a \cdot i_b \in \mathbb{I}$ since $\mathbb{I}$ absorbs multiplication and is closed under addition.

Therefore, addition and multiplication are well-defined for the cosets of $\mathbb{I}$: No matter which elements from two cosets you add or multiply, all results belong to the same coset. As a consequence, the function $f(x) = \mathbb{I} + x = \{i + x \mid i \in \mathbb{I}\}$ is a ring homomorphism, where $\mathbb{I}$ is the additive identity because $\mathbb{I} + \mathbb{I} = \mathbb{I}$ (closure under addition) and $a \cdot \mathbb{I} = \mathbb{I}$ for any $a \in \mathbb{R}$ (absorption of multiplication), and $\mathbb{I}$ is also the kernel of f because for all $i \in \mathbb{I}$ and only those is, $f(i) = \mathbb{I} + i = \mathbb{I}$:

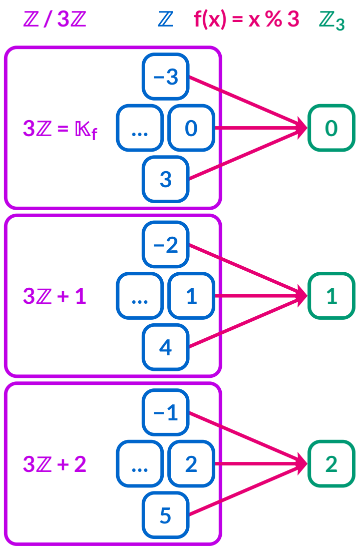
1. **Preservation of addition**: For all $a, b \in \mathbb{R}$, $f(a + b) = \mathbb{I} + (a + b) = \mathbb{I} + \mathbb{I} + (a + b) = (\mathbb{I} + a) + (\mathbb{I} + b) = f(a) + f(b)$.
2. **Preservation of multiplication**: For all $a, b \in \mathbb{R}$, $f(a \cdot b) = \mathbb{I} + (a \cdot b) = \mathbb{I} \cdot \mathbb{I} + \mathbb{I} \cdot b + a \cdot \mathbb{I} + a \cdot b = (\mathbb{I} + a) \cdot (\mathbb{I} + b) = f(a) \cdot f(b)$, where $\mathbb{A} \cdot \mathbb{B} = \{a \cdot b \mid a \in \mathbb{A}, b \in \mathbb{B}\}$ instead of the Cartesian product of sets.
3. **Preservation of the multiplicative identity**: $f(1) = \mathbb{I} + 1$, which is the multiplicative identity because $(\mathbb{I} + a) \cdot (\mathbb{I} + 1) = \mathbb{I} \cdot \mathbb{I} + \mathbb{I} \cdot 1 + a \cdot \mathbb{I} + a \cdot 1 = \mathbb{I} + a$ for any $a \in \mathbb{R}$.

Since the ideal $\mathbb{I}$ "divides" (partitions) the ring $\mathbb{R}$, the ring of cosets is called the quotient ring of $\mathbb{R}$ modulo $\mathbb{I}$, written as $\mathbb{R}/\mathbb{I}$. So not only does a ring homomorphism define a kernel, which is an ideal, an ideal also defines a homomorphism, whose kernel is the ideal itself. A ring homomorphism f maps all the elements of a coset of $\mathbb{K}_f$ to a single element because for all $a \in \mathbb{K}_f + b$, $f(a) = f(k_a + b) = f(k_a) + f(b) = 0 + f(b) = f(b)$ for some $k_a \in \mathbb{K}_f$. We thus have an invertible mapping between the cosets of $\mathbb{K}_f$ and the image of f, which means that the quotient ring of $\mathbb{R}$ modulo $\mathbb{K}_f$ is isomorphic to the image $f(\mathbb{R})$ of f:

$$\mathbb{R}/\mathbb{K}_f \cong f(\mathbb{R})$$

This theorem has some interesting edge cases. If the ideal (and thus the kernel) is the ring itself, all elements are mapped to zero, and the quotient ring is isomorphic to the zero ring: $\mathbb{R}/\mathbb{R} \cong \{0\}$. (This is the case if the multiplicative identity belongs to the ideal.) On the other hand, if the ideal contains only zero, the quotient ring is isomorphic to the ring itself: $\mathbb{R}/\{0\} \cong \mathbb{R}$.

In the scope of this article, we're interested only in integers modulo an integer m, where we have $\mathbb{Z}/m\mathbb{Z} \cong \mathbb{Z}_m$. As noted above, $m\mathbb{Z}$ is the ideal which consists of all multiples of m. We can visualize what we learned in this box as follows for $m = 3$:



The homomorphism $f(x) = x \mathbin{\%} 3$ maps the elements of the infinite ring $\mathbb{Z}$ (in blue) to the elements of the finite ring $\mathbb{Z}_3$ (in green). All and only the multiples of 3 are mapped to 0, which means that $\mathbb{K}_f = 3\mathbb{Z}$, which is an infinite ideal. The cosets of this ideal are the elements of the quotient ring $\mathbb{Z}/3\mathbb{Z}$ (in purple). Since the elements of each coset are mapped to a distinct element of $\mathbb{Z}_3$, we have $\mathbb{Z}/3\mathbb{Z} \cong \mathbb{Z}_3$.

▼ **Quotient groups**

The ring-related concepts from the previous boxes have group-related equivalents:

Ring	Group
Ring homomorphism	Group homomorphism
Kernel as preimage of 0	Kernel as preimage of E
Two-sided ideal	Normal subgroup
Quotient ring	Quotient group

An overview of ring- and group-related concepts.

Just like an additive subgroup of a ring has to satisfy an additional requirement to be an ideal, a subgroup $\mathbb{H}$ of a group $\mathbb{G}$ is a normal subgroup if and only if the left coset $G \circ \mathbb{H}$ equals the right coset $\mathbb{H} \circ G$ for all $G \in \mathbb{G}$. In commutative groups, every subgroup is normal. There's a great explanation on why we need this condition for non-commutative groups on Stack Exchange. See also this answer on Quora, which introduces conjugacy classes. Both answers use the symmetric group S_3 as an example.

Finite fields

Field axioms

A field is a commutative ring where all non-zero elements are units (i.e. have a multiplicative inverse) and $0 \neq 1$. While the elements of a ring form a commutative group only under addition, the non-zero elements of a field form a commutative group also under multiplication. Even though most of it is repetition, we can state the field axioms for a set $\mathbb{F}$ and two binary operations, called addition (+) and multiplication (·), as follows (written in a compact notation using universal and existential quantifiers):

- The elements of the field form a commutative group under addition:
  - **Closure**: $\forall\, a, b \in \mathbb{F}\ a + b \in \mathbb{F}$
  - **Associativity**: $\forall\, a, b, c \in \mathbb{F}\ (a + b) + c = a + (b + c)$
  - **Identity**: $\exists\, 0 \in \mathbb{F}\ \forall\, a \in \mathbb{F}\ a + 0 = a$
  - **Invertibility**: $\forall\, a \in \mathbb{F}\ \exists\, -a \in \mathbb{F}\ a + (-a) = 0$
  - **Commutativity**: $\forall\, a, b \in \mathbb{F}\ a + b = b + a$
- The non-zero elements of the field form a commutative group under multiplication:
  - **Closure**: $\forall\, a, b \in \mathbb{F}\ a \cdot b \in \mathbb{F}$

  - **Associativity**: $\forall\, a, b, c \in \mathbb{F}\ (a \cdot b) \cdot c = a \cdot (b \cdot c)$
  - **Identity**: $\exists\, 1 \in \mathbb{F}\ \forall\, a \in \mathbb{F}\ a \cdot 1 = a$
  - **Invertibility**: $\forall\, a \in \mathbb{F} \setminus \{0\}\ \exists\, a^{-1} \in \mathbb{F}\ a \cdot a^{-1} = 1$
  - **Commutativity**: $\forall\, a, b \in \mathbb{F}\ a \cdot b = b \cdot a$
- Multiplication distributes over addition:
  - **Distributivity**: $\forall\, a, b, c \in \mathbb{F}\ a \cdot (b + c) = a \cdot b + a \cdot c$
- The additive identity is different from the multiplicative identity:
  - **Non-triviality**: $0 \neq 1$

▼ **Infinite fields**

Examples for infinite fields are the rational numbers $\mathbb{Q}$, the real numbers $\mathbb{R}$, and the complex numbers $\mathbb{C}$.

▼ **Multiplicative group of a field**

Given that I quantified the various multiplication axioms over different sets, it's admittedly not obvious why the non-zero elements of a field form a group under multiplication. Clearly, if associativity, identity, and commutativity hold for all the elements of a field, then they also hold for a subset of them. Since every field is a ring, multiplication by 0 always results in 0. Therefore, the inverse of a non-zero element is another non-zero element because you wouldn't get 1 otherwise. As we saw above and will prove again below, invertibility implies that there are no non-trivial zero divisors. Consequently, the product of two non-zero elements is another non-zero element, which means that the non-zero elements are closed under multiplication.

▼ **Commutativity of addition**

Strictly speaking, commutativity of addition is not required as an axiom because it follows from the other axioms.

Integers modulo p

The ring $\mathbb{Z}_m$ is a field if and only if the modulus m is prime. As we saw earlier, an integer has a multiplicative inverse modulo m if and only if it is coprime with m. If m is prime, then all the positive integers smaller than m are relatively prime to m. If m is composite, its factors are zero divisors, which prevents $\mathbb{Z}_m$ from being a field. Excluding non-coprime integers from the field as we did in the case of multiplicative groups no longer works because repeatedly adding 1 to itself generates all the integers from 0 to $m - 1$. Put differently, filtering non-coprime integers breaks the closure of addition. I added such a filtering option to the operation table of additive groups just so that you can convince yourself that this is indeed the case.

▼ **Cyclicity of the additive group**

In the case of integers modulo a prime number p, the additive group is cyclic because the multiplicative identity 1 generates the whole group. In the article about coding theory, we will extend these so-called prime fields and see that the additive group of a proper extension field is not cyclic.

▼ **Cyclicity of the multiplicative group**

The multiplicative group of any field is cyclic: As mentioned but not yet proven earlier, a polynomial of degree $d > 0$ over any field evaluates to 0 for at most d distinct inputs. If we label the multiplicative group of a field $\mathbb{F}$ as $\mathbb{F}^\times$, where $\mathbb{F}^\times = \mathbb{F} \setminus \{0\}$, we have that $X^{|\mathbb{F}^\times|} = 1$ for all $X \in \mathbb{F}^\times$ due to Lagrange's theorem. The exponent of $\mathbb{F}^\times$ is the smallest positive integer n so that $X^n = 1$ for all $X \in \mathbb{F}^\times$. Since the polynomial $X^n - 1$ can evaluate to 0 for at most n elements but $X^n - 1 = 0$ for all $X \in \mathbb{F}^\times$, n cannot be smaller than $|\mathbb{F}^\times|$. As proven earlier, an element of order $n = |\mathbb{F}^\times|$ exists. Therefore, $\mathbb{F}^\times$ is cyclic.

Field notation

We are interested only in finite fields, i.e. fields which contain a finite number of elements. Finite fields are also called Galois fields (GF), named after Évariste Galois (1811 – 1832), who died at the age of 20 in a duel. The number of elements in a finite field is called its order. Without explaining why, the order of a finite field is always a prime power, and all the finite fields of the same order are isomorphic to one another. Given

$q = p^e$, where p is a prime number and e is a positive integer, the unique field of order q is denoted as $\mathbb{F}_q$ or $GF(q)$. For integers modulo a prime number, I will also use $\mathbb{Z}_p$, where p indicates that the modulus is prime.

Derived properties

Many properties, such as that the additive identity and the multiplicative identity are unique or that the additive inverse and the multiplicative inverse of an element are unique, follow from the fact that addition and multiplication (in the latter case without 0) form groups. Since every field is a ring, the properties we proved for rings still apply. Other consequences of the field axioms are:

Multiplicative inverse of minus one

$$\boxed{(-1)^{-1} = -1}$$

$$\begin{aligned} -(-1) &= (-1) \cdot (-1) && \text{as proven above, where } a = -1 \\ 1 &= (-1) \cdot (-1) && \text{using the double inverse theorem} \\ 1 \cdot (-1)^{-1} &= (-1) \cdot (-1) \cdot (-1)^{-1} && \text{multiplying both sides by } (-1)^{-1} \\ (-1)^{-1} &= -1 && \text{simplifying both sides as usual} \end{aligned}$$

Multiplicative inverse of additive inverse

$$\boxed{\forall\, a \in \mathbb{F}\ (-a)^{-1} = -(a^{-1})}$$

$$\begin{aligned} (-a)^{-1} &= ((-1) \cdot a)^{-1} && \text{using multiplication by minus one} \\ &= (-1)^{-1} \cdot a^{-1} && \text{using inversion of combination} \\ &= (-1) \cdot a^{-1} && \text{using multiplicative inverse of minus one} \\ &= -(a^{-1}) && \text{using multiplication by minus one} \end{aligned}$$

Stated in words, the multiplicative inverse of the additive inverse is the same as the additive inverse of the multiplicative inverse for every element of every field. What we knew from the rational numbers with ordinary division ($\frac{1}{-a} = -\frac{1}{a}$) thus holds in any field.

No non-trivial zero divisors

$$\boxed{\forall\, a, b \in \mathbb{F}\ \ a \cdot b = 0 \implies a = 0 \lor b = 0}$$

If $a = 0$, the above implication is true. If $a \neq 0$, then a has a multiplicative inverse, and thus

$$\begin{aligned} a \cdot b &= 0 && \text{starting with the premise of the above statement} \\ a^{-1} \cdot a \cdot b &= a^{-1} \cdot 0 && \text{multiplying both sides by } a^{-1} \\ b &= 0 && \text{simplifying both sides as usual,} \end{aligned}$$

which makes $a = 0 \lor b = 0$ true in the other case as well. This is also known as the zero-product property.

▼ **Material implication $\implies$**

The implication operator $\implies$ is a binary truth function with the following truth table (using $\top$ for true and $\bot$ for false):

P	Q	$P \implies Q$
$\bot$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$

The definition of $\implies$.

The technical term for this operation is material implication in order to distinguish it from how we use the term "implication" in everyday language. In logic, implication does not require causation. For example, $7 \text{ is odd} \implies 3 \text{ is prime}$ is a true statement.

▼ Logical disjunction $\lor$

The logical disjunction $\lor$ ("or") is true if and only if one of the operands are true ($\top$) instead of false ($\bot$):

P	Q	$P \lor Q$
$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$
$\top$	$\top$	$\top$

The definition of $\lor$.

Solutions of squares

$$\boxed{\forall\, a, b \in \mathbb{F}\; a^2 = b^2 \iff a = b \lor a = -b}$$

$$\begin{aligned} a^2 = b^2 &\iff a^2 - b^2 = 0 && \text{subtracting } b^2 \text{ from both sides} \\ &\iff (a - b) \cdot (a + b) = 0 && \text{using distributivity and commutativity} \\ &\iff a - b = 0 \lor a + b = 0 && \text{using no non-trivial zero divisors} \\ &\iff a = b \lor a = -b && \text{taking } b \text{ to the other side} \end{aligned}$$

$\iff$ stands for material equivalence ("if and only if"). The above theorem states that any two elements which result in the same element when being squared differ at most in their sign (i.e. one is the additive inverse of the other if they are not equal). Therefore, the equation $x^2 = c$ has at most two solutions for a given element c. (If $c = 0$, there is only one solution, namely 0, and as we will see in the next section, there is no solution for half of the non-zero elements.) A different way to see this is that in a field, the polynomial $f(x) = x^2 - c$ of degree two can have at most two solutions, as mentioned but not yet proved earlier.

▼ Yet another proof

In the field of integers modulo a prime p, the only square roots of 1 are 1 and -1 due to Euclid's lemma as we saw earlier. Thus:

$$a^2 =_p b^2 \iff a^2/b^2 =_p 1 \iff (a/b)^2 =_p 1 \iff a/b =_p \pm 1 \iff a =_p \pm b$$

The following three sections are important to solve the equation of elliptic curves of the form $y^2 = x^3 + a \cdot x + b$ over a finite field.

Quadratic residues

An integer a is called a quadratic residue modulo m if there exists an integer x so that $a =_m x^2$. If no such integer exists, a is called a quadratic non-residue. Just like 1 is technically prime but excluded from the set of prime numbers to make theorems involving prime numbers easier, 0 is technically a quadratic residue but excluded from the set of quadratic residues to make theorems about quadratic residues and non-residues easier. Given this definition and an odd prime number p, there are as many quadratic residues modulo p as quadratic non-residues modulo p. As we saw in the previous paragraph, the square of an integer equals the square of its additive inverse in any field, including the field $\mathbb{Z}_p$. Therefore, there are at most $\frac{p-1}{2}$ quadratic residues modulo p:

$$\begin{aligned} 1^2 &=_p (p-1)^2 \\ 2^2 &=_p (p-2)^2 \\ &\vdots \\ (\frac{p-1}{2})^2 &=_p (\frac{p+1}{2})^2 \end{aligned}$$

Since any two elements of the field which result in the same element when being squared differ at most in their sign, these $\frac{p-1}{2}$ squares are all different. Not accounting for 0, all the other $\frac{p-1}{2}$ elements in $\mathbb{Z}_p$ are quadratic non-residues.

▼ **Non-quadratic residue**

It would make much more sense to use the term "non-quadratic residue" instead of "quadratic non-residue" for the integers which are not a residue of a square number. Gauss is to blame for this. He's the one who introduced the terms "quadratic residue" and "quadratic non-residue" (respectively their Latin sources "residua quadratica" and "non-residua quadratica") in his book Disquisitiones Arithmeticae in 1801. These terms have been used since then, often even without "quadratic" if it's clear from the context what "residue" and "non-residue" refer to. Furthermore, "nonresidue" is frequently written without a hyphen.

▼ **Proof using group theory**

Given the field $\mathbb{Z}_p$ of integers modulo an odd prime p, we denote its multiplicative group, which consists of all its elements except 0, as $\mathbb{Z}_p^\times$. The squaring function $f(x) =_p x^2$ is a group homomorphism because $f(a \cdot b) =_p (a \cdot b)^2 =_p a^2 \cdot b^2 =_p f(a) \cdot f(b)$. As a consequence, the image of this function – the set of quadratic residues — is a subgroup of $\mathbb{Z}_p^\times$. The kernel $\mathbb{K}_f$ of this homomorphism is $\{1, -1\}$. f maps two inputs a and b to the same output if and only if they belong to the same coset $\mathbb{K}_f$: $a^2 =_p b^2 \iff a^2/b^2 =_p 1 \iff (a/b)^2 =_p 1 \iff a/b \in \mathbb{K}_f \iff a \in \mathbb{K}_f \cdot b$. Since $|\mathbb{K}_f| = 2$ and all cosets are of the same size, always two elements are mapped to the same output, which means that half of all elements are quadratic residues. This is no surprise since we already proved earlier that every element of the function's image has exactly two preimages.

▼ **Products of residues and non-residues**

The following table summarizes what you get when you multiply quadratic residues and non-residues modulo an odd prime p:

Factor 1	·	Factor 2	=	Product
Residue		Residue		Residue
Residue		Non-residue		Non-residue
Non-residue		Residue		Non-residue
Non-residue		Non-residue		Residue

The possible products of residues and non-residues.

Given that multiplication of integers is commutative, there are three cases to analyze:

1. **Residue · residue**: The set of quadratic residues is closed under multiplication because for any residues a^2 and b^2, their product $a^2 \cdot b^2 =_p (a \cdot b)^2$ is another quadratic residue. (We already saw in the previous box that the quadratic residues form a subgroup of $\mathbb{Z}_p^\times$. Since $1^2 =_p 1$, 1 is a residue. And the inverse of a residue is another residue: $(a^2)^{-1} =_p (a^{-1})^2$.)
2. **Residue · non-residue**: Since $\mathbb{Z}_p^\times$ is a multiplicative group, the function $f(x) =_p a \cdot x$ is a permutation for any $a \in \mathbb{Z}_p^\times$ (i.e. distinct inputs are mapped to distinct outputs). Since we know that exactly half of all elements in $\mathbb{Z}_p^\times$ are residues and that the product of two residues is another residue, the function $f(x) =_p b \cdot x$ has to map all non-residues to non-residues for any residue b. If just a single non-residue was mapped to a residue, more than half of all outputs would be residues.
3. **Non-residue · non-residue**: Since the product of a residue and a non-residue is a non-residue, the function $f(x) =_p c \cdot x$ has to map all non-residues to residues for any non-residue c. If just a single non-residue was mapped to a non-residue, more than half of all outputs would be non-residues.

The second point can also be proven as follows: Let b be a residue and c be a non-residue. If $b \cdot c$ was a residue, $b^{-1} \cdot b \cdot c =_p c$ would be a residue according to the first point, which is a contradiction. Therefore, $b \cdot c$ has to be a non-residue.

The second and the third case can also be derived using our advanced group concepts: Let $\mathbb{S}_p$ denote the subgroup of quadratic residues modulo the odd prime p, then the quadratic non-residues are a coset of $\mathbb{S}_p$. Let's call this coset of non-residues $\mathbb{T}_p$, then $\mathbb{S}_p \cdot s = \mathbb{S}_p$ for any $s \in \mathbb{S}_p$, and $\mathbb{S}_p \cdot t = \mathbb{T}_p$ for any $t \in \mathbb{T}_p$. Now $\mathbb{S}_p$ can also be interpreted as a kernel, where the resulting quotient group $\mathbb{Z}_p^\times/\mathbb{S}_p$ consists of the elements $\mathbb{S}_p$ and $\mathbb{T}_p$, where $\mathbb{S}_p$ is the identity element and the order of the group is 2. Since the order of $\mathbb{T}_p$ cannot be greater than 2, we have $\mathbb{T}_p \cdot \mathbb{T}_p = \mathbb{S}_p$. This means that $\mathbb{Z}_p^\times/\mathbb{S}_p \cong \mathbb{Z}_2^+$ as groups of prime order are cyclic and cyclic groups are isomorphic to the additive group of the same order.

For the above table to be correct, 0 has to be excluded from the set of quadratic residues. Otherwise, a residue times a non-residue can result in a residue because 0 times any number is 0. |residues| = |non-residues| wouldn't hold either otherwise.

Euler's criterion

Given an odd prime p, there's a simple formula for determining whether an integer a is a quadratic residue modulo p:

$$a^{\frac{p-1}{2}} =_p \begin{cases} 1 & \text{if } a \text{ is a quadratic residue modulo } p, \\ -1 & \text{if } a \text{ is a quadratic non-residue modulo } p, \\ 0 & \text{if } a \text{ is a multiple of } p. \end{cases}$$

This is known as Euler's criterion, named after Leonhard Euler (1707 – 1783). According to Fermat's little theorem, $a^{p-1} - 1 =_p 0$ for all integers a which are coprime with p. Since p is odd, $p - 1$ is even. Thus, the previous equation can be factored as follows:

$$(a^{\frac{p-1}{2}} - 1) \cdot (a^{\frac{p-1}{2}} + 1) =_p 0$$

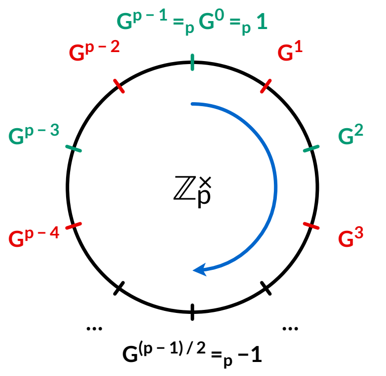
For every $a \in \mathbb{Z}_p$ except 0, either the first or the second factor has to be 0 because a field has no non-trivial zero divisors. If a is a quadratic residue, there is an integer x so that $a =_p x^2$. Since Fermat's little theorem holds for x as well, we have:

$$a^{\frac{p-1}{2}} =_p (x^2)^{\frac{p-1}{2}} =_p x^{p-1} =_p 1$$

Therefore, the first factor is 0 for the $\frac{p-1}{2}$ quadratic residues modulo p. Since $f(a) =_p a^{\frac{p-1}{2}} - 1$ can be 0 for at most $\frac{p-1}{2}$ elements of the field as mentioned earlier, the $\frac{p-1}{2}$ quadratic non-residues make the second factor 0.

▼ More intuitive proof

Since p is an odd prime, $\mathbb{Z}_p^\times$ is cyclic and its order is even. In a cyclic group of even order, an element $A = G^a$ is a quadratic residue if and only if the exponent a is even, where G is a generator of the group. The reason for this is that a quadratic residue is of the form $A = X^2 = (G^x)^2 = G^{2 \cdot x}$. Since the modulus $p - 1$ of the repetition ring is even, wrapping around cannot make an even exponent odd, i.e. $2 \cdot x \text{ mod } p - 1$ is still even. This can be visualized as follows:



The cyclic group $\mathbb{Z}_p^\times$ with quadratic residues in green and quadratic non-residues in red.

If you repeat any element $p - 1$ times, you get the 1. For the elements of the form $A = X^2$, this is the case already after $\frac{p-1}{2}$ repetitions. All other elements need to be squared one more time to get there (i.e. to have an exponent which is a multiple of $p - 1$). As explained earlier, the element midway through the cycle is -1.

▼ Legendre symbol

Adrien-Marie Legendre (1752 – 1833) introduced the following notation for Euler's criterion, called the Legendre symbol:

$$\left(\frac{a}{p}\right) =_p a^{\frac{p-1}{2}}$$

▼ Euler's criterion of a product

Given two integers a and b, Euler's criterion of their product equals the product of their Euler's criteria: $(a \cdot b)^{\frac{p-1}{2}} =_p a^{\frac{p-1}{2}} \cdot b^{\frac{p-1}{2}}$. This makes it much easier to analyze whether the products of residues and non-residues are residues or non-residues.

Square roots

In the previous two sections, we learned what a quadratic residue is and how we can determine whether an element is a quadratic residue using Euler's criterion. In this section, we want to compute a square root x of a quadratic residue a modulo an odd prime p so that $x^2 =_p a$. As shown earlier, a square root of a field element is unique up to its sign. We use the radical symbol and the plus-minus sign to write this as $\sqrt{a} =_p \pm x$. If $p + 1$ is divisible by 4 (i.e. $p =_4 3$), we can compute the square root of a as follows:

$$\sqrt{a} =_p \pm a^{\frac{p+1}{4}} \quad \text{because} \quad \big(\pm a^{\frac{p+1}{4}}\big)^2 =_p a^{\frac{p+1}{2}} =_p a \cdot a^{\frac{p-1}{2}} =_p a \cdot 1 =_p a$$

The following tool implements this calculation. In practice, it's common to skip Euler's criterion and to verify simply whether the square of the output is equal to the input. If $p + 1$ is not divisible by 4 (i.e. $p =_4 1$), you can use the Tonelli-Shanks algorithm. The box after that generalizes the Tonelli-Shanks algorithm to composite moduli, and the last box of this chapter explains why computing square roots is as difficult as factorizing integers. Since neither of them is relevant for our purposes, you can skip them both.

Modulus p: 11 Next prime Previous prime Input a: 5

Euler's criterion:	$a^{(p-1)/2} =_p 5^{(11-1)/2} =_{11} 1$
Square root of a:	$a^{(p+1)/4} =_p 5^{(11+1)/4} =_{11} \pm 4 =_{11} \pm 7$

▼ **Tonelli-Shanks algorithm**

The Tonelli-Shanks algorithm, named after Alberto Tonelli (1849 – 1920) and Daniel Shanks (1917 – 1996), finds a square root x of a quadratic residue a modulo any odd prime p by repeatedly adjusting an initial guess with the help of any quadratic non-residue b. The best known algorithm for finding a quadratic non-residue is to compute Euler's criterion for random elements of the field $\mathbb{Z}_p$ until you find a non-residue. Since half of all elements are non-residues, it takes on average just two attempts to find a non-residue (because this is a geometric distribution with a success probability of $\frac{1}{2}$). Since p is odd, $p - 1$ is even and can be written as $2^c \cdot d$, where $c > 0$ and d is odd. Given that a is a quadratic residue and b is a quadratic non-residue, we have:

$$\begin{aligned} p - 1 &= 2^c \cdot d \\ a^{\frac{p-1}{2}} &=_p a^{2^{c-1} \cdot d} =_p (a^d)^{(2^{c-1})} =_p 1 \\ b^{\frac{p-1}{2}} &=_p b^{2^{c-1} \cdot d} =_p (b^d)^{(2^{c-1})} =_p -1 \end{aligned}$$

We set $x =_p a^{\frac{d+1}{2}}$ initially. At the core of the algorithm is the expression $(x^2 \cdot a^{-1})^{(2^e)}$ with an exponent e. When $e = c - 1$, the expression evaluates to $(x^2 \cdot a^{-1})^{(2^{c-1})} =_p ((a^{\frac{d+1}{2}})^2 \cdot a^{-1})^{(2^{c-1})} =_p (a^{d+1} \cdot a^{-1})^{(2^{c-1})} =_p (a^d)^{(2^{c-1})} =_p 1$. Next, we want to decrement e while making sure that the expression keeps evaluating to 1. If we can get e to 0 so that $(x^2 \cdot a^{-1})^{(2^e)} =_p x^2 \cdot a^{-1} =_p 1$, then x is a square root of a. As explained when we discussed the Miller-Rabin primality test, 1 and -1 are the only possible square roots of 1 modulo a prime number. If the expression still evaluates to 1 for an e decremented by 1, we can keep decrementing e if e is still greater than 0 without doing anything. If, on the other hand, the expression evaluates to -1 for some $e < c - 1$, we have to adjust the value of x to keep the expression at 1. You get from -1 to 1 with a factor of -1. Since x is raised to the 2^{e+1}-th power, we look for a value y such that $y^{(2^{e+1})} =_p -1$. As it turns out, $y = (b^d)^{(2^{c-e-2})}$ does the trick ($c - e - 2 \geq 0$ because $e \leq c - 2$ at this point): $((b^d)^{(2^{c-e-2})})^{(2^{e+1})} =_p (b^d)^{(2^{c-1})} =_p -1$. Therefore, multiplying x by $(b^d)^{(2^{c-e-2})}$ brings the expression back to 1, thereby maintaining our loop invariant. If we haven't reached $e = 0$, we decrement e again.

The following tool implements the Tonelli-Shanks algorithm for arbitrarily large numbers. To make the output deterministic, it searches for a quadratic non-residue in the natural order of the field's elements. In an adversarial context, this can affect the performance of the algorithm negatively. If $c = 1$, $e = c - 1 = 0$ immediately, which means that the initial x is the final x, and thus the search for a quadratic non-residue can be skipped. The tool displays the adjusted x only in the next row. If the value in the third column is 1 (in green), the same x is used in the next row and the value in the last column is ignored (in gray). I included the ignored values in the last column anyway so that you can see that the last column does not depend on the input a and that it's easier to observe that each value in the last column is the square of the value above it. When you step through the quadratic residues of the field with the buttons next to the input a, you'll see that x sometimes has to be adjusted in every step and other times not at all. The algorithm presented here can be optimized in several ways. For example, computing the multiplicative inverse of a can be avoided (see Wikipedia), but I think my version of the algorithm is the easiest one to understand.

Modulus p: 97 Next prime Previous prime Input a: 11 Next residue Previous residue

p – 1 = $2^c \cdot$ d
97 – 1 = $2^5 \cdot 3$

Initial x $=_p a^{(d+1)/2} =_p 11^{(3+1)/2} =_{97} 24$

Potential non-residue b	Euler's criterion $b^{(p-1)/2}$
2	1

Potential non-residue b	Euler's criterion $b^{(p-1)/2}$
3	1
4	1
5	−1

e	x	$(x^2 \cdot a^{-1})^{(2^e)}$	$(b^d)^{(2^{c-e-2})}$
3	24	−1	28
2	90	1	8
1	90	−1	64
0	37	1	22

$x =_p \pm 37 =_{97} \pm 60$

$x^2 =_p 11$

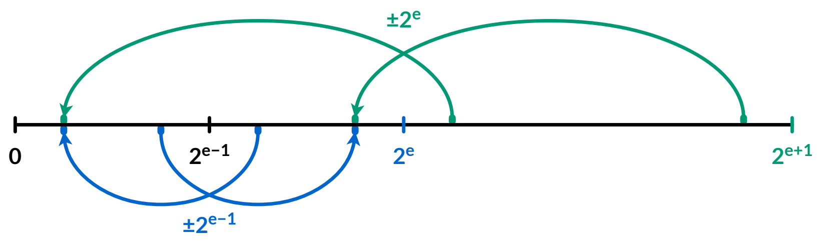
▼ **Square roots modulo composite numbers**

Given an integer a and a composite modulus m, how can we compute an x so that $x^2 =_m a$? If we know the prime factorization $\prod_{i=1}^{l} p_i^{e_i}$ of m, we can determine a square root modulo each of the factors since $\mathbb{Z}_m \cong \mathbb{Z}_{p_1^{e_1}} \times \ldots \times \mathbb{Z}_{p_l^{e_l}}$, and then use the Chinese remainder theorem to revert the isomorphism. If we don't know the prime factorization of m, determining whether a is a quadratic residue modulo m and computing square roots modulo m is generally (but not always) as difficult as factorizing m. This fact is used in some cryptosystems. So far, we have discussed only how to compute a square modulo an odd prime p. Modulo a prime power p^e for some integer $e \geq 1$, there are several cases to consider, which I derived mostly myself:

- $a =_{p^e} 0$: $x =_{p^e} cp^d$, where $d = \lceil \frac{e}{2} \rceil$ and $c \in \{0, 1, \ldots, p^{e-d} - 1\}$.
- $a =_{p^e} p^b$ for some integer $b \geq 1$: If b is even, $x =_{p^e} \pm(p^{b/2} + cp^d)$, where $d = \max(\lceil \frac{e}{2} \rceil, e - \frac{b}{2} - (1 - p \mathbin{\%} 2))$ and $c \in \{0, 1, \ldots, p^{e-d} - 1\}$, because $x^2 =_{p^e} (p^{b/2} + cp^d)^2 =_{p^e} p^b + 2cp^{b/2+d} + c^2p^{2d} =_{p^e} a$ due to the previous choice of d. Otherwise, there are no square roots because $p^b =_{p^e} x^2 \iff p^b + tp^e = p^b(1 + tp^{e-b}) = x^2$ for some integer t. As the prime factorization of an integer is unique, an integer has an integer square root if and only if every exponent of its prime factorization is even. Since $1 + tp^{e-b}$ is not a multiple of p (it is one off from a multiple of p), its prime factors don't include p, which leaves us with the odd exponent b for the prime factor p. Thus, $p^b(1 + tp^{e-b})$ has no square roots among the integers.
- $\gcd(a, p^e) = 0$:
  - p **is odd**: As we saw earlier, the multiplicative group modulo a power of an odd prime is cyclic, and Euler's criterion can be generalized to any cyclic multiplicative group by replacing $p - 1$ with $\varphi(p^e)$ in this box. With the same replacement, you can also generalize the Tonelli-Shanks algorithm to any cyclic multiplicative group. Alternatively, you can "lift" a solution modulo p^{e-1} to modulo p^e (recursively starting from p^1) as explained in section 12.5.2 of Victor Shoup's book.
  - p **is even** (i.e. $p = 2$): This case is almost always ignored, and the only algorithm I've found had no explanation. Fortunately, we can derive an algorithm for powers of 2 with the following observations:
    1. If $x^2 =_{p^e} a$ (i.e. $x^2 - a$ is a multiple of p^e), then $x^2 =_{p^d} a$ for some positive integer $d \leq e$ because any multiple of p^e is also a multiple of p^d.
    2. Whenever $x^2 =_{p^e} a$, $-x$ is also a solution because $(-x)^2 = ((-1)x)^2 = (-1)^2x^2 = (-(-1))x^2 = x^2$.
    3. Whenever $x^2 =_{2^e} a$, $x + 2^{e-1}$ is also a solution because $(x + 2^{e-1})^2 =_{2^e} x^2 + 2x2^{e-1} + 2^{2(e-1)} =_{2^e} x^2$.
    4. If $e \geq 3$, $x^2 =_{2^e} 1$ has exactly four solutions, namely $1, 1 + 2^{e-1}, -1$, and $-(1 + 2^{e-1}) =_{2^e} -1 + 2^{e-1}$, as $-2^{e-1} =_{2^e} 2^{e-1}$. Since $x^2 - 1 = (x + 1)(x - 1)$ is a multiple of 2^e and $(x + 1) - (x - 1) = 2$, both $x + 1$ and $x - 1$ are even (if they both were odd, their product couldn't be a multiple of 2^e), and one of them is not a multiple of 4. Consequently, the other one has to be a multiple of 2^{e-1} for their product to be a multiple of 2^e. Since the only multiples of 2^{e-1} modulo 2^e are 0 and 2^{e-1}, x has to be next to them, which gives us the four aforementioned solutions.
    5. In the case of $e = 3$, the solutions of $x^2 =_8 1$ are $1, 3, 5$, and 7 according to the previous point, which you can verify with the repetition table of multiplicative groups. As you can also see, 3, 5, and 7 have no square roots modulo 8.
    6. Combined with the first observation, $x^2 =_{2^e} a$ for some $e \geq 3$ has no solutions if a is odd and $a \mathbin{\%} 8 \neq 1$.
    7. The integers which are coprime with the modulus (i.e. all odd integers in the case of $m = 2^e$) form a multiplicative group. As we saw earlier, the power function (squaring in this case) is a homomorphism. As explained in the fourth observation, the size of its kernel is 4. Since a homomorphism maps two inputs to the same output if and only if they belong to the same coset of its kernel, an odd a has either four square roots or none modulo 2^e for some $e \geq 3$ since all cosets contain the same number of elements as the kernel.

Given a solution x, you obtain the other solutions by multiplying x with each element of the kernel. Since the odd integer x can be written as $1 + b \cdot 2$ for some integer b, $x(1 + 2^{e-1}) =_{2^e} x + (1 + b \cdot 2)2^{e-1} =_{2^e} x + 2^{e-1} + b \cdot 2^e =_{2^e} x + 2^{e-1}$, which matches our third observation.

8. The four solutions x, $x + 2^e$, $-x$, and $-x + 2^e$ to $x^2 =_{2^{e+1}} a$ for an odd a and some $e \geq 3$ are also solutions to $x^2 =_{2^e} a$ according to the first observation. In this case, however, $x + 2^e =_{2^e} x$ and $-x + 2^e =_{2^e} -x$, which means that the four solutions modulo 2^{e+1} map to two of the four solutions modulo 2^e. Moreover, either x or $x + 2^e$ modulo 2^{e+1} is already between 0 and 2^e, which means that one element stays the same when the two halves are mapped to one half by computing modulo 2^e. The same is true for $-x$ and $-x + 2^e$ as well, which we can depict as follows:



The four solutions to $x^2 =_{2^{e+1}} a$ in green are mapped to two of the four solutions modulo 2^e in blue.
When given any of the four solutions in blue, you may have to add 2^{e-1} to obtain a solution in green.

9. Given a solution x to $x^2 =_{2^e} a$, either x or $x + 2^{e-1}$ is also a solution to $x^2 =_{2^{e+1}} a$ because the two solutions which $x^2 =_{2^e} a$ and $x^2 =_{2^{e+1}} a$ have in common differ in their sign and not by 2^{e-1} according to the previous point.

		$\pm 2^{e-1}$
$\cdot(-1)$	x	$x + 2^{e-1}$
	$-x$	$-x + 2^{e-1}$

When given any of the four solutions to $x^2 =_{2^e} a$, the initial solution may be in the wrong column.
The row, on the other hand, doesn't matter: If x isn't a solution to $x^2 =_{2^{e+1}} a$, then neither is $2^e - x$.

These observations result in the following algorithm: Since $\gcd(a, 2^e) = 0$ in this branch, a and any solution x are odd. If $e = 1$, return $\{1\}$. If $e = 2$, return $\{1, 3\}$ if $a =_4 1$ and $\{\}$ otherwise. (There are only two solutions in this case because $x + 2 =_4 -x$ for $x \in \{1, 3\}$.) If $a \mathbin{\%} 8 \neq 1$, return $\{\}$. Set $x := 1$. For i from 3 to $e - 1$, add 2^{i-1} to x if $x^2 \neq_{2^{i+1}} a$. Return $\{x, 2^{e-1} - x, 2^{e-1} + x, 2^e - x\}$.

- $\gcd(a, p^e) = p^b$ for some integer $b \geq 1$: This means that $a =_{p^e} p^b c$ for some integer c, where $\gcd(c, p^e) = 0$. You get the solutions by multiplying each solution x_1 to $x^2 =_{p^e} p^b$ with each solution x_2 to $x^2 =_{p^e} c$. You find the solutions to these subproblems according to the previous two cases. Please note that these products are not all different from one another. Clearly, all these products are solutions to $x^2 =_{p^e} a$ because $(x_1 x_2)^2 =_{p^e} x_1^2 x_2^2 =_{p^e} p^b c =_{p^e} a$. It's less obvious why these are the only solutions. Since $p^b c =_{p^e} x^2 \iff p^b c + tp^e = p^b(c + tp^{e-b}) = x^2$ for some integer t and $c + tp^{e-b}$ is not a multiple of p because c isn't a multiple of p, the prime factorization of x has to include p with an exponent of $\frac{b}{2}$. But as noted in the second case, solutions to $x^2 =_{p^e} p^b$ can be shifted by multiples of p^d. The other prime factors are determined by $c + tp^{e-b} = x^2 \iff c =_{p^{e-b}} x^2$, which means that you can use the solutions to $x^2 =_{p^{e-b}} c$ instead of the solutions to $x^2 =_{p^e} c$.

The following tool computes the square roots of the given input modulo the specified composite integer as described above:

Modulus m: 72 Input a: 28

m = 72 = $2^3 \cdot 3^2$

p^e	Equation	Solutions
2^3	$x^2 =_8 4$	$x \in \{2, 6\}$
3^2	$x^2 =_9 1$	$x \in \{1, 8\}$
Combined:	$x^2 =_{72} 28$	$x \in \{10, 26, 46, 62\}$

▼ Why integer factorization isn't more difficult

In the previous box, we saw how the square roots modulo a composite number can be computed efficiently if the prime factorization of the modulus is known. This means that an efficient algorithm for factorizing integers would lead to an efficient algorithm for computing the square roots modulo a composite number. In this box, we show that the opposite is also true: An efficient algorithm for computing the square roots modulo a composite number would lead to an efficient algorithm for factorizing integers. Being able to reduce each of these two problems to the other one efficiently (i.e. in polynomial time) implies that these two problems are of similar computational complexity. Since no efficient algorithm is known for either of these two problems, both problems are believed to be difficult to solve for large inputs.

For the sake of simplicity, we assume that the number n, which we want to factorize, is the product of two odd primes p and q (i.e. $n = p \cdot q$). Since there are two solutions to $x^2 =_p a$ and two solutions to $x^2 =_q a$ as proven earlier, there are four solutions to $x^2 =_n a$ because $\mathbb{Z}_n \cong \mathbb{Z}_p \times \mathbb{Z}_q$. (Two of the solutions are the additive inverses of the other two.) Since we haven't specified whether the algorithm has to find one square root of the given input a or all of them, there are two possibilities to consider:

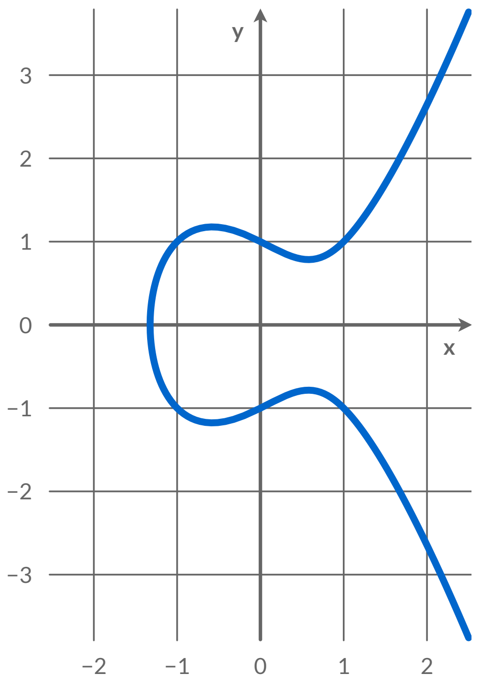
- The efficient algorithm for computing square roots modulo a composite number returns all four solutions: Compute the four square roots of an arbitrary quadratic residue a, such as 1. Choose distinct x and y from the four square roots so that $x \neq_n -y$. Since $x^2 =_n y^2 =_n a$, $x^2 - y^2 = (x + y)(x - y)$ is a multiple of n. However, n divides neither $x + y$ (as $x \neq_n -y$) nor $x - y$ (as $x \neq_n y$), which means that p has to divide one of the factors and q the other. (If the prime factorization of one of these factors contained both p and q, n would divide this factor.) Therefore, $\gcd(x \pm y, n)$, which can be computed efficiently with the Euclidean algorithm, equals either p or q, which means that we have successfully factorized n.
- The efficient algorithm for computing square roots modulo a composite number returns just one pair of solutions: Choose a random x between 0 and n, and compute $a =_n x^2$. By feeding this value a into the square-root-finding algorithm, you get a value y so that $y^2 =_n a$. Since the value x has been chosen randomly, the chance that $y =_n \pm x$ is exactly 50%. If $y =_n \pm x$, you repeat the process for another randomly chosen x until $x \neq_n \pm y$. Now again, $\gcd(x \pm y, n)$ equals either p or q.

Elliptic curves (EC)

Elliptic curves are the other popular way to construct linear one-way functions as they lead to shorter outputs and faster group operations than multiplicative groups for the same level of security. Many cryptosystems which rely on the discrete-logarithm problem of multiplicative groups have been adapted to work over elliptic curves. Since elliptic-curve cryptography (ECC) is in practice always implemented with standardized elliptic curves, we'll focus on how to work with given curve parameters instead of how to come up with new parameters. Unlike earlier chapters, this chapter covers just the bare minimum without any proofs.

Curve equation

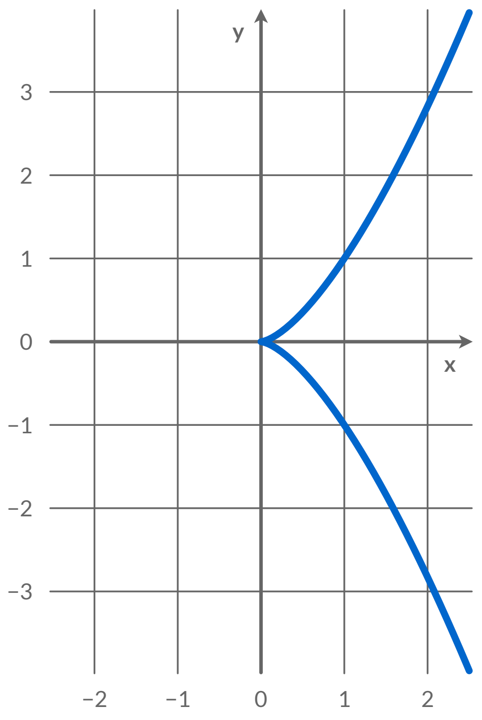
An elliptic curve consists of the two-dimensional points whose x- and y-coordinates satisfy the equation $y^2 = x^3 + a \cdot x + b$, where a and b are the parameters of the curve. By an appropriate change of variables (see below), almost any cubic curve can be written in the above form, which is known as the Weierstrass normal form, named after Karl Theodor Weierstrass (1815 – 1897). The equation is defined over a field, to which a, b, and all coordinates belong. Over the real numbers, an elliptic curve looks like this:



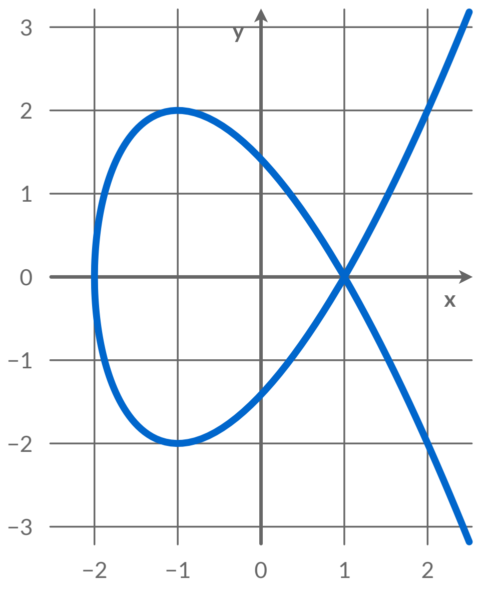
The curve $y^2 = x^3 - x + 1$ over the real numbers.

▼ **Smoothness**

For the group operation to be well defined (see the next section), the curve has to be smooth, which means every point on the curve has a unique tangent. An elliptic curve is smooth (non-singular) if it has no cusps and does not intersect itself. This is the case if and only if the polynomial on the right-hand side of the equation has no repeated roots, which is the case if and only if its discriminant $-4a^3 - 27b^2$ is not zero. Let's look at two examples, where this is not the case:



When $a = 0$ and $b = 0$, we have $y^2 = x^3$ with a cusp at $(0, 0)$. (0 is a root of multiplicity 3.)



When $a = -3$ and $b = 2$, we have $y^2 = x^3 - 3x + 2 = (x-1)^2(x+2)$. (1 is a root of multiplicity 2.)

▼ **What about x²?**

You may have noticed that the above equation defining elliptic curves misses the x^2 term. The reason for this is that a cubic polynomial of the form $c(z) = sz^3 + tz^2 + uz + v$ can be turned into a so-called depressed cubic of the form $d(x) = x^3 + ax + b$ by replacing z with $x - \frac{t}{3s}$, thereby shifting the input, which means $c(x - \frac{t}{3s}) = d(x) \iff c(z) = d(z + \frac{t}{3s})$:

$$\begin{aligned}
c(x - \frac{t}{3s}) &= s(x - \frac{t}{3s})^3 + t(x - \frac{t}{3s})^2 + u(x - \frac{t}{3s}) + v \\
&= s(x^3 - 3\frac{t}{3s}x^2 + 3\frac{t^2}{3^2s^2}x - \frac{t^3}{3^3s^3}) + t(x^2 - 2\frac{t}{3s}x + \frac{t^2}{3^2s^2}) + u(x - \frac{t}{3s}) + v \\
&= sx^3 + ((-t) + t)x^2 + (\frac{t^2}{3s} - 2\frac{t^2}{3s} + u)x + (-\frac{t^3}{3^3s^2}) + \frac{t^3}{3^2s^2} - u\frac{t}{3s} + v \\
&= sx^3 + \frac{3su - t^2}{3s}x + \frac{2t^3 - 9stu + 27s^2v}{27s^2}
\end{aligned}$$

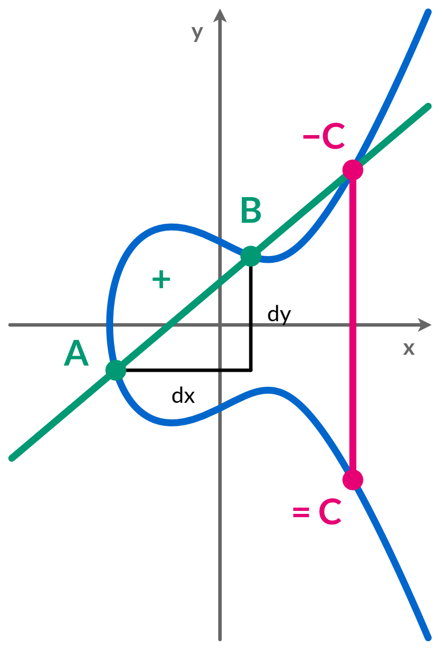
In order to get the promised form, we need to divide the polynomial by s, thereby scaling the output. Therefore, we have:

$$\begin{aligned}
a &= \frac{3su - t^2}{3s^2} \\
b &= \frac{2t^3 - 9stu + 27s^2v}{27s^3}
\end{aligned}$$

This change of variable doesn't work if the polynomial is defined over a finite field whose so-called characteristic is 3 because in this case, $3s = 0$, and we cannot divide by 0. In the case of elliptic curves, there are even more terms to consider, such as xy, which you cannot get rid of if the characteristic of the underlying field is 2. As you can see, things become complicated quickly.

Point addition

Together with the point at infinity, which we'll discuss in the next section, the points on an elliptic curve form a group under the operation described in this and the next section. Even though this operation has nothing to do with arithmetical addition, it's called point addition, and it's written using the additive notation. I denote the x-coordinate of a point A as A_x and its y-coordinate as A_y, which means that the point A is defined by the ordered pair (A_x, A_y). Given two points A and B where $A_x \neq B_x$, you get their sum $C = A + B$ by determining where the straight line which passes through A and B intersects the elliptic curve for the third time and then reflecting this point across the x axis:



The sum C of A and B where $A_x \neq B_x$.

As we will see in the next section, the point $-C = (C_x, -C_y)$ is the inverse of C. Since the curve equation is $y^2 = x^3 + a \cdot x + b$ and $y^2 = (-y)^2$ in any field, every elliptic curve is symmetric around the x axis, and thus C is guaranteed to be on the curve as well. In order to implement elliptic curves on a computer, we have to work out the geometric operation algebraically, i.e. as equations involving variables. The equation of a straight line is $y = s \cdot x + t$, where s is the slope of the line and t is the value at which the line crosses the y axis. The slope of the line which passes through A and B is calculated as the ratio of their vertical difference (marked as dy in the graph above) and their horizontal difference (marked as dx):

$$s = \frac{dy}{dx} = \frac{B_y - A_y}{B_x - A_x} = \frac{A_y - B_y}{A_x - B_x}$$

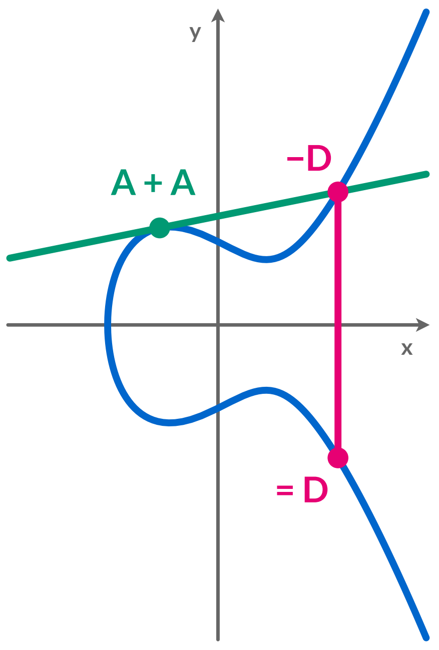
Since the formulas for C_x and C_y won't involve t, we don't have to determine its value. The line through A and B intersects the elliptic curve wherever they have the same y value for the same x value. This means that the x-coordinate of any intersection point has to fulfill $(sx + t)^2 = x^3 + ax + b$, which can be rewritten as $x^3 - s^2x^2 + (a - 2st)x + b - t^2 = 0$. Based on the geometric interpretation, we know that the solutions of this equation are A_x, B_x, and C_x. Since a value r is a so-called root of a polynomial $f(x)$ if and only if $(x - r)$ divides $f(x)$ (we'll discuss this in the article about coding theory, see Wikipedia for now), we have $x^3 - s^2x^2 + (a - 2st)x + b - t^2 = (x - A_x)(x - B_x)(x - C_x) = x^3 + (-A_x - B_x - C_x)x^2 + (A_xB_xC_x + A_xC_x + B_xC_x)x - A_xB_xC_x$. Since the coefficients of two equal polynomials are equal, we have $-s^2 = -A_x - B_x - C_x$, which we can solve for C_x:

$$C_x = s^2 - A_x - B_x$$

We get the value $-C_y$ by extrapolating the slope from A or B: $-C_y = A_y + s(C_x - A_x) = B_y + s(C_x - B_x)$. Therefore:

$$C_y = s(A_x - C_x) - A_y = s(B_x - C_x) - B_y$$

Based on the geometric construction and these formulas, it's apparent that this operation is commutative, i.e. $A + B = B + A$. We can move the points A and B closer and closer together until $A = B$, at which point the line through A and B becomes a tangent:



The sum D of A and A (called doubling).

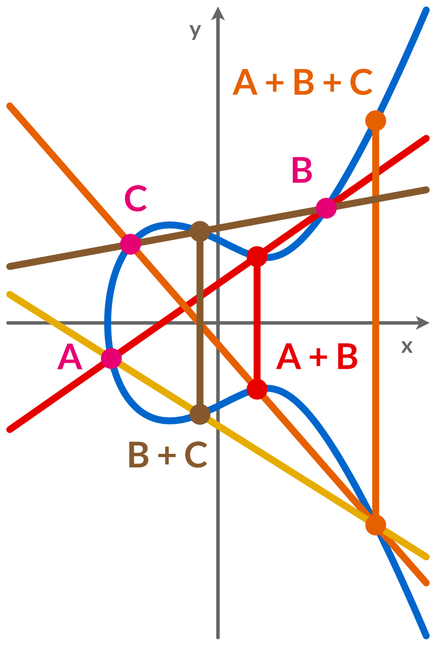
Since in this case $dx = B_x - A_x = 0$, we can no longer compute the slope s as $\frac{dy}{dx}$. Instead, we have to differentiate the function $f(x) = (x^3 + ax + b)^{\frac{1}{2}}$, whose derivative is $f'(x) = \frac{1}{2}(x^3 + ax + b)^{-\frac{1}{2}}(3x^2 + a) = \frac{3x^2+a}{2f(x)}$ according to the chain rule. We get the slope s' of the tangent by evaluating f' at A_x. The coordinates of the point D are then determined similarly to the point C above:

$$\begin{aligned} s' &= \frac{3A_x^2 + a}{2A_y} \\ D_x &= s'^2 - 2A_x \\ D_y &= s'(A_x - D_x) - A_y \end{aligned}$$

Please note that you can get a tangent not just by doubling a point. The same tangent is used when determining $(-D) + A = -A$ with the previous set of equations. In other words, the point $-C$ can be equal to the point A or B in the first graphic of this section.

▼ **Associativity of point addition**

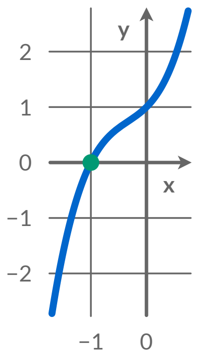
In order to form a group, point addition has to be associative, i.e. for any points A, B, and C, it has to be that $(A + B) + C = A + (B + C)$. Unfortunately, it's rather difficult to prove that this is always the case. You find explanations for this fact here and here. I'll just visualize what associativity means geometrically:



No matter whether you add $A + B$ or $B + C$ first, you end up in the same spot when determining $A + B + C$.

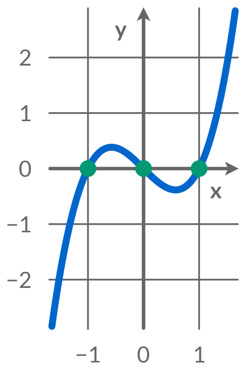
▼ Why is there always a third intersection?

As explained above, a point is at an intersection of an elliptic curve and a straight line if and only if its x-coordinate satisfies $x^3 - s^2x^2 + (a - 2st)x + b - t^2 = 0$. The left-hand side of this equation is a cubic function, which crosses the x axis at least once because it is continuous and goes from $f(-\infty) = -\infty$ to $f(\infty) = \infty$ or vice versa. This can be visualized as follows:



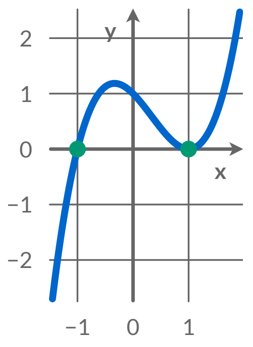
$$y = x^3 + x^2 + x + 1 = (x + 1)(x^2 + 1)$$

The cubic function $x^3 + x^2 + x + 1$ has only a single root because its factor $x^2 + 1$ has no roots among the real numbers (but i and $-i$ among the complex numbers). Other cubic functions cross the x axis three times, which is apparent from their factors:



$$y = x^3 - x = (x + 1)x(x - 1)$$

Instead of crossing the x axis again, a cubic function, such as $x^3 - x^2 - x + 1$, can just “touch” the x axis (at $(1, 0)$ in this case):

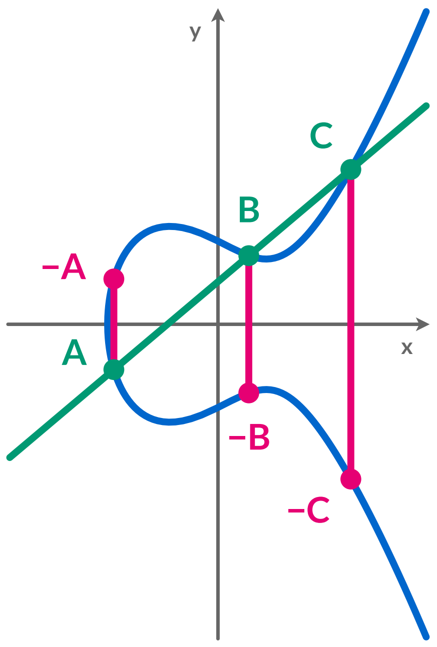


$y = x^3 - x^2 - x + 1 = (x + 1)(x - 1)^2$

As you can see from its factors, the root at $x = 1$ has a multiplicity of 2. This has to be the case because when you factor out two roots from a polynomial of degree 3, you're left with a polynomial of degree 1, which always has a root. Therefore, a cubic function has either one or three real roots, which don't have to be distinct. Since point addition involves two points (roots), you're guaranteed to find a third point (root) on the same line, where a point of tangency is simply counted twice.

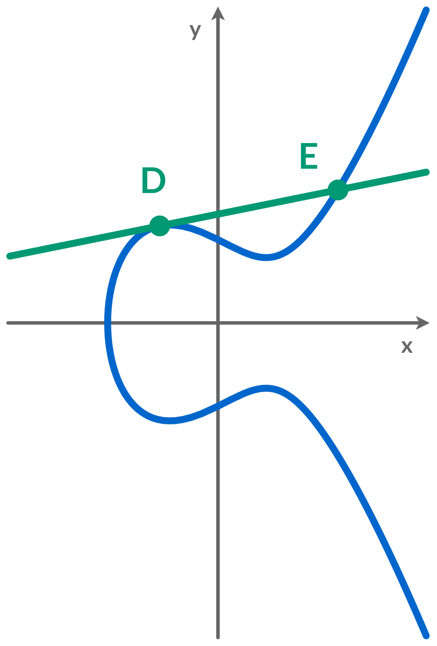
▼ **Why flip the intersection over the x axis?**

If we didn't reflect the third intersection across the x axis, we would have $A + B = C$, $A + C = B$, and $B + C = A$ because the collinearity of the three points is symmetric (i.e. if C lies on the same line as A and B, B lies on the same line as A and C):



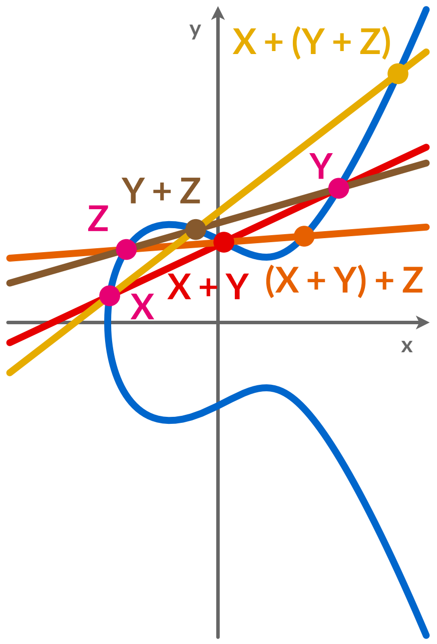
Three collinear points A, B, and C with their reflections.

By inserting the value of B from the second equation into the first equation, we would get $A + (A + C) = C$. Assuming that point addition forms a group (this is our goal after all), $A + A$ would have to equal the identity element. Since A is an arbitrary point on the elliptic curve, every point on the curve would have an order of 2, which is not what we want. We can see this more directly by repeating an element D. If $D + D$ equaled E, we would get back to D by adding D again, i.e. $E + D = D$:



The tangent at D intersects the elliptic curve at E.

This would make any such point E an identity element, which contradicts the uniqueness of the identity element in a group. Moreover, the modified operation would not be associative, which can be seen on the basis of the following counterexample:

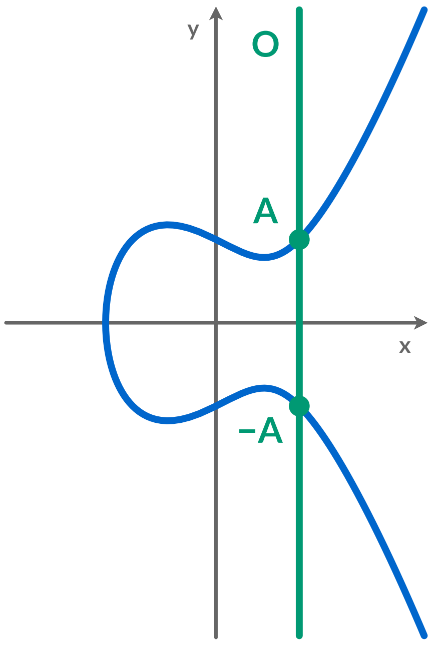


How associativity is violated by the modified operation. (I know that this graph is ugly, but it wasn't easy to keep all the intersections in the image.)

Now let's go back to the three collinear points A, B, and C at the beginning of this box. If $A + B = C$ and $A + C = B$, then $A + B - C = A - B + C$, which works only if each element is its own inverse. If, on the other hand, $A + B = -C$ and $A + C = -B$, which corresponds to the actual point addition above, we get the same expression in both cases, namely $A + B + C = O$, where O is the identity element, which we'll discuss next.

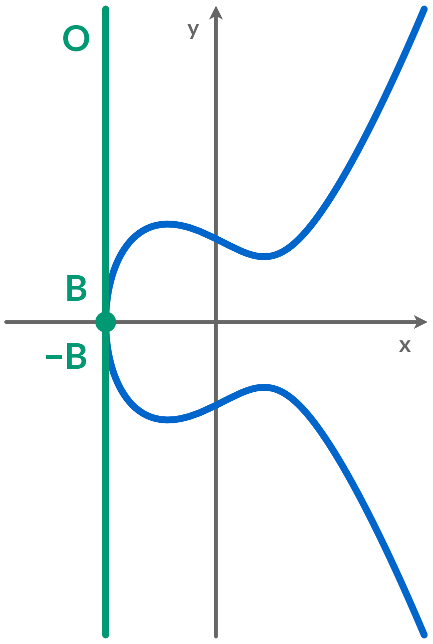
Point at infinity

Due to closure, we have to be able to add any two points together, including a point and its opposite reflected across the x axis. In this case, however, the line which passes through them intersects the elliptic curve only twice because every element of a field has at most two roots. Since a vertical line cannot be expressed as $y = s \cdot x + t$, this does not violate the explanation of why you always find a third intersection. In order to give vertical lines a third intersection, we introduce a special point, which is known as the point at infinity and usually denoted by the capital letter O. It is a single point which lies on every vertical line. Since parallel lines don't intersect in a normal Euclidean plane, elliptic curves are defined in a so-called projective plane, where any two distinct lines intersect at exactly one point. The point at infinity serves as the identity element of our group. The inverse of any element other than O is its reflection across the x axis, i.e. $-A = (A_x, -A_y)$. The point at infinity is its own inverse, though, i.e. $-O = O$. This is why you can't imagine O to be at $(0, \infty)$, unless you're willing to turn the plane into a cylinder in order to make $(0, -\infty) = (0, \infty)$. The idea of fitting a line through the points that we add breaks down anyway in the case of O. By being the identity element, it results in itself when it is added to itself, i.e. $O + O = O$. Therefore, don't imagine too much and simply treat O as a special case.



$A + (-A) = O$, but also $O + A = A$ by reflecting the other intersection.

Where the elliptic curve crosses the x axis, the point of intersection is its own inverse, and the tangent to the curve is vertical:

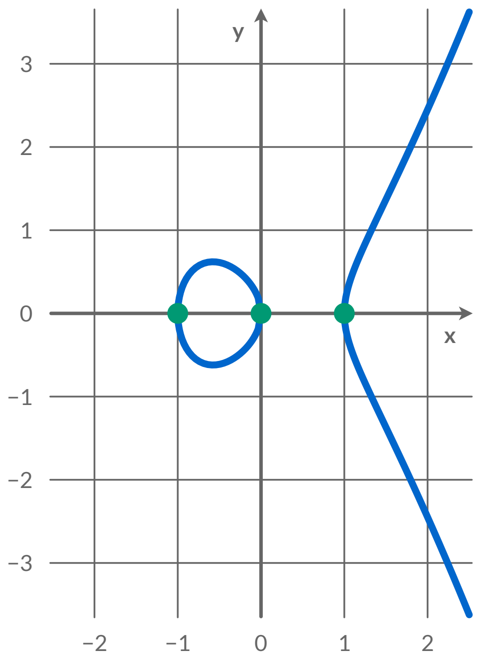


$B = -B$ and $B + B = O$.

With different parameters, an elliptic curve can cross the x axis more than once. You find an example of this in the first box below.

▼ **Example of separated curve**

An elliptic curve separates into two components if its discriminant $-4a^3 - 27b^2$ is positive:



The curve $y^2 = x^3 - x$ (i.e. $a = -1$ and $b = 0$).
This curve crosses the x axis at $x = -1, 0,$ and 1.

▼ Pseudocode for point addition

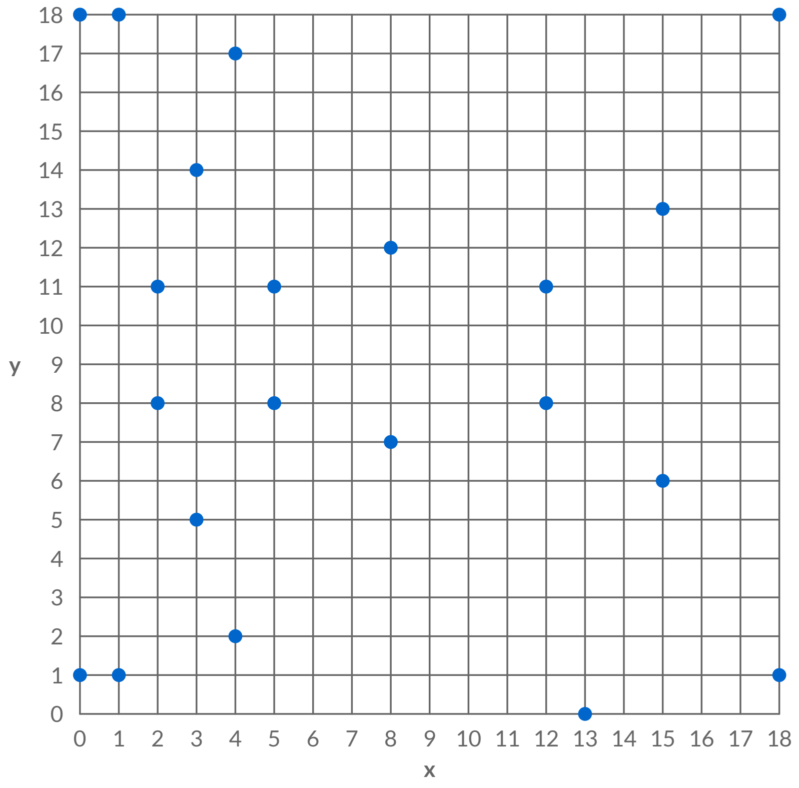
We can combine the various cases of point addition into the following pseudocode, but make sure to read the warnings below:

```
const $a := \ldots$
function $add(A, B)$ {
  if $(A = O)$ {
    return $B$
  } else if $(B = O)$ {
    return $A$
  } else if $(A = -B)$ {
    return $O$
  } else {
    let $s$
    if $(A = B)$ {
      $s := \frac{3A_x^2 + a}{2A_y}$
    } else {
      $s := \frac{B_y - A_y}{B_x - A_x}$
    }
    let $x := s^2 - A_x - B_x$
    return new $Point(x, s(A_x - x) - A_y)$
  }
}
```

While this code works and is exactly how I implemented point addition for the tools in this chapter, you should not use it for cryptographic purposes. In order to avoid timing attacks, cryptographic algorithms should run in constant time, i.e. perform the same operations no matter the inputs. How to achieve this is beyond the scope of this article. Common techniques include using homogeneous/projective coordinates instead of Cartesian coordinates and working with (twisted) Edwards curves, named after Harold Mortimer Edwards (1936 – 2020), instead of curves in Weierstrass form. The advantages of Edwards curves are that the formulas for addition and doubling are the same and that the identity element is an ordinary point.

Discrete curves

Elliptic curves as presented so far are not suitable for computers because computers can represent real numbers only with limited precision, which leads to inaccuracies. Fortunately, we can define elliptic curves also over finite fields, using the same equations for point addition as before, even though a tangent is no longer well defined. The elliptic curve we used so far looks as follows over $\mathbb{F}_{19}$:



The elliptic curve $y^2 =_{19} x^3 - x + 1$ over the field $\mathbb{F}_{19}$.

We can make several observations based on this example:

- **Symmetry**: The elliptic curve is still symmetric about the x axis, i.e. if y is a solution to the curve equation for some value of x, then so is $-y$ but no other value. Since $-y$ is usually represented as $p - y$ when computing modulo p, the symmetry that you see in the graph above is about $y = \frac{p}{2} = 9.5$ rather than about $y = 0$ because the values from -9 to -1 are displayed above and not below the values from 0 to 9. If we ignore extension fields, p is odd, and thus either y or $p - y$ is even while the other value is odd.
- **Gaps**: Not every element of a finite field has square roots. If $x^3 + a \cdot x + b$ results in a quadratic non-residue, the elliptic curve has no points at such an x value. In the example above, this is the case for $x \in \{6, 7, 9, 10, 11, 14, 16, 17\}$.
- **Order**: The point at $(13, 0)$ is its own inverse. Since its order is 2, we know that this group has an even number of points. (You will count only 21 points, but you also have to include the point at infinity.)

▼ Counting the points on elliptic curves

The elements of an elliptic-curve group are $\{(x, y) \in \mathbb{F} \times \mathbb{F} \mid y^2 = x^3 + a \cdot x + b \wedge 4a^3 + 27b^2 \neq 0\} \cup \{O\}$. In order to determine the group's order, we have to count the points on the elliptic curve. A naive approach of doing so is to determine for each possible x whether $x^3 + a \cdot x + b$ is a quadratic residue. If this is the case, you increase your counter by 2. If $x^3 + a \cdot x + b = 0$, you increase your counter by 1. However, this approach is computationally infeasible for the large finite fields that we need in elliptic-curve cryptography. René Schoof (born in 1955) published an efficient algorithm for counting the points on elliptic curves over finite fields in 1985. Unfortunately, his algorithm is too complicated for this introductory article.

Point calculator

The following tool implements point addition and repetition for elliptic curves in Weierstrass normal form over prime fields of arbitrary size. If you want a visualization of point addition on discrete curves, you can use this tool by Andrea Corbellini.

Modulus p:	19 [Next prime] [Previous prime]	B x value:	1
Parameter a:	−1	B y even:	✓
Parameter b:	4	Coefficient c:	15

A x value:	1
A y even:	✓

Elliptic curve: $y^2 = x^3 - x + 4$ over $\mathbb{F}_{19}$

A = (1, 2)
B = (1, 2)
A + B = (1, 2) + (1, 2) = (3, 16)
−B = −(1, 2) = (1, 17)
A − B = (1, 2) − (1, 2) = O
cA = 15(1, 2) = (15, 18)

Operation table

The following tool generates the operation table for elliptic curves in Weierstrass normal form over prime fields smaller than 100:

Modulus p: 7 Next prime Previous prime Parameter a: −1 Parameter b: 4

$y^2 = x^3 - x - 3$ over $\mathbb{F}_7$ with 10 points (10 = 2 · 5)

+	**O**	**(0, 2)**	**(0, 5)**	**(1, 2)**	**(1, 5)**	**(3, 0)**	**(4, 6)**	**(4, 1)**	**(6, 2)**	**(6, 5)**
O	O	(0, 2)	(0, 5)	(1, 2)	(1, 5)	(3, 0)	(4, 6)	(4, 1)	(6, 2)	(6, 5)
(0, 2)	(0, 2)	(4, 6)	O	(6, 5)	(1, 2)	(6, 2)	(4, 1)	(0, 5)	(1, 5)	(3, 0)
(0, 5)	(0, 5)	O	(4, 1)	(1, 5)	(6, 2)	(6, 5)	(0, 2)	(4, 6)	(3, 0)	(1, 2)
(1, 2)	(1, 2)	(6, 5)	(1, 5)	(0, 2)	O	(4, 1)	(3, 0)	(6, 2)	(0, 5)	(4, 6)
(1, 5)	(1, 5)	(1, 2)	(6, 2)	O	(0, 5)	(4, 6)	(6, 5)	(3, 0)	(4, 1)	(0, 2)
(3, 0)	(3, 0)	(6, 2)	(6, 5)	(4, 1)	(4, 6)	O	(1, 5)	(1, 2)	(0, 2)	(0, 5)
(4, 6)	(4, 6)	(4, 1)	(0, 2)	(3, 0)	(6, 5)	(1, 5)	(0, 5)	O	(1, 2)	(6, 2)
(4, 1)	(4, 1)	(0, 5)	(4, 6)	(6, 2)	(3, 0)	(1, 2)	O	(0, 2)	(6, 5)	(1, 5)
(6, 2)	(6, 2)	(1, 5)	(3, 0)	(0, 5)	(4, 1)	(0, 2)	(1, 2)	(6, 5)	(4, 6)	O
(6, 5)	(6, 5)	(3, 0)	(1, 2)	(4, 6)	(0, 2)	(0, 5)	(6, 2)	(1, 5)	O	(4, 1)

Since point addition is commutative, the above table is symmetric about the diagonal from the upper left to the lower right. The tool highlights the point at infinity with a green background and elements which are their own inverse (i.e. elements whose y-coordinate is 0) with a gray background. Since inverses are displayed next to each other and $A + B = C$ implies that $(-A) + (-B) = (-C)$, you have two by two squares with inverses diagonally opposite of each other:

∘	…	**B**	**−B**	…
⋮		⋮	⋮	
A	…	C	D	…
−A	…	−D	−C	…
⋮		⋮	⋮	

Symmetries within the operation table.

Repetition table

Since the points on an elliptic curve together with the point at infinity form a group under point addition, repeatedly adding a point to itself, which is known as point multiplication, generates a subgroup, whose order divides the order of the group according to Lagrange's theorem. You can verify that this is the case for moduli up to 100 with the following tool:

Modulus p: 7 Next prime Previous prime Repeat:

Parameter a:	−1	Order:	
Parameter b:	4	Totient:	

$y^2 = x^3 - x - 3$ over $\mathbb{F}_7$ with 10 points (10 = 2 · 5)

1A	2A	3A	4A	5A	6A	7A	8A	9A	10A
O									
(0, 2)	(4, 6)	(4, 1)	(0, 5)	O					
(0, 5)	(4, 1)	(4, 6)	(0, 2)	O					
(1, 2)	(0, 2)	(6, 5)	(4, 6)	(3, 0)	(4, 1)	(6, 2)	(0, 5)	(1, 5)	O
(1, 5)	(0, 5)	(6, 2)	(4, 1)	(3, 0)	(4, 6)	(6, 5)	(0, 2)	(1, 2)	O
(3, 0)	O								
(4, 6)	(0, 5)	(0, 2)	(4, 1)	O					
(4, 1)	(0, 2)	(0, 5)	(4, 6)	O					
(6, 2)	(4, 6)	(1, 2)	(0, 5)	(3, 0)	(0, 2)	(1, 5)	(4, 1)	(6, 5)	O
(6, 5)	(4, 1)	(1, 5)	(0, 2)	(3, 0)	(0, 5)	(1, 2)	(4, 6)	(6, 2)	O

If you play around with the above tool, you can observe the following facts:

- **Non-cyclic groups**: Not all elliptic curves are cyclic. Example: p = 7, a = 3, b = 0.
- **Groups of prime order**: There are elliptic curves whose order is prime, which is not possible in the case of multiplicative groups. Example: p = 7, a = 3, b = 5.
- **Elements of even order**: Whenever an element has an even order, the element at half its order is its own inverse and thus marked with a gray background. Example: p = 7, a = 3, b = 1.

Subgroup cosets

For the sake of completeness, the following tool shows the cosets of the subgroup which is generated by the given point (see above):

Modulus p:	7 [Next prime] [Previous prime]	A y even:	✓
Parameter a:	−1	Unique:	
Parameter b:	4	Delay:	0.5
A x value:	0 [Random]		

O	**(0, 2)**	**(4, 6)**	**(4, 1)**	**(0, 5)**
(0, 2)	(4, 6)	(4, 1)	(0, 5)	O
(0, 5)	O	(0, 2)	(4, 6)	(4, 1)
(1, 2)	(6, 5)	(3, 0)	(6, 2)	(1, 5)
(1, 5)	(1, 2)	(6, 5)	(3, 0)	(6, 2)
(3, 0)	(6, 2)	(1, 5)	(1, 2)	(6, 5)
(4, 6)	(4, 1)	(0, 5)	O	(0, 2)
(4, 1)	(0, 5)	O	(0, 2)	(4, 6)
(6, 2)	(1, 5)	(1, 2)	(6, 5)	(3, 0)
(6, 5)	(3, 0)	(6, 2)	(1, 5)	(1, 2)

Elliptic-curve discrete-logarithm problem (ECDLP)

Similar to the discrete-logarithm problem of multiplicative groups, it's believed that determining how many times a point on an elliptic curve has been repeated is computationally infeasible if the order of the point is large enough and the curve has no known weakness. This means that under the right conditions, you cannot find the coefficient k so that $kG = K$ in a reasonable amount of time, where G is a generator and K is an arbitrary point on the curve. Since many algorithms for finding the number of repetitions work in multiplicative groups and on elliptic curves, this problem is still known as the discrete-logarithm problem, even though “point-division problem” would be more accurate, given that the additive notation is used for groups based on elliptic curves.

Curve parameters

In elliptic-curve cryptography, an elliptic curve is defined by the following six parameters:

- p specifies the prime field $\mathbb{F}_p$ over which the elliptic curve is defined.
- a is the first parameter of the curve equation $y^2 = x^3 + a \cdot x + b$.
- b is the second parameter of the curve equation $y^2 = x^3 + a \cdot x + b$.
- G is the generator which all users of a particular curve share.
- n specifies the order of G, i.e. $n = |G|$ and $nG = O$.
- h is the cofactor of the subgroup generated by G.

As mentioned in the previous section, certain choices of parameters have known weaknesses and should thus be avoided. The bigger problem in the standardization of elliptic curves are potential weaknesses which are not yet publicly known. Ideally, all parameters are chosen in a predictable manner in order to reduce the risk of a hidden backdoor.

▼ **Curve secp256k1**

Bitcoin, Ethereum, and many other cryptocurrencies use an elliptic curve called `secp256k1` in their signature algorithm, which is defined in Standards for Efficient Cryptography (SEC) by the Standards for Efficient Cryptography Group (SECG). The letter `p` indicates that the curve is defined over a prime field. It is followed by the length of the prime p in bits. The letter `k` stands for Koblitz curve, named after Neal Koblitz (born in 1948), which allows for especially efficient implementations. The `1` at the end is just a sequence number. `secp256k1` has the following curve parameters in hexadecimal notation (see section 2.4.1 on page 9):

- $p =$ `FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F` $= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$
  (The proximity to 2^{256} allows for faster implementations of the modulo operation.)
- $a = 0$
- $b = 7$
- $G =$ (`79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798`, `483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8`)
- $n =$ `FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141`
- $h = 1$

You can click on the hexadecimal numbers to convert them to decimal numbers with the tool in the next box. You can verify with the Miller-Rabin primality test that p is indeed prime. Given the above a and b, the curve equation is $y^2 =_p x^3 + 7$. While counting the points on elliptic curves is complicated, you can verify with the point calculator above that $nG = O$ (the tool also recovers the y-coordinate of G correctly). Since $p \mathbin{\%} 4 = 3$, computing square roots modulo p is simple. Since n is prime and G is not the point at infinity, the order of G cannot be smaller than n. Since the cofactor is 1, G generates all points on the elliptic curve. Since n is prime, every element except O is a generator. And since n is odd, we know that the curve has no point with a y-coordinate of 0.

▼ **Integer conversion**

The following tools converts the given integer to the decimal and the hexadecimal numeral system in the preferred formatting:

Integer: 15

Decimal: 15

Hexadecimal: 0xF

DL algorithms

We now have two families of finite groups for which it is presumably difficult to determine how many times an element has been repeated if the parameters of the group have been chosen carefully:

- **Multiplicative groups**: Given a modulus m, a generator G, and an element K, it’s hard to find the integer k so that $G^k =_m K$.

- **Elliptic curves**: Given an elliptic curve, a generator G, and a point K, it's hard to find the integer k so that $kG = K$.

In both groups, this is known as the discrete-logarithm problem (DLP); in the latter case also as the elliptic-curve discrete-logarithm problem (ECDLP). Since group operations are associative and repeating an element is fast, this gives us the linear one-way function we were aiming for. In this chapter, we'll study some of the best known algorithms for solving the discrete-logarithm problem. This allows us to understand why the number of steps we need to solve a discrete-logarithm problem is in the order of the square root of the largest prime factor of the generator's order, and why it's even less in the case of multiplicative groups. Before we do so, we'll revisit how an element of a group can be repeated efficiently. This gives you a better feeling for why one direction of the linear one-way function is so much easier than the other.

Repetition revisited

As explained earlier, repeating a generator G k times takes just $\log_2(k)$ steps, which corresponds to the bit-length of k. The algorithm below uses the fact that one can "rebuild" a binary number by doubling, thereby shifting the current binary digits (bits) one to the left, and by adding one, thereby changing the least-significant bit from 0 to 1. Using subscripts to denote the l bits of k from right to left so that $k = (k_{l-1} \ldots k_0)_2 = \sum_{i=0}^{l-1} k_i 2^i$ and $k_{l-1} = 1$, we can rebuild the positive integer k in the exponent or coefficient of G by combining the current result with itself and by combining it with G:

Multiplicative | Additive | Both

Multiplicative

```
function $binaryRepeat(G, k)$ {
  let $K := G$
  for ($i$ from $l - 2$ to $0$) {
    $K := K^2$
    if ($k_i = 1$) {
      $K := K \cdot G$
    }
  }
  return $K$
}
```

Additive

```
function $binaryRepeat(G, k)$ {
  let $K := G$
  for ($i$ from $l - 2$ to $0$) {
    $K := 2K$
    if ($k_i = 1$) {
      $K := K + G$
    }
  }
  return $K$
}
```

This algorithm is similar to the one I covered earlier, but this time we work explicitly with a binary representation of k and process its bits from the left to the right instead of the other way around. The following tool implements this algorithm for multiplicative groups and elliptic curves. You can change which group is being displayed in all the remaining tools by double-clicking on its tab.

Multiplicative group | Elliptic curve | Both

Multiplicative group

Modulus m:	97	G's order n:	96
Generator G:	56	Input k:	42

k = 101010_2 (6 bits)

k_i	$G^k =_m K$	Action	Exponent k in binary
1	$G^1 =_m 56$		1
0	$G^2 =_m 32$	square	10
1	$G^5 =_m 17$	square and multiply	101
0	$G^{10} =_m 95$	square	1010

k_i	$G^k =_m K$	Action	Exponent k in binary
1	$G^{21} =_m 30$	square and multiply	10101
0	$G^{42} =_m 27$	square	101010

Elliptic curve

Modulus p:	97	G y even:	✓
Parameter a:	1	G's order n:	89
Parameter b:	4	Input k:	42
G x value:	56		

k = 101010_2 (6 bits)

k_i	kG = K	Action	Coefficient k in binary
1	1G = (56, 94)		1
0	2G = (82, 3)	double	10
1	5G = (84, 5)	double and add	101
0	10G = (73, 30)	double	1010
1	21G = (1, 54)	double and add	10101
0	42G = (31, 85)	double	101010

▼ Limitations of the above tool

The above tool shares its state with the other tools in this chapter. Since some algorithms for solving the discrete-logarithm problem work only if the order of the generator is known, we have to determine it even in the tools that don't need it. In order to determine the order of an element, you have to know the prime factorization of the group's order, which is itself not always easy to determine. Hence, the tools in this chapter have the following limitations:

- **Multiplicative groups**: In order to determine the order of a multiplicative group, we have to evaluate Euler's totient function for its modulus m, which requires the prime factorization of m. If m isn't prime, the tools in this chapter perform up to 100'000 rounds of Pollard's rho factorization algorithm, which is usually enough to factorize the product of two 35-bit primes, i.e. numbers with up to 21 decimal digits. If successful, it performs at most another 100'000 rounds of the same algorithm to factorize the group's order. If either of these factorizations fail, you have to enter the order of G yourself, which gives you the option of choosing the modulus in a smart way. If you just want to know the result of a modular exponentiation, you can use the calculator for rings above, which doesn't have this limitation.
- **Elliptic curves**: Since I didn't implement Schoof's algorithm, the tool counts the number of points only for elliptic curves whose modulus p is smaller than 100'000. Once the order of the group has been determined, it also performs up to 100'000 rounds of Pollard's rho factorization algorithm in order to determine the order of G. If p > 100'000 or the factorization fails, you have to enter the order of G yourself, which you can determine with SageMath as explained in the next box. If you just want to know the result of a point multiplication, you can use the point calculator above, which doesn't have this limitation.

▼ How to use the playground of SageMath

SageMath is an open-source software library for mathematics. You can use its playground for many things, which includes computations on elliptic curves over finite fields (see the documentation of `cardinality`, `order`, `lift_x`, and `plot`):

```
# Determine the order of the given curve:
EllipticCurve(Zmod(97), [1, 4]).cardinality()

# Determine the order of the given element:
EllipticCurve(Zmod(97), [1, 4]).lift_x(56).order()

# List all the points with the given x-coordinate:
EllipticCurve(Zmod(97), [1, 4]).lift_x(56, all=True)

# Plot the points of the discrete curve (only for small curves):
EllipticCurve(Zmod(97), [1, 4]).plot()
```

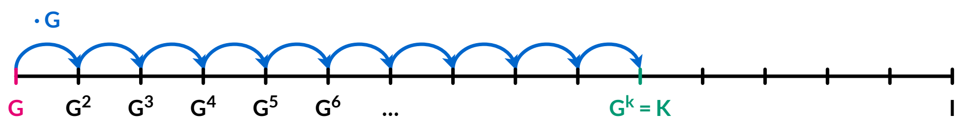
```
# Plot the elliptic curve with the given parameters over the real numbers:
EllipticCurve([1, 4]).plot()
```

Exhaustive search

The simplest algorithm for solving the discrete-logarithm problem is to simply try all possible values for k. The effort for doing so scales exponentially with the bit-length of k. The search for the input k so that you get the output K can be visualized as follows:

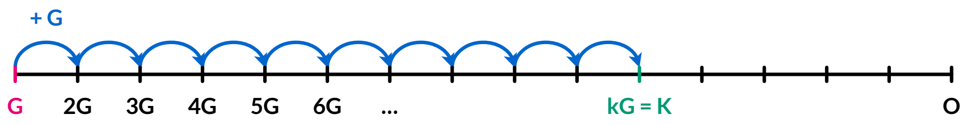
Multiplicative | Additive | Both

Multiplicative



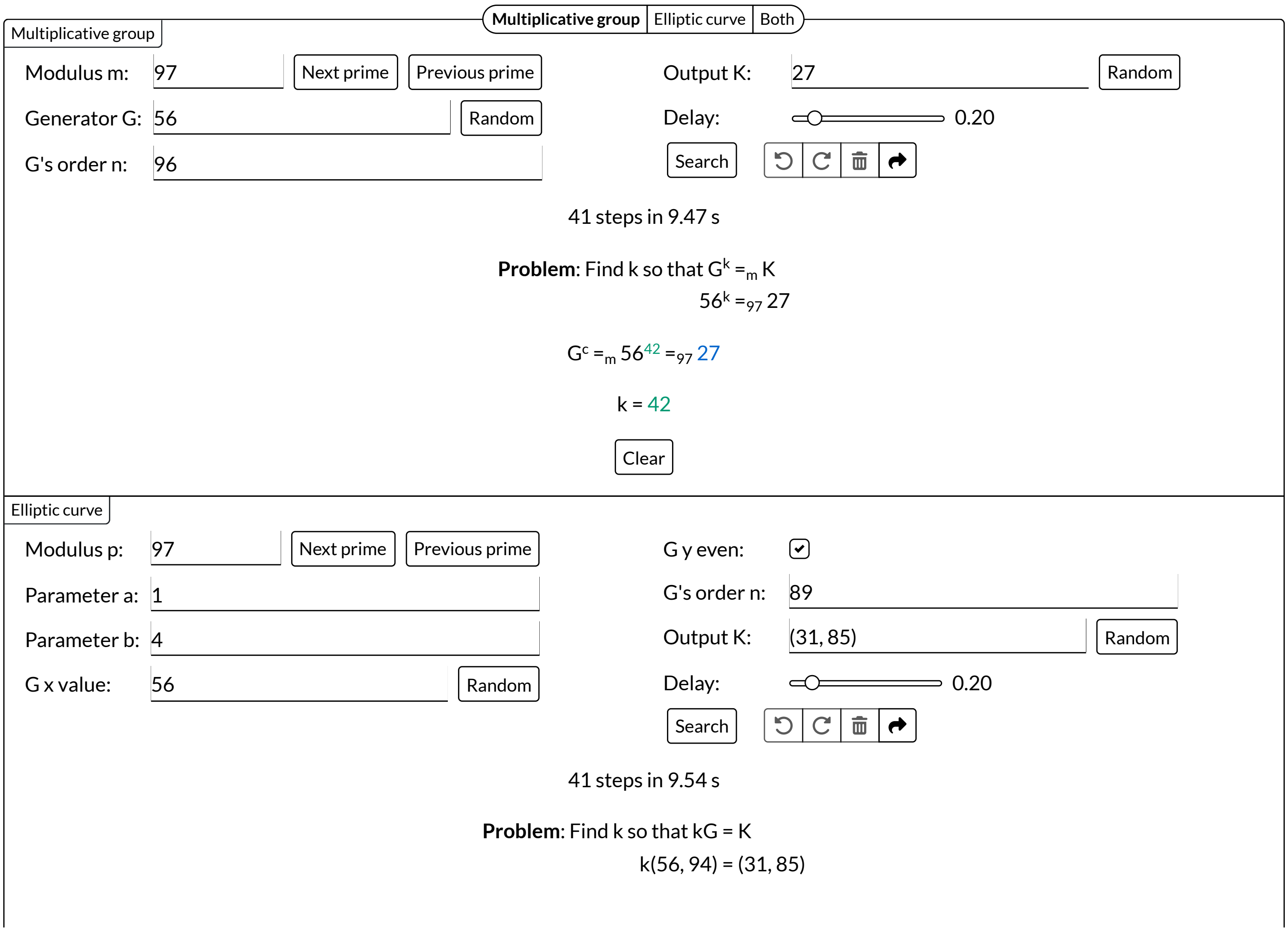
You simply count how many times you have to repeat the generator G to reach the output K.

Additive



You simply count how many times you have to repeat the generator G to reach the output K.

Please note that the above graphic is linear only for the coefficient/exponent of G. If you ordered the elements on the line according to their value (such as the x-coordinate in the case of elliptic curve points), the steps in blue would look completely chaotic (see the repetition tables of multiplicative groups and elliptic curves for actual sequences). The following tool implements exhaustive search for multiplicative groups and elliptic curves. The tool spends most of its time waiting so that you can actually see what's happening. If you want the result as quickly as possible, set the delay to 0, in which case the tool updates its output only every 1 million steps.



Multiplicative group | Elliptic curve | Both

Multiplicative group

Modulus m:	97	Output K:	27
Generator G:	56	Delay:	0.20
G's order n:	96		

41 steps in 9.47 s

Problem: Find k so that $G^k =_m K$

$56^k =_{97} 27$

$G^c =_m 56^{42} =_{97} 27$

k = 42

Elliptic curve

Modulus p:	97	G y even:	✓
Parameter a:	1	G's order n:	89
Parameter b:	4	Output K:	(31, 85)
G x value:	56	Delay:	0.20

41 steps in 9.54 s

Problem: Find k so that kG = K

k(56, 94) = (31, 85)

cG = 42(56, 94) = (31, 85)

k = 42

Clear

▼ Output not in generated subgroup

The tools in this chapter don't require that G generates the whole group. In the case of multiplicative groups, it can be desirable for G to generate just a subgroup so that its order can be prime (as mentioned earlier) and that the size of cryptographic keys and signatures can be smaller (as explained later). Moreover, some cryptosystems, such as RSA, use non-cyclic groups, which have no generators. If G doesn't generate the whole group, you can set the output K to an element which isn't in the subgroup generated by G. While subgroup membership can be tested easily in cyclic groups, I didn't implement this check because there is no simple formula to determine whether an elliptic curve is cyclic and I had to handle failures anyway for non-cyclic groups.

▼ Subgroup membership test in cyclic groups

Multiplicative | Additive | Both

Multiplicative

Given two elements G and K of a cyclic group, there exists an integer k so that $G^k = K$ if and only if $K^{|G|} = I$. Proof:

- If $G^k = K$, $K \in \langle G \rangle$. According to Lagrange's theorem, $|K|$ divides $|G|$. Therefore, $K^{|G|} = I$.
- If $K^{|G|} = I$, $|K|$ divides $|G|$. Now, K and $G^{|G|/|K|}$ generate a subgroup of order $|K|$. As proven earlier, a cyclic group has a single subgroup of order $|K|$. Thus, $\langle K \rangle = \langle G^{|G|/|K|} \rangle$. Since $K \in \langle G^{|G|/|K|} \rangle$, there exists an integer k so that $G^k = K$.

Additive

Given two elements G and K of a cyclic group, there exists an integer k so that $kG = K$ if and only if $(|G|)K = O$. Proof:

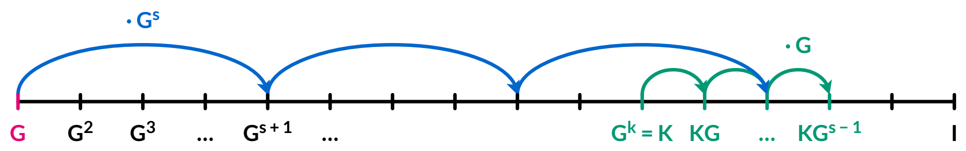
- If $kG = K$, $K \in \langle G \rangle$. According to Lagrange's theorem, $|K|$ divides $|G|$. Therefore, $(|G|)K = O$.
- If $(|G|)K = O$, $|K|$ divides $|G|$. Now, K and $\frac{|G|}{|K|}G$ generate a subgroup of order $|K|$. As proven earlier, a cyclic group has a single subgroup of order $|K|$. Thus, $\langle K \rangle = \langle \frac{|G|}{|K|}G \rangle$. Since $K \in \langle \frac{|G|}{|K|}G \rangle$, there exists an integer k so that $kG = K$.

Baby-step giant-step

Exhaustive search is very slow because it takes steps of size 1. If we take larger steps of size s by repeatedly multiplying the current element by G^s instead of G (or by adding sG instead of G), we jump over K if $k - 1$ is not a multiple of s. Instead of checking after each step whether we have reached K, we can compute $s - 1$ neighbors of K and check whether we have reached one of them:

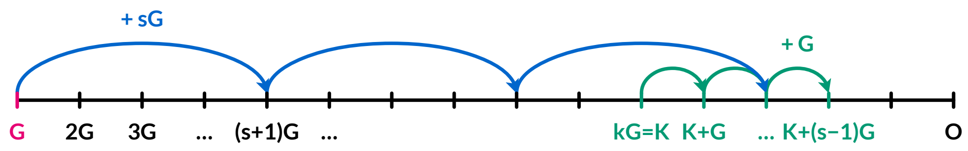
Multiplicative | Additive | Both

Multiplicative



If the landing area consists of s elements, we can take steps of size s without missing it.

Additive

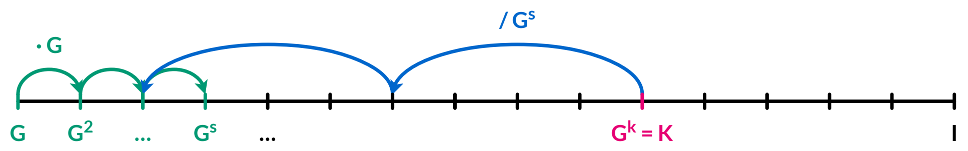


If the landing area consists of s elements, we can take steps of size s without missing it.

Since you make giant steps (in blue) after a series of baby steps (in green), this algorithm is known as baby-step giant-step. As you may want to compute the discrete logarithm of several elements, you typically compute the neighbors of G instead of K and then take the giant steps backwards from the current element K so that you can reuse the computed neighbors in later runs:

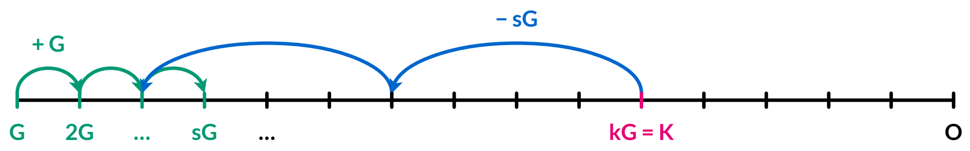
Multiplicative | Additive | Both

Multiplicative



In the more common variant of the algorithm, you take the giant steps backwards instead of forwards.

Additive



In the more common variant of the algorithm, you take the giant steps backwards instead of forwards.

To look up whether you have reached one of the green elements in constant time, you store these elements with their index in a hash table. If you have enough storage for $\sqrt{n}$ elements ($n = |G|$), it takes $\sqrt{n}$ baby steps to compute them and at most $\sqrt{n}$ giant steps to reach one of them from K. This is a massive improvement over exhaustive search, which takes in the order of n steps to find k, assuming that k was chosen randomly between 0 and n. The following tool implements the baby-step giant-step algorithm, but it makes at most 99 baby steps for the sake of visualization. If you're just interested in the result, you can use the next algorithm.

Multiplicative group | Elliptic curve | Both

Multiplicative group

Modulus m:	97	Output K:	27
Generator G:	56	Delay:	0.20
G's order n:	96		

14 steps in 3.95 s

Problem: Find k so that $G^k =_m K$

$56^k =_{97} 27$

The neighbors of G as our landing area:

$G^1 =_m 56$ $G^2 =_m 32$ $G^3 =_m 46$ $G^4 =_m 54$ $G^5 =_m 17$ $G^6 =_m 79$ $G^7 =_m 59$ $G^8 =_m 6$ $G^9 =_m 45$ $G^{10} =_m 95$

$K / G^c =_m 27 / 56^{40} =_{97} 32$

$k = c + i = 40 + 2 = 42$

Clear

Elliptic curve

Modulus p:	97	G y even:	✓
Parameter a:	1	G's order n:	89
Parameter b:	4	Output K:	(31, 85)
G x value:	56	Delay:	0.20

13 steps in 3.75 s

Problem: Find k so that kG = K

k(56, 94) = (31, 85)

The neighbors of G as our landing area:

1G = (56, 94)	3G = (44, 58)	5G = (84, 5)	7G = (29, 50)	9G = (40, 11)
2G = (82, 3)	4G = (6, 56)	6G = (62, 29)	8G = (74, 6)	

K − cG = (31, 85) − 36(56, 94) = (62, 29) k = c + i = 36 + 6 = 42

Clear

Pollard's rho algorithm

In order to achieve an expected running time of $\sqrt{n}$ with baby-step giant-step, you need to store $\sqrt{n}$ elements, which requires an impossible amount of memory for practical values of n. For example, if n is a 256-bit number, you'd have to store $2^{128} \cdot 256$ bits ≈ 10^{40} bytes in the case of elliptic curves, ignoring the overhead for the indexes and the data structure. For comparison, the global data storage capacity is around 10^{22} bytes in 2022. Pollard's rho algorithm, which is named after John M. Pollard (born in 1941), achieves the same expected running time of $\sqrt{n}$ while storing only 2 elements and 4 indexes by exploiting the birthday paradox.

Pollard's rho algorithm uses the following, non-cryptographic "hash" function, which maps the integers a and b to an element C:

Multiplicative | Additive | Both

Multiplicative

$$h(a, b) = G^a \cdot K^b = C$$

Additive

$$h(a, b) = aG + bK = C$$

Once we find a collision, i.e. inputs (a_1, b_1) and (a_2, b_2) where $b_1 \neq_n b_2$ so that $h(a_1, b_1) = h(a_2, b_2)$, we can solve for k as follows:

Multiplicative | Additive | Both

Multiplicative

$$\begin{aligned} G^{a_1} \cdot K^{b_1} &= G^{a_2} \cdot K^{b_2} \\ K^{b_1}/K^{b_2} &= G^{a_2}/G^{a_1} \\ (G^k)^{b_1 - b_2} &= G^{a_2 - a_1} \\ (b_1 - b_2)k &=_n a_2 - a_1 \end{aligned}$$

Additive

$$\begin{aligned} a_1G + b_1K &= a_2G + b_2K \\ b_1K - b_2K &= a_2G - a_1G \\ (b_1 - b_2)kG &= (a_2 - a_1)G \\ (b_1 - b_2)k &=_n a_2 - a_1 \end{aligned}$$

If $b_1 - b_2$ is coprime with G's order n, it has a multiplicative inverse, and thus $k =_n (a_2 - a_1) \cdot (b_1 - b_2)^{-1}$. As we'll see in the next section, we typically run Pollard's rho algorithm only if n is prime, in which case $b_1 - b_2 \neq_n 0$ is guaranteed to be coprime with n. If $b_1 - b_2$ is not coprime with n, there are several solutions for k, and we have to test one after the other in order to find the discrete logarithm of K. Depending on the prime factorization of n, there can be quite a lot of solutions to check, unfortunately (see below).

We can use insights from graph theory to find a collision. First, we define a sequence of elements S_i with the following function:

Multiplicative | Additive | Both

Multiplicative

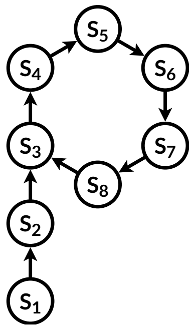
$$S_{i+1} = f(S_i) = \begin{cases} S_i \cdot G & \text{if } S_i =_3 0, \\ S_i \cdot K & \text{if } S_i =_3 1, \\ S_i \cdot S_i & \text{if } S_i =_3 2. \end{cases}$$

Additive

$$S_{i+1} = f(S_i) = \begin{cases} S_i + G & \text{if } (S_i)_x =_3 0, \\ S_i + K & \text{if } (S_i)_x =_3 1, \\ S_i + S_i & \text{if } (S_i)_x =_3 2. \end{cases}$$

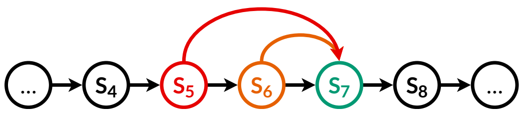
In the case of elliptic curves, we consider just the x-coordinate of each point when partitioning the elements into three sets.

If K is in the subgroup generated by G and we start with the identity element as S_1, all the elements S_i of the sequence are in the subgroup generated by G. Since we're interested only in finite groups with finitely many elements, the sequence has to reach an element which was already encountered earlier at some point. Since the next element of the sequence S_{i+1} depends only on the previous element S_i, the sequence repeats from there. Such a sequence thus forms the Greek letter ρ (rho) with a tail and a cycle:



The sequence repeats at $S_9 = S_3$.

In general, the tail can be empty, and the cycle can consist of a single element, which the function f maps to itself. Now, we can use Floyd's cycle-finding algorithm, named after Robert W. Floyd (1936 – 2001), to find two elements S_u and S_v in this sequence so that $S_u = S_v$ and $u \neq v$. The algorithm moves two elements with different speeds along the sequence. Alluding to one of Aesop's fables, these elements are usually called the tortoise and the hare. They both start at S_1, but whenever the tortoise makes one step, the hare makes two. As a consequence, the hare is ahead until the tortoise enters the cycle as well. Since the hare cannot jump over the tortoise as the distance between them decreases by one in each iteration, the hare catches up to the tortoise until they meet again:



If the hare (in red) is one element behind the tortoise (in orange), they'll be at the same element (in green) after the next iteration.

Once the tortoise and the hare meet again, we have the collision that we were looking for. In order to determine k, though, we must represent each element of the sequence as $h(a, b)$ for some integers a and b. The sequence function f has been chosen such that it's easy to keep track of a and b: After starting with $S_1 = h(a, b) = G^a \cdot K^b$ (or $S_1 = aG + bK$) for some initial values of a and b, you add 1 to a if $S_i =_3 0$, add 1 to b if $S_i =_3 1$, and double both a and b if $S_i =_3 2$. Using the subscript 1 to denote the values of the tortoise and the subcript 2 to denote the values of the hare, it's very unlikely that $b_1 =_n b_2$ because the hare travelled further than the tortoise. In the rare case that $b_1 =_n b_2$, you can start over with different initial values for a and b. I implemented this only for the Pohlig-Hellman algorithm in the next section. In order to have a deterministic outcome, the tool below sets a and b to 0 initially.

The following tool implements Pollard's rho algorithm. It visualizes the chase of the tortoise and the hare if G's order $n \leq 400$ and there is a delay. Since (without the visualization) you have to store only the current element C with the corresponding integers a and b for both the tortoise $\square_1$ and the hare $\square_2$, the algorithm requires only a very small and fixed amount of memory. As we'll see in the second box, it takes on average only around $\sqrt{n}$ iterations until $C_1 = C_2$. Therefore, Pollard's rho algorithm has the same expected running time as baby-step giant-step and is usually preferable to the latter due to its minimal memory consumption.

Multiplicative group | Elliptic curve | Both

Multiplicative group

Modulus m:	97 Next prime Previous prime	Output K:	27 Random
Generator G:	56 Random	Delay:	0.20
G's order n:	96		Search

25 steps in 6.11 s

Problem: Find k so that $G^k =_m K$

$56^k =_{97} 27$

G's order n: $96 = 2^5 \cdot 3$

sqrt(96) ≈ 9

The sequence of elements:

$S_1 =_m 1$
$S_2 =_m 27$
$S_3 =_m 57$
$S_4 =_m 88$
$S_5 =_m 48$
$S_6 =_m 69$
$S_7 =_m 81$

$S_8 =_m 74$
$S_9 =_m 44$
$S_{10} =_m 93$
$S_{11} =_m 67$
$S_{12} =_m 63$
$S_{13} =_m 36$
$S_{14} =_m 76$
$S_{15} =_m 15$
$S_{16} =_m 64$
$S_{17} =_m 79$
$S_{18} =_m 96$
$S_{19} =_m 41$
$S_{20} =_m 32$
$S_{21} =_m 54$
$S_{22} =_m 17$
$S_{23} =_m 95$
The start of the cycle: $S_{24} =_m 4$
$S_{25} =_m 11$
$S_{26} =_m 24 =_{97} 56^{40} \cdot 27^{2} =_{97} 56^{40} \cdot 27^{18}$
$S_{27} =_m 83$
$S_{28} =_m 2$

$$C_1 =_m G^{a_1} \cdot K^{b_1} =_m 56^{40} \cdot 27^{2} =_{97} 24$$
$$C_2 =_m G^{a_2} \cdot K^{b_2} =_m 56^{40} \cdot 27^{18} =_{97} 24$$

$$(b_1 - b_2) \cdot k =_n a_2 - a_1$$
$$(2 - 18) \cdot k =_{96} 40 - 40$$
$$80 \cdot k =_{96} 0$$

$$k \in \{0, 6, 12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 72, 78, 84, 90\}$$

$56^{0} =_{97} 1$
$56^{6} =_{97} 79$
$56^{12} =_{97} 33$
$56^{18} =_{97} 85$
$56^{24} =_{97} 22$
$56^{30} =_{97} 89$
$56^{36} =_{97} 47$
$56^{42} =_{97} 27$
$56^{48} =_{97} 96$
$56^{54} =_{97} 18$
$56^{60} =_{97} 64$
$56^{66} =_{97} 12$
$56^{72} =_{97} 75$
$56^{78} =_{97} 8$
$56^{84} =_{97} 50$
$56^{90} =_{97} 70$

$k = 42$

Clear

Elliptic curve

Modulus p:	97 Next prime Previous prime	G y even:	✓
Parameter a:	1	G's order n:	89
Parameter b:	4	Output K:	(31, 85) Random
G x value:	56 Random	Delay:	0.20
		Search	↺ ↻ 🗑 ➦

18 steps in 4.73 s

Problem: Find k so that kG = K
k(56, 94) = (31, 85)

G's order n: 89 is prime
sqrt(89) ≈ 9

The sequence of elements:

S_1 = O
S_2 = (56, 94)
S_3 = (82, 3)
S_4 = (20, 78)
S_5 = (56, 3)
S_6 = (82, 94)
S_7 = (28, 81)
S_8 = (29, 47)
S_9 = (61, 39)
S_{10} = (69, 25)
S_{11} = (26, 90)
S_{12} = (77, 61)
The start of the cycle: S_{13} = (38, 20)
S_{14} = (10, 74)
S_{15} = (29, 50)
S_{16} = (61, 58)
S_{17} = (32, 42)
S_{18} = (45, 24)
S_{19} = (87, 35) = 11(56, 94) + 66(31, 85) = 39(56, 94) + 6(31, 85)
S_{20} = (63, 57)
S_{21} = (0, 95)

$C_1 = a_1G + b_1K$ = 11(56, 94) + 66(31, 85) = (87, 35)
$C_2 = a_2G + b_2K$ = 39(56, 94) + 6(31, 85) = (87, 35)

$(b_1 - b_2) \cdot k =_n a_2 - a_1$
$(66 - 6) \cdot k =_{89} 39 - 11$
$60 \cdot k =_{89} 28$

k ∈ {42}

42(56, 94) = (31, 85)

k = 42

Clear

▼ Solving modular equations

How can we solve an equation of the form $a \cdot x =_m c$ for x? If a has a multiplicative inverse modulo m, $x =_m c \cdot a^{-1}$. If a has no multiplicative inverse, $a \cdot x =_m c$ has $d = \text{gcd}(a, m)$ solutions in $\mathbb{Z}_m$ if c is a multiple of d and no solution otherwise as we saw earlier. Using the extended Euclidean algorithm, we can find integers b and n so that $d = a \cdot b + m \cdot n$. After multiplying both sides by c/d, we get $c = a \cdot (b \cdot c/d) + m \cdot (n \cdot c/d)$. Modulo m, we have $a \cdot (b \cdot c/d) =_m c$, which means that $x =_m b \cdot c/d$ is a solution to $a \cdot x =_m c$. Since the d solutions are equally apart, we get the other $d - 1$ solutions by adding multiples of the offset $o = m/d$ to x. The following tool does all of that for you:

Integer a:	9	Modulus m:	15
Integer c:	12		

d = gcd(a, m) = b · a + n · m
d = gcd(9, 15) = 2 · 9 + (−1) · 15 = 3

$a \cdot x =_m c$

9 · x =$_{15}$ 12

x =$_m$ b · c / d
x =$_m$ 2 · 12 / 3 =$_{15}$ 8

o = m / d
o = 15 / 3 = 5

x ∈ {3, 8, 13}

▼ **Birthday paradox**

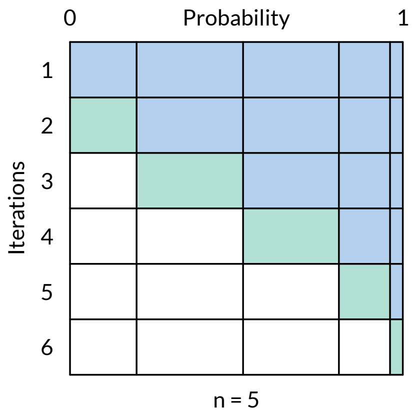
The birthday paradox denotes the counterintuitive fact that in a group of 23 people, it's likelier than not that at least two of them have the same birthday. Since we're not interested in birthdays, we analyze this problem more generally: When we put i items randomly into n buckets, how likely is it that at least one bucket contains at least two items? If $i > n$, the probability of this happening is 1 according to the pigeonhole principle. If $i \leq n$, we first analyze the probability $\overline{P}(i, n)$ that this is not happening, i.e. the probability that all buckets contain at most one item. The probability that the first item lands in an empty bucket is 1. For the second item, $n - 1$ of the n buckets are still empty. The probability of landing in one of those is $\frac{n-1}{n}$. Thus:

$$\overline{P}(i, n) = \frac{n}{n} \cdot \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \ldots \cdot \frac{n-i+1}{n} = \frac{n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-i+1)}{n^i} = \frac{n!}{n^i (n-i)!}$$

The probability that at least one bucket contains at least two items is then $P(i, n) = 1 - \overline{P}(i, n)$. Applied to the birthday paradox, we have $P(23, 365) = 1 - 365!/(365^{23}(365 - 23)!) =$ 0.5073. So how is this related to Pollard's rho algorithm? If the function $f(S_i)$ gives us a random element in the subgroup generated by G of order n, $P(i, n)$ describes how likely it is that f returned at least one element twice after i invocations. Unlike the birthday paradox, we're not interested in the median but rather in the expected number of iterations $E(n)$ until we obtain an element twice. The expected value of a random variable is determined as the sum of each possible outcome multiplied by the probability of this outcome. The first possible outcome is that we get a repetition in the second iteration. The probability that we get the element from the first iteration again in the second iteration is $\frac{1}{n}$. The probability that we get the first repetition in the third iteration is the probability that we don't get a repetition in the second iteration ($\frac{n-1}{n}$) times the probability that we get one of the first two elements in the third iteration ($\frac{2}{n}$). The probability that we sample n different elements and get the repetition only in the $(n + 1)$th iteration is $\frac{(n-1)!}{n^{n-1}} = \frac{n!}{n^n}$. Thus:

$$E(n) = 2 \cdot \frac{1}{n} + 3 \cdot \frac{n-1}{n} \cdot \frac{2}{n} + 4 \cdot \frac{n-1}{n} \cdot \frac{n-2}{n} \cdot \frac{3}{n} + \ldots + (n+1) \cdot \frac{n!}{n^n}$$
$$= \sum_{i=2}^{n+1} \frac{i \cdot (n-1)! \cdot (i-1)}{(n-i+1)! \cdot n^{i-1}} = \sum_{i=2}^{n+1} \frac{i \cdot (i-1)}{n-i+1} \cdot \frac{n!}{(n-i)! \cdot n^i}$$

This formula determines the colored area of the following outcome × probability table by adding up the area of each column:



The green area indicates the iteration in which the first repetition occurs.
The width of each column indicates how likely the particular outcome is.
As just explained, the width of each column is determined by $\frac{n!(i-1)}{(n-i+1)!n^i}$.

We can simplify $E(n)$ by summing over the rows instead of the columns. After the first iteration, the probability that we need at least one more iteration is 1. After the second iteration, the probability that we didn't get a repetition and thus need more iterations is $\frac{n-1}{n}$. After the third iteration, we have to multiply this probability times the probability that the third element wasn't one of the first two: $\frac{n-1}{n} \cdot \frac{n-2}{n}$. This gives us the following formula $Q(n)$ for the blue area in the above graphic:

$$Q(n) = 1 + \frac{n-1}{n} + \frac{n-1}{n} \cdot \frac{n-2}{n} + \ldots = \sum_{i=1}^{n} \frac{n!}{(n-i)! \cdot n^i} = \frac{n!}{n^n} \cdot \sum_{i=1}^{n} \frac{n^{n-i}}{(n-i)!}$$

By adding 1 for the green area, we have $E(n) = 1 + Q(n)$. To approximate the value of $Q(n)$, we need another formula $R(n)$:

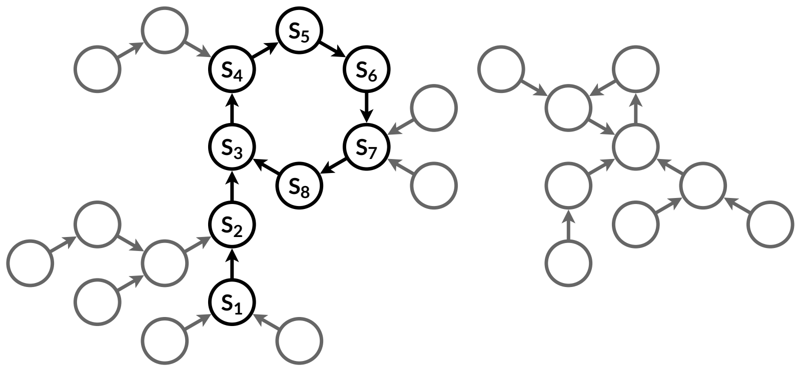
$$R(n) = 1 + \frac{n}{n+1} + \frac{n}{n+1} \cdot \frac{n}{n+2} + \ldots = \sum_{i=0}^{\infty} \frac{n! \cdot n^i}{(n+i)!} = \frac{n!}{n^n} \cdot \sum_{i=0}^{\infty} \frac{n^{n+i}}{(n+i)!}$$

For sufficiently large n, $Q(n) \approx R(n)$ because the difference between $\frac{n-1}{n}$ and $\frac{n}{n+1}$ gets smaller and smaller as n gets larger, and the addends from $i = n + 1$ to ∞ get vanishingly small as they are the product of lots of numbers smaller than 1. Using the Taylor series, named after Brook Taylor (1685 – 1731), of the exponential function (i.e. $e^n = \sum_{i=0}^{\infty} \frac{n^i}{i!}$) and Stirling's formula, named after James Stirling (1692 – 1770), to approximate factorials (i.e. $n! \approx \sqrt{2\pi n}\frac{n^n}{\mathrm{e}^n}$), we get:

$$Q(n) + R(n) = \frac{n!}{n^n} \cdot \Big(\sum_{i=1}^{n} \frac{n^{n-i}}{(n-i)!} + \sum_{i=0}^{\infty} \frac{n^{n+i}}{(n+i)!} \Big) = \frac{n!}{n^n} \cdot \Big(\sum_{i=0}^{\infty} \frac{n^i}{i!} \Big) = \frac{n!}{n^n} \cdot \mathrm{e}^n \approx \sqrt{2\pi n}$$

Therefore, $Q(n) \approx \frac{1}{2}\sqrt{2\pi n} = \sqrt{\frac{\pi}{2}n} \approx 1.25\sqrt{n}$ since $\frac{1}{2} = \sqrt{\frac{1}{4}}$, which means that it takes in the order of $\sqrt{n}$ iterations until you get a repetition when randomly sampling from n elements. The functions $Q(n)$ and $R(n)$ were introduced for this analysis by Donald Ervin Knuth (born in 1938) in section 1.2.11.3 in the first volume of his book The Art of Computer Programming.

While the function $f(S_i)$ used by Pollard's rho algorithm isn't random, it is random enough for $\sqrt{n}$ to be a really good estimate for how many steps are needed. In order to make it easier for you to judge this, the tool above displays the number of steps that it took to find a repetition at the top and the square root (sqrt) of G's order n below the problem statement. The above graphic depicting ρ showed only the elements in the sequence defined by the starting element and $f(S_i)$. However, $f(S_i)$ maps any of the n elements to a somewhat random other element. No matter where you start, you get a first repetition and thus a cycle after around $1.25\sqrt{n}$ steps on average. The complete picture looks like this:



The n elements are depicted as circles and the mappings defined by $f(S_i)$ as arrows.

▼ Pollard's rho factorization algorithm

We can apply the same idea to integer factorization. Confusingly, this algorithm is also called Pollard's rho algorithm. Given a composite number n and a divisor d of n, we can use the function $f(x) =_d x^2 + 1$ to define a sequence of integers s_i, where $s_{i+1} = f(s_i)$. Since we compute $f(x)$ modulo d, this function evaluates to at most d different values. If the sequence generated by $f(x)$ is random enough, it revisits an already visited integer after around $\sqrt{d}$ iterations on average as we saw in the previous box. We can use Floyd's cycle-finding algorithm to find two integers a and b so that $a =_d b$. If $a - b$ is a multiple of d, then so is $\gcd(n, |a - b|)$ because d is a factor of n. Now the problem is that we don't know d, otherwise we would have already found a factorization of n. Instead of computing $f(x)$ modulo d and checking whether $a =_d b$, we can compute $x^2 + 1$ modulo n and check whether $\gcd(n, |a - b|) > 1$. Since we don't have to reduce $f(x)$ modulo any integer and the former condition still implies the latter, the latter algorithm takes at most as many iterations of Floyd's cycle-finding algorithm as the former. If $\gcd(n, |a - b|) > 1$ and $a \neq_n b$, $\gcd(n, |a - b|)$ is a non-trivial divisor of n. Since this algorithm no longer depends on d and the above reasoning applies to all divisors of n, the number of iterations it takes to find a factor of n is expected to be around the square root of the smallest prime factor of n. We can implement Pollard's rho algorithm in pseudocode as follows:

```
let $o := 1$
function $f(x, n)$ {
  return $(x^2 + o)$ % $n$
}

function $factorize(n)$ {
    let $a := 2$
    let $b := 2$
    let $d := 1$
    while $(d = 1)$ {
        $a := f(a, n)$
        $b := f(f(b, n), n)$
        $d := \gcd(n, |a - b|)$
    }
    if $(d = n)$ {
        return “failure”
    } else {
        return $d$
    }
}
```

Usually, this algorithm finds the smallest prime factors of n first, but the returned d can be any divisor of n and doesn't have to be prime. If you want to find the prime factorization of n, you recursively factorize d and $\frac{n}{d}$ until all factors are prime, which you can test efficiently with a probabilistic primality test, such as the Miller-Rabin primality test. If $a =_n b$, $\gcd(n, |a - b|) = n$, and the algorithm fails. In this case, you can increment the offset o in the function $f(x)$ and try again. If the input $n = 4$, neither incrementing o nor starting from a different value makes the algorithm succeed. For this reason, it's common to halve the input n until it is odd and then to run Pollard's rho factorization algorithm from there.

Pollard's rho factorization algorithm can be optimized in the following two ways:

- Use Brent's cycle-finding algorithm, named after Richard Peirce Brent (born in 1946): Instead of moving both the tortoise and the hare along the sequence in each iteration, you move only the hare one step ahead, replacing three evaluations of $f(x)$ with one. After every power of 2 steps, you set the tortoise to the current value of the hare. The tortoise then acts as a waypoint so that it can stop the hare when it passes by. While Brent's cycle-finding algorithm evaluates $f(x)$ as often as Floyd's one does in the worst case, Brent showed that his algorithm requires 36% fewer evaluations of $f(x)$ on average.
- Combine several differences before running the Euclidean algorithm: Instead of computing the greatest common divisor in every iteration, you compute the product of the differences $|a - b|$ modulo n over several iterations and run the Euclidean algorithm only once in a while. This works because if $\gcd(n, x) > 1$, then $\gcd(n, x \cdot y) > 1$ for any integers x and y.

The following tool implements Pollard's rho factorization algorithm. Unlike my implementation of trial division above, it checks whether the remaining factor is prime. Its running time is thus in the order of the square root of the second largest prime factor if the exponent of the largest prime factor is 1. (It's possible to test for perfect powers efficiently as explained in the note 3.6 in the Handbook of Applied Cryptography and on Wikipedia; I just didn't implement this because you rarely encounter them.)

Integer: 231 Totients: Delay: 0.20 Factorize

4 steps in 2.48 s

231 = 3 · 7 · 11

Input	Greatest common divisor	Steps	Sqrt	+
231	gcd(231, |5 − 26|) = 21	1	4	1
21	gcd(21, |17 − 11|) = 3	3	2	2
3	is prime			
7	is prime			
11	is prime			

Clear

Pohlig-Hellman algorithm

The Pohlig-Hellman algorithm, which is named after Stephen Pohlig (1953 – 2017) and Martin Edward Hellman (born in 1945), reduces the discrete-logarithm problem in a large group of non-prime order to several discrete-logarithm problems in smaller groups, which are easier to solve since the running time of the above DL algorithms depend only on the size of the group. Given a generator G, its order n, and an output K of the linear one-way function, the Pohlig-Hellman algorithm finds the input k as follows:

Multiplicative | Additive | Both

Multiplicative

1. Find the prime factorization of n so that $n = p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}$ for distinct primes p_i and integers $e_i \geq 1$. Since the Pohlig-Hellman decomposition succeeds only if all the prime factors are sufficiently small, Pollard's rho factorization algorithm is a good choice for this step. If the factorization fails because some factors are too large, the steps below wouldn't work either.
2. For i from 1 to l, do the following:
   1. Compute $G_i := G^{n/p_i^{e_i}}$. Since the order of G is n, the order of G_i is $p_i^{e_i}$.
   2. Compute $K_i := K^{n/p_i^{e_i}}$. Since $G^k = K$, we have that $K_i = (G^k)^{n/p_i^{e_i}} = (G^{n/p_i^{e_i}})^k = G_i^k$, which means that the discrete logarithm of K to the base G is also a discrete logarithm of K_i to the base G_i. As we saw earlier, a discrete logarithm is unique only up to the order of its base. Therefore, the discrete logarithm of K_i to the base G_i, which I denote as k_i, is not necessarily a discrete logarithm of K to the base G as $k_i \in \{0, \ldots, p_i^{e_i} - 1\}$, whereas $k \in \{0, \ldots, n - 1\}$. A discrete logarithm is unique up to the order of its base, though, which implies that $k =_{p_i^{e_i}} k_i$.

      Instead of determining k so that $G^k = K$ in the original group of order n, we can thus determine k_i so that $G_i^{k_i} = K_i$ in the subgroup generated by G_i of order $p_i^{e_i}$, which improves the expected running time from $\sqrt{n}$ to $\sqrt{p_i^{e_i}}$ when using Pollard's rho algorithm for this. Depending on the prime factorization of n, this can already be a big improvement. The following insight allows us to determine k_i in the steps 2.3 to 2.5 below without ever having to compute a discrete logarithm in a group larger than p_i, which improves the expected running time even further to $\sqrt{p_i}$. If you're not interested in this, you can continue with the step 3 at the bottom of this box.

      Insight: We can write k_i in base p_i as $k_i = d_0 + d_1p_i + d_2p_i^2 + \ldots + d_{e_i-1}p_i^{e_i-1} = \sum_{j=0}^{e_i-1} d_jp_i^j$, where each digit $d_j \in \{0, \ldots, p_i - 1\}$. Now, $G_i^{k_i} = G_i^{d_0+d_1p_i+\ldots+d_{e_i-1}p_i^{e_i-1}} = G_i^{d_0} \cdot G_i^{d_1p_i} \cdot \ldots \cdot G_i^{d_{e_i-1}p_i^{e_i-1}} = K_i$. When we raise both sides to the power of $p_i^{e_i-1}$, we get $(G_i^{d_0})^{p_i^{e_i-1}} = (G_i^{p_i^{e_i-1}})^{d_0} = K_i^{p_i^{e_i-1}}$ because all the other factors vanish as $(G_i^{d_jp_i^{1+?}})^{p_i^{e_i-1}} = (G_i^{p_i^{e_i+?}})^{d_j} = I^{d_j} = I$ due to Lagrange's theorem. Once we have determined d_0 as the discrete logarithm of $K_i^{p_i^{e_i-1}}$ to the base $G_i^{p_i^{e_i-1}}$ of order p_i with one of the above DL algorithms, we can take $G_i^{d_0}$ to the other side of the equation in red: $G_i^{d_1p_i} \cdot \ldots \cdot G_i^{d_{e_i-1}p_i^{e_i-1}} = K_i/G_i^{d_0}$. When we raise both sides to the power of $p_i^{e_i-2}$, we get $(G_i^{d_1p_i})^{p_i^{e_i-2}} = (G_i^{p_i^{e_i-1}})^{d_1} = (K_i/G_i^{d_0})^{p_i^{e_i-2}}$ because all the other factors vanish again. After determining d_1 in the same group of order p_i as before, we move $G_i^{d_1p_i}$ also to the other side of the equation, which gives us $G_i^{d_2p_i^2} \cdot \ldots \cdot G_i^{d_{e_i-1}p_i^{e_i-1}} = K_i/G_i^{d_0}/G_i^{d_1p_i} = K_i/G_i^{d_0+d_1p_i}$. By continuing like this, we can determine all the digits d_j of k_i without ever having to solve a discrete-logarithm problem in a group larger than p_i. We implement this as follows:
   3. Compute $H_i := G_i^{p_i^{e_i-1}}$. Since the order of G_i is $p_i^{e_i}$, the order of H_i is p_i. I colored the term $G_i^{p_i^{e_i-1}}$ in the explanation above so that it's easier for you to see that each digit d_j is determined in the group generated by H_i.
   4. For j from 0 to $e_i - 1$, do the following:
      1. If $j = 0$, set $s_j := 0$. Otherwise, set $s_j := s_{j-1} + d_{j-1} \cdot p_i^{j-1}$. s_j is the sum of all the digits that we have found so far, where each digit is multiplied by the value of its position. I colored this partial sum in the explanation above so that it's easier for you to see where it's being used.
      2. Compute $D_j := (K_i/G_i^{s_j})^{p_i^{e_i-1-j}}$. I also colored the occurrences of this expression in the explanation above.
      3. Find d_j so that $H_i^{d_j} = D_j$ with one of the above DL algorithms.
   5. Compute $k_i := d_0 + d_1p_i + d_2p_i^2 + \ldots + d_{e_i-1}p_i^{e_i-1} = \sum_{j=0}^{e_i-1} d_jp_i^j$. (Alternatively, set $k_i := s_{e_i-1} + d_{e_i-1}p_i^{e_i-1}$.)
3. Use the Chinese remainder theorem to solve the system of congruences $k =_{p_i^{e_i}} k_i$ efficiently.
4. Return k as the solution to $G^k = K$.

Additive

1. Find the prime factorization of n so that $n = p_1^{e_1} \cdot \ldots \cdot p_l^{e_l}$ for distinct primes p_i and integers $e_i \geq 1$. Since the Pohlig-Hellman decomposition succeeds only if all the prime factors are sufficiently small, Pollard's rho factorization algorithm is a good choice for this step. If the factorization fails because some factors are too large, the steps below wouldn't work either.
2. For i from 1 to l, do the following:
   1. Compute $G_i := (n/p_i^{e_i})G$. Since the order of G is n, the order of G_i is $p_i^{e_i}$.
   2. Compute $K_i := (n/p_i^{e_i})K$. Since $kG = K$, we have that $K_i = (n/p_i^{e_i})kG = k(n/p_i^{e_i})G = kG_i$, which means that the discrete logarithm of K to the base G is also a discrete logarithm of K_i to the base G_i. As we saw earlier, a discrete logarithm is unique only up to the order of its base. Therefore, the discrete logarithm of K_i to the base G_i, which I denote as k_i, is not necessarily a discrete logarithm of K to the base G as $k_i \in \{0, \ldots, p_i^{e_i} - 1\}$, whereas $k \in \{0, \ldots, n - 1\}$. A discrete logarithm is unique up to the order of its base, though, which implies that $k =_{p_i^{e_i}} k_i$.

      Instead of determining k so that $kG = K$ in the original group of order n, we can thus determine k_i so that $k_iG_i = K_i$ in the subgroup generated by G_i of order $p_i^{e_i}$, which improves the expected running time from $\sqrt{n}$ to $\sqrt{p_i^{e_i}}$ when using Pollard's rho algorithm for this. Depending on the prime factorization of n, this can already be a big improvement. The following insight allows us to determine k_i in the steps 2.3 to 2.5 below without ever having to compute a discrete logarithm in a group larger than p_i, which improves the expected running time even further to $\sqrt{p_i}$. If you're not interested in this, you can continue with the step 3 at the bottom of this box.

Insight: We can write k_i in base p_i as $k_i = d_0 + d_1p_i + d_2p_i^2 + \ldots + d_{e_i-1}p_i^{e_i-1} = \sum_{j=0}^{e_i-1} d_jp_i^j$, where each digit $d_j \in \{0, \ldots, p_i - 1\}$. Now, $k_iG_i = (d_0 + d_1p_i + \ldots + d_{e_i-1}p_i^{e_i-1})G_i = d_0G_i + d_1p_iG_i + \ldots + d_{e_i-1}p_i^{e_i-1}G_i = K_i$. When we multiply both sides by $p_i^{e_i-1}$, we get $p_i^{e_i-1}(d_0G_i) = d_0(p_i^{e_i-1}G_i) = p_i^{e_i-1}K_i$ because all the other terms vanish as $p_i^{e_i-1}(d_jp_i^{1+?}G_i) = d_j(p_i^{e_i+?}G_i) = d_jO = O$ due to Lagrange's theorem. Once we have determined d_0 as the discrete logarithm of $p_i^{e_i-1}K_i$ to the base $p_i^{e_i-1}G_i$ of order p_i with one of the above DL algorithms, we can take d_0G_i to the other side of the equation in red: $d_1p_iG_i + \ldots + d_{e_i-1}p_i^{e_i-1}G_i = K_i - d_0G_i$. When we multiply both sides by $p_i^{e_i-2}$, we get $p_i^{e_i-2}(d_1p_iG_i) = d_1(p_i^{e_i-1}G_i) = p_i^{e_i-2}(K_i - d_0G_i)$ because all the other terms vanish again. After determining d_1 in the same group of order p_i as before, we move $d_1p_iG_i$ also to the other side of the equation, which gives us $d_2p_i^2G_i + \ldots + d_{e_i-1}p_i^{e_i-1}G_i = K_i - d_0G_i - d_1p_iG_i = K_i - (d_0 + d_1p_i)G_i$. By continuing like this, we can determine all the digits d_j of k_i without ever having to solve a discrete-logarithm problem in a group larger than p_i. We implement this as follows:

3. Compute $H_i := p_i^{e_i-1}G_i$. Since the order of G_i is $p_i^{e_i}$, the order of H_i is p_i. I colored the term $p_i^{e_i-1}G_i$ in the explanation above so that it's easier for you to see that each digit d_j is determined in the group generated by H_i.
4. For j from 0 to $e_i - 1$, do the following:
    1. If $j = 0$, set $s_j := 0$. Otherwise, set $s_j := s_{j-1} + d_{j-1} \cdot p_i^{j-1}$. s_j is the sum of all the digits that we have found so far, where each digit is multiplied by the value of its position. I colored this partial sum in the explanation above so that it's easier for you to see where it's being used.
    2. Compute $D_j := p_i^{e_i-1-j}(K_i - s_jG_i)$. I also colored the occurrences of this expression in the explanation above.
    3. Find d_j so that $d_jH_i = D_j$ with one of the above DL algorithms.
5. Compute $k_i := d_0 + d_1p_i + d_2p_i^2 + \ldots + d_{e_i-1}p_i^{e_i-1} = \sum_{j=0}^{e_i-1} d_jp_i^j$. (Alternatively, set $k_i := s_{e_i-1} + d_{e_i-1}p_i^{e_i-1}$.)

3. Use the Chinese remainder theorem to solve the system of congruences $k =_{p_i^{e_i}} k_i$ efficiently.
4. Return k as the solution to $kG = K$.

The above algorithm shows that the difficulty of computing discrete logarithms is determined by the size of the largest prime factor p_l of the group's order n, ignoring even its exponent e_l. This means that if the multiplicative group or the elliptic curve isn't chosen carefully, it can be relatively easy to invert the linear one-way function. For example, the tool below computes discrete logarithms modulo the 107-digit/354-bit prime 22'708'823'198'678'103'974'314'518'195'029'102'158'525'052'496'759'285'596'453'269'189'798'311'427'475'159'776'411'276'642'277'139'650'833'937 in a matter of seconds because the largest prime factor of this prime minus one is just 350'377. (I took this example from the note 3.66 in the Handbook of Applied Cryptography.) On the other hand, if you choose a safe prime as the modulus of a multiplicative group, the Pohlig-Hellman algorithm reduces the difficulty of the group's discrete-logarithm problem only by a single bit. In the case of elliptic curves, it makes sense to choose the parameters such that the group contains a prime number of points to begin with.

The following tool implements the Pohlig-Hellman algorithm. Since Pollard's rho algorithm has a decent chance of total failure in small groups, the tool uses exhaustive search to find the discrete logarithm in groups smaller than 100 and Pollard's rho algorithm with a randomly chosen starting element in the case of failure otherwise. Unlike the other DL algorithms, you cannot configure a delay because the only aspects worth animating are the sub-algorithms that we've already covered. This tool simply links to the corresponding tool with the current values so that you can inspect the steps of the sub-algorithm there.

Multiplicative group | Elliptic curve | Both

Multiplicative group

Modulus m:	97 Next prime Previous prime	G's order n:	96
Generator G:	56 Random	Output K:	27 Random

Search

0 steps in 1.40 s

Problem: Find k so that $G^k =_m K$

$56^k =_{97} 27$

G's order n: $96 = 2^5 \cdot 3$

1. Subproblem: Find k_1 so that $G_1^{k_1} =_m K_1$, where

$p_1 = 2$

$e_1 = 5$

$K_1 =_m K^{n / p_1^{e_1}} =_{97} 89$

$G_1 =_m G^{n / p_1^{e_1}} =_{97} 46$

$H_1 =_m G_1^{p_1^{e_1 - 1}} =_{97} 96$

j	$s_j = s_{j-1} + d_{j-1} \cdot p_1^{j-1}$	$D_j =_m (K_1 / G_1^{s_j})^{p_1^{e_1-1-j}}$	d_j so that $H_1^{d_j} =_m D_j$	Steps
0	0	1	0	0

j	$s_j = s_{j-1} + d_{j-1} \cdot p_1^{j-1}$	$D_j =_m (K_1 / G_1^{s_j})^{p_1^{e_1-1-j}}$	d_j so that $H_1^{d_j} =_m D_j$	Steps
1	0	96	1	0
2	2	1	0	0
3	2	96	1	0
4	10	1	0	0

$k_1 = 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 = 10$

2. Subproblem: Find k_2 so that $G_2^{k_2} =_m K_2$, where

$p_2 = 3$

$e_2 = 1$

$K_2 =_m K^{n / p_2^{e_2}} =_{97} 1$

$G_2 =_m G^{n / p_2^{e_2}} =_{97} 35$

$H_2 =_m G_2^{p_2^{e_2-1}} =_{97} 35$

j	$s_j = s_{j-1} + d_{j-1} \cdot p_2^{j-1}$	$D_j =_m (K_2 / G_2^{s_j})^{p_2^{e_2-1-j}}$	d_j so that $H_2^{d_j} =_m D_j$	Steps
0	0	1	0	0

$k_2 = 0 \cdot 3^0 = 0$

Problem: $k =_{p_1^{e_1}} k_1 =_{32} 10$

$k =_{p_2^{e_2}} k_2 =_3 0$

Solution: $k =_{96} 42$

Clear

Elliptic curve

Modulus p:	97 Next prime Previous prime	G y even:	✓
Parameter a:	1	G's order n:	89
Parameter b:	4	Output K:	(31, 85) Random
G x value:	56 Random	Search	

41 steps in 1.26 s

Problem: Find k so that kG = K

k(56, 94) = (31, 85)

G's order n: 89 is prime

1. Subproblem: Find k_1 so that $k_1G_1 = K_1$, where

$p_1 = 89$

$e_1 = 1$

$K_1 = (n / p_1^{e_1})K = (31, 85)$

$G_1 = (n / p_1^{e_1})G = (56, 94)$

$H_1 = (p_1^{e_1-1})G_1 = (56, 94)$

j	$s_j = s_{j-1} + d_{j-1} \cdot p_1^{j-1}$	$D_j = p_1^{e_1-1-j}(K_1 - s_jG_1)$	d_j so that $d_jH_1 = D_j$	Steps
0	0	(31, 85)	42	41

$k_1 = 42 \cdot 89^0 = 42$

Problem: $k =_{p_1^{e_1}} k_1 =_{89} 42$

Solution: $k =_{89} 42$

Clear

▼ **Potential for backdoors in multiplicative groups**

Before you rely on a multiplicative group which was provided by someone else to be a linear one-way function, you have to verify that the group's order has a sufficiently large prime factor. If the group's order or its prime factorization is not known to you (because they aren't provided to you or you can't determine them), you shouldn't use the group. Otherwise, an attacker can choose the modulus m as the product of two large primes p and q, where both $p - 1$ and $q - 1$ have only relatively small prime factors. Since it's difficult to factorize such a modulus m, only the attacker can determine the group's order $\varphi(m) = (p - 1) \cdot (q - 1)$ and use the Pohlig-Hellman algorithm to compute discrete logarithms in this group. In order to avoid such a backdoor, you must insist that the other party shares the group's order and its prime factorization with you.

Index-calculus algorithm

The index-calculus algorithm computes discrete logarithms in multiplicative groups much faster than Pollard's rho algorithm. It exploits the circumstance that sufficiently many integers are sufficiently smooth, i.e. their prime factorization contains no factors larger than some integer S. Let P_1, …, P_l denote all the prime numbers smaller than S. An S-smooth number A can then be written as $A = P_1^{e_1} \cdot \ldots \cdot P_l^{e_l}$, where $e_i = 0$ if P_i is not a prime factor of A for each $i \in \{1, \ldots, l\}$. If we assume for now that the modulus m is prime and greater than S, and that G generates the whole group $\mathbb{Z}_m^\times$, there exists an integer p_i for each P_i so that $G^{p_i} =_m P_i$. To determine these integers, we repeatedly choose a random integer $r \in \{1, \ldots n\}$, where $n = |G|$, until $G^r =_m R$ is S-smooth:

$$G^r =_m R = P_1^{e_1} \cdot \ldots \cdot P_l^{e_l} =_m (G^{p_1})^{e_1} \cdot \ldots \cdot (G^{p_1})^{e_l} = G^{e_1 \cdot p_1 + \ldots + e_l \cdot p_l}$$

After taking the logarithm to base G on both ends, we get $r =_n e_1 \cdot p_1 + \ldots + e_l \cdot p_l$ in the repetition ring of G. If we find enough such equations, we can solve the following system of linear equations using Gaussian elimination, named after Carl Friedrich Gauss (1777 – 1855). (I'll explain matrix multiplication and Gaussian elimination in the article about coding theory.)

$$\begin{aligned} e_{1,1} \cdot p_1 + \ldots + e_{1,l} \cdot p_l &=_n r_1 \\ \vdots &=_n \vdots \\ e_{l,1} \cdot p_1 + \ldots + e_{l,l} \cdot p_l &=_n r_l \end{aligned} \quad \iff \quad \begin{bmatrix} e_{1,1} & \cdots & e_{1,l} \\ \vdots & & \vdots \\ e_{l,1} & \cdots & e_{l,l} \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ \vdots \\ p_l \end{bmatrix} =_n \begin{bmatrix} r_1 \\ \vdots \\ r_l \end{bmatrix}$$

Since not all rows are linearly independent from one another and not all elements are invertible in the ring of integers modulo n, we usually need more than just l equations to solve for the integers p_1 to p_l. Once we have found these values, we increment a counter c starting from 0 until $K \cdot G^c =_m T$ is S-smooth. Remember: We want to find an integer k so that $G^k =_m K$. Given $K \cdot G^c =_m T = P_1^{e_1} \cdot \ldots \cdot P_l^{e_l} =_m (G^{p_1})^{e_1} \cdot \ldots \cdot (G^{p_1})^{e_l} = G^{e_1 \cdot p_1 + \ldots + e_l \cdot p_l}$ for some new exponents e_1 to e_l, we can determine k as follows:

$$\begin{aligned} k + c &=_n e_1 \cdot p_1 + \ldots + e_l \cdot p_l \\ k &=_n e_1 \cdot p_1 + \ldots + e_l \cdot p_l - c \end{aligned}$$

If you want to compute several discrete logarithms in the same group, you have to determine the integers p_1 to p_l only once, which makes the index-calculus algorithm even faster. (The discrete logarithm of P_i with respect to G is also called the index of P_i with respect to G, hence the name index calculus.) The algorithm doesn't work for elliptic curves because points cannot be factorized. The running time of the index-calculus algorithm is $\mathrm{e}^{(\sqrt{2}+\underline{o}(1))\sqrt{\log_\mathrm{e}(m) \cdot \log_\mathrm{e}(\log_\mathrm{e}(m))}}$, which is subexponential and superpolynomial.

The following tool implements the index-calculus algorithm. In order to make its output reproducible, the tool increments r starting from 1 in its search for smooth group elements instead of choosing r randomly. It continues the search until the matrix of exponents becomes invertible modulo n. As long as you disable the delay, it handles 50-bit moduli just fine. For example, it takes only around 7'800'000 steps to compute discrete logarithms modulo the safe prime 627'789'652'071'083. In order to share its inputs with the DL algorithms above, the tool doesn't allow you to configure the number of prime bases yourself. It chooses $l = \min(\log_\mathrm{e}(n), 50)$.

Multiplicative group

Multiplicative group

Modulus m: 97 Next prime Previous prime

Generator G: 56 Random

G's order n: 96

Output K: 27 Random

Delay: 0.20

Search

12 steps in 3.90 s

Problem: Find k so that $\mathrm{G}^\mathrm{k} =_\mathrm{m} \mathrm{K}$

$56^\mathrm{k} =_{97} 27$

Modulus m: 97 is prime

G's order n: $96 = 2^5 \cdot 3$

5 prime bases: 2, 3, 5, 7, 11

6 equations:

$56^{1} =_{97} 56 = 2^{3} \cdot 7^{1}$

$56^{2} =_{97} 32 = 2^{5}$

$56^{4} =_{97} 54 = 2^{1} \cdot 3^{3}$

$56^{8} =_{97} 6 = 2^{1} \cdot 3^{1}$

$56^{9} =_{97} 45 = 3^{2} \cdot 5^{1}$

$56^{12} =_{97} 33 = 3^{1} \cdot 11^{1}$

$$\begin{bmatrix} 3 & 0 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 58 \\ 46 \\ 13 \\ 19 \\ 62 \end{bmatrix} =_{96} \begin{bmatrix} 1 \\ 2 \\ 4 \\ 8 \\ 9 \\ 12 \end{bmatrix}$$

$56^{58} =_{97} 2$

$56^{46} =_{97} 3$

$56^{13} =_{97} 5$

$56^{19} =_{97} 7$

$56^{62} =_{97} 11$

$\mathrm{K} \cdot \mathrm{G}^{c} =_{m} 27 \cdot 56^{0} =_{97} 27 = 3^{3}$

$\mathrm{k} =_{96} 3 \cdot 46 - 0 =_{96} 42$ ✓

Clear

Index calculus in subgroups

What if G generates only a proper subgroup of $\mathbb{Z}_m^\times$? Since the running time of the index-calculus algorithm depends on the modulus m rather than G's order n, Pollard's rho algorithm is faster than the index-calculus algorithm when n is small enough. If n has many small factors, the Pohlig-Hellman algorithm is even faster, of course. (Since the index-calculus algorithm is not the best choice for relatively small subgroups, it usually doesn't make sense to use it as the DL algorithm in the Pohlig-Hellman algorithm.) If n is neither small nor smooth, you can solve the discrete-logarithm problem $G^k =_m K$ as follows according to section 3.6.6 of the Handbook of Applied Cryptography (I don't divide h and g by the cofactor of the subgroup for simplicity):

1. Find an element $H \in \mathbb{Z}_m^\times$, which generates the whole group. (We still assume that m is prime, and thus $|\mathbb{Z}_m^\times| = m - 1$.)
2. Use the index-calculus algorithm to find an integer g so that $H^g =_m G$.
3. Use the index-calculus algorithm to find an integer h so that $H^h =_m K$.
4. Return $k :=_n h \cdot g^{-1}$.

This works because $(H^g)^k =_m H^h$ implies that $g \cdot k =_{m-1} h$. Since n divides $m - 1$, we have that $g \cdot k - h$ is also a multiple of n, and thus $g \cdot k =_n h$. Since k is unique up to a multiple of n, g has to be invertible modulo n. Therefore, $k =_n h \cdot g^{-1}$.

Interestingly, the above tool succeeds even if one of the bases P_i is not in the subgroup generated by G. For example, it solves discrete logarithms in the subgroup of order 32 modulo 96 just fine. I have no idea why the index-calculus algorithm still works in this case, and I haven't found any information about this phenomenon. If you know the answer, please let me know.

Index calculus with composite moduli

If the modulus m is composite, the above tool fails to invert the matrix of exponents. I understand that the prime factors of G's order n determine which exponents have a multiplicative inverse modulo n, which is why the above tool displays the prime factorization of n. However, n can have many small factors no matter whether m is composite or prime. If you find a composite modulus for which the tool succeeds for a subgroup larger than four, or if you know why this isn't possible, please let me know.

Bits of security

As we learned in this chapter, it takes only around $\sqrt{n}$ steps to compute discrete logarithms in groups of prime order n due to Pollard's rho algorithm. If the order n is composite, the Pohlig-Hellman algorithm reduces the difficulty of computing discrete logarithms to the square root of the largest prime factor of n. In the case of multiplicative groups, it takes even fewer steps due to the index-calculus algorithm. The bit-length of the expected number of steps required to break a cryptographic primitive defines its security level. When we say that a primitive has s bits of security, it takes on average around 2^s steps to break it. Given the current state of computers, cryptographic primitives become computationally intractable at around 100 bits of security. In order to ensure that our linear one-way functions are indeed one-way functions, we have to choose sufficiently large parameters as follows:

Bits of security	Elliptic curves: length of G's order n in bits	Multiplicative groups: length of modulus m in bits
80	160	1'024
112	224	2'048
128	256	3'072
192	384	7'680
256	512	15'360

Comparable security strengths as listed in section 5.6.1.1 starting on page 53 in this recommendation by the National Institute of Standards and Technology (NIST). The security level with the red background can no longer be considered secure.

As explained earlier, just the modulus of a multiplicative group has to be this large. The subgroup generated by G can be as "small" as an elliptic curve group of the same security level. As we will see in the next article about cryptosystems, the input k to the linear one-way function is usually a private key, while the output K is the corresponding public key. The advantages of elliptic curves over multiplicative groups are that their public keys are smaller and that their group operations are faster since the numbers are smaller.

Group properties (appendix)

Reduced group axioms

Using universal and existential quantifiers to make statements about the elements of a set, the group axioms can be reduced to:

Generic | Multiplicative | Additive | All

Generic

- **Closure (G1):** $\forall\ A, B \in \mathbb{G}\ A \circ B \in \mathbb{G}$
- **Associativity (G2):** $\forall\ A, B, C \in \mathbb{G}\ (A \circ B) \circ C = A \circ (B \circ C)$
- **Identity (G3):** $\exists\ E \in \mathbb{G}\ \forall\ A \in \mathbb{G}\ A \circ E = A$
- **Invertibility (G4):** $\forall\ A \in \mathbb{G}\ \exists\ B \in \mathbb{G}\ A \circ B = E$

Multiplicative

- **Closure (G1):** $\forall\ A, B \in \mathbb{G}\ A \cdot B \in \mathbb{G}$
- **Associativity (G2):** $\forall\ A, B, C \in \mathbb{G}\ (A \cdot B) \cdot C = A \cdot (B \cdot C)$
- **Identity (G3):** $\exists\ I \in \mathbb{G}\ \forall\ A \in \mathbb{G}\ A \cdot I = A$
- **Invertibility (G4):** $\forall\ A \in \mathbb{G}\ \exists\ B \in \mathbb{G}\ A \cdot B = I$

Additive

- **Closure (G1):** $\forall\ A, B \in \mathbb{G}\ A + B \in \mathbb{G}$
- **Associativity (G2):** $\forall\ A, B, C \in \mathbb{G}\ (A + B) + C = A + (B + C)$
- **Identity (G3):** $\exists\ O \in \mathbb{G}\ \forall\ A \in \mathbb{G}\ A + O = A$
- **Invertibility (G4):** $\forall\ A \in \mathbb{G}\ \exists\ B \in \mathbb{G}\ A + B = O$

Commutative groups have an additional axiom:

Generic | Multiplicative | Additive | All

Generic

Commutativity (G5): $\forall\ A, B \in \mathbb{G}\ A \circ B = B \circ A$

Multiplicative

Commutativity (G5): $\forall\, A, B \in \mathbb{G}\; A \cdot B = B \cdot A$

Additive

Commutativity (G5): $\forall\, A, B \in \mathbb{G}\; A + B = B + A$

▼ **Less ambiguous notation**

Since the third axiom (G3) does not specify how many identity elements there are, this notation is still a bit sloppy as it leaves the meaning of the identity element in the last axiom (G4) open. Must each element have an inverse for some identity element or for all identity elements? Keeping the quantifiers at the beginning of the statement and using the logical conjunction $\land$ ("and"), the group axioms can be written less ambiguously as:

Generic | Multiplicative | Additive | All

Generic

$$\exists\, E \in \mathbb{G}\, \forall\, A, B, C \in \mathbb{G}\, \exists\, D \in \mathbb{G}\; A \circ B \in \mathbb{G} \land (A \circ B) \circ C = A \circ (B \circ C) \land A \circ E = A \land A \circ D = E$$

Multiplicative

$$\exists\, I \in \mathbb{G}\, \forall\, A, B, C \in \mathbb{G}\, \exists\, D \in \mathbb{G} A \cdot B \in \mathbb{G} \land (A \cdot B) \cdot C = A \cdot (B \cdot C) \land A \cdot I = A \land A \cdot D = I$$

Additive

$$\exists\, O \in \mathbb{G}\, \forall\, A, B, C \in \mathbb{G}\, \exists\, D \in \mathbb{G}\; A + B \in \mathbb{G} \land (A + B) + C = A + (B + C) \land A + O = A \land A + D = O$$

If, on the other hand, we replace the invertibility axiom with the following, then we do not define a group:

Generic | Multiplicative | Additive | All

Generic

Fake invertibility: $\forall\, A \in \mathbb{G}\, \exists\, B \in \mathbb{G}\, \forall\, C \in \mathbb{G}\; C \circ (A \circ B) = C$

Multiplicative

Fake invertibility: $\forall\, A \in \mathbb{G}\, \exists\, B \in \mathbb{G}\, \forall\, C \in \mathbb{G}\; C \cdot (A \cdot B) = C$

Additive

Fake invertibility: $\forall\, A \in \mathbb{G}\, \exists\, B \in \mathbb{G}\, \forall\, C \in \mathbb{G}\; C + (A + B) = C$

For example, the binary operation could then be defined as $X \circ Y = X$ so that every element is a right identity and every element is a right inverse for every element, which is clearly not the same (and therefore not a group). We will see soon that there can be only one identity element and that each element has a unique inverse, which is why this aspect is often ignored.

▼ **Group-like algebraic structures**

If you remove one or several axioms from the above definition, you get other algebraic structures, most of which have names:

Structure	Closure	Associativity	Identity	Invertibility
Semigroupoid	✗	✓	✗	✗
Small category	✗	✓	✓	✗
Groupoid	✗	✓	✓	✓
Magma	✓	✗	✗	✗
Quasigroup	✓	✗	✗	✓
Unital magma	✓	✗	✓	✗
Semigroup	✓	✓	✗	✗
Loop	✓	✗	✓	✓
Inverse semigroup	✓	✓	✗	✓
Monoid	✓	✓	✓	✗
Group	✓	✓	✓	✓

The axioms of various group-like algebraic structures.

▼ **Alternative group axioms**

Instead of requiring that there is an identity and that each element has an inverse, we can simply require the following axiom:

Generic | Multiplicative | Additive | All

Generic

Solvability: $\forall\, A, B \in \mathbb{G}\ \exists\, X, Y \in \mathbb{G}\ X \circ A = A \circ Y = B$

Multiplicative

Solvability: $\forall\, A, B \in \mathbb{G}\ \exists\, X, Y \in \mathbb{G}\ X \cdot A = A \cdot Y = B$

Additive

Solvability: $\forall\, A, B \in \mathbb{G}\ \exists\, X, Y \in \mathbb{G}\ X + A = A + Y = B$

This means that any element can be reached from any other element both from the left and from the right. This implies that every element can be reached from itself, i.e. every element has an identity element. In order to prove that this axiom, together with closure and associativity, defines the same algebraic structure as above, we just need show that the identity element is the same for all elements:

Generic | Multiplicative | Additive | All

Generic

For arbitrary elements A and B, there exist

- E_A and E_B so that $E_A \circ A = A$ and $B \circ E_B = B$,
- and X and Y so that $X \circ B = E_A$ and $A \circ Y = E_B$.

Thus, $E_A = X \circ B = X \circ (B \circ E_B) = (X \circ B) \circ E_B = E_A \circ E_B = E_A \circ (A \circ Y) = (E_A \circ A) \circ Y = A \circ Y = E_B$.

Multiplicative

For arbitrary elements A and B, there exist

- I_A and I_B so that $I_A \cdot A = A$ and $B \cdot I_B = B$,
- and X and Y so that $X \cdot B = I_A$ and $A \cdot Y = I_B$.

Thus, $I_A = X \cdot B = X \cdot (B \cdot I_B) = (X \cdot B) \cdot I_B = I_A \cdot I_B = I_A \cdot (A \cdot Y) = (I_A \cdot A) \cdot Y = A \cdot Y = I_B$.

Additive

For arbitrary elements A and B, there exist

- O_A and O_B so that $O_A + A = A$ and $B + O_B = B$,
- and X and Y so that $X + B = O_A$ and $A + Y = O_B$.

Thus, $O_A = X + B = X + (B + O_B) = (X + B) + O_B = O_A + O_B = O_A + (A + Y) = (O_A + A) + Y = A + Y = O_B$.

As every equation with two known values and one unknown value has a solution (see the axiom), every element has an inverse.

Properties of equality

Before we can derive properties from the above group axioms, we need to discuss the four basic properties of equality:

- **Reflexivity (E1)**: $\forall\, A \in \mathbb{G}\ A = A$
- **Symmetry (E2)**: $\forall\, A, B \in \mathbb{G}\ A = B \implies B = A$
- **Transitivity (E3)**: $\forall\, A, B, C \in \mathbb{G}\ A = B \land B = C \implies A = C$
- **Substitution (E4)**: $\forall\, F : \mathbb{G} \to \mathbb{G}\ \forall\, A, B \in \mathbb{G}\ A = B \implies F(A) = F(B)$

These properties will be used a lot in the proofs below. Because they are so elementary, I won't point out when I use them.

▼ **Logical conjunction $\land$**

The logical conjunction $\land$ ("and") is true if and only if both operands are true ($\top$) instead of false ($\bot$):

P	Q	$P \wedge Q$
$\bot$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$
$\top$	$\bot$	$\bot$
$\top$	$\top$	$\top$

The definition of $\wedge$.

If you struggle to remember which of $\wedge$ and $\vee$ is which, it may help to know that the logical conjunction $\wedge$ corresponds to the intersection $\cap$ of the two "events" P and Q, and that the logical disjunction $\vee$ corresponds to their union $\cup$.

Derived group properties

Preservation of equality (G6)

It follows directly from the substitution property that equality is preserved when we apply the same element on both sides:

Generic | Multiplicative | Additive | All

Generic

$$\forall\, A, B, C \in \mathbb{G}\; A = B \implies A \circ C = B \circ C$$

using substitution (E4) with $F(X) = X \circ C$

Multiplicative

$$\forall\, A, B, C \in \mathbb{G}\; A = B \implies A \cdot C = B \cdot C$$

using substitution (E4) with $F(X) = X \cdot C$

Additive

$$\forall\, A, B, C \in \mathbb{G}\; A = B \implies A + C = B + C$$

using substitution (E4) with $F(X) = X + C$

Generalized associative law (G7)

We prove by induction that every possible parenthesization of $n > 3$ elements is equivalent ($n = 3$ is covered by the axiom G2):

Generic | Multiplicative | Additive | All

Generic

Every parenthesization of $A_1 \circ A_2 \circ A_3 \circ \ldots \circ A_n$ is equal to the left-associated expression $(((A_1 \circ A_2) \circ A_3) \circ \ldots) \circ A_n$ because any such expression can be written as $B \circ C$, where $B = A_1 \circ \ldots \circ A_m$, $C = A_{m+1} \circ \ldots \circ A_n$, and m is the position of the outermost operation. Both B and C are parenthesized in an unknown way, but since they both contain fewer elements than n, we know by induction that they have left-associated equivalents B' and C'. If $m = n - 1$, $B' \circ C$ is a left-associated expression and we are done. Otherwise, C' can be written as $D \circ A_n$. Now, $B' \circ (D \circ A_n) = (B' \circ D) \circ A_n$ because of associativity (G2). Since $B' \circ D$ contains less than n elements, it has a left-associated equivalent, which makes the expression left-associated. (We know that B' and D can be represented by elements of the set because of closure (G1).)

Multiplicative

Every parenthesization of $A_1 \cdot A_2 \cdot A_3 \cdot \ldots \cdot A_n$ is equal to the left-associated expression $(((A_1 \cdot A_2) \cdot A_3) \cdot \ldots) \cdot A_n$ because any such expression can be written as $B \cdot C$, where $B = A_1 \cdot \ldots \cdot A_m$, $C = A_{m+1} \cdot \ldots \cdot A_n$, and m is the position of the outermost operation. Both B and C are parenthesized in an unknown way, but since they both contain fewer elements than n, we know by induction that they have left-associated equivalents B' and C'. If $m = n - 1$, $B' \cdot C$ is a left-associated expression and we are done. Otherwise, C' can be written as $D \cdot A_n$. Now, $B' \cdot (D \cdot A_n) = (B' \cdot D) \cdot A_n$ because of associativity (G2). Since $B' \cdot D$ contains less than n elements, it has a left-associated equivalent, which makes the expression left-associated. (We know that B' and D can be represented by elements of the set because of closure (G1).)

Additive

Every parenthesization of $A_1 + A_2 + A_3 + \ldots + A_n$ is equal to the left-associated expression $(((A_1 + A_2) + A_3) + \ldots) + A_n$ because any such expression can be written as $B + C$, where $B = A_1 + \ldots + A_m$, $C = A_{m+1} + \ldots + A_n$, and m is the position of the outermost operation. Both B and C are parenthesized in an unknown way, but since they both contain fewer elements than n, we know by induction that they have left-associated equivalents B' and C'. If $m = n - 1$, $B' + C$ is a left-associated expression and we are done. Otherwise, C'

can be written as $D + A_n$. Now, $B' + (D + A_n) = (B' + D) + A_n$ because of associativity (G2). Since $B' + D$ contains less than n elements, it has a left-associated equivalent, which makes the expression left-associated. (We know that B' and D can be represented by elements of the set because of closure (G1).)

Uniqueness of right identity (G8)

▼ **Idempotence**

We say that an element is idempotent if it equals itself when it is combined with itself:

Generic | Multiplicative | Additive | All

Generic

Idempotence (IP): $A \circ A = A$

Multiplicative

Idempotence (IP): $A \cdot A = A$

Additive

Idempotence (IP): $A + A = A$

All idempotent elements of a group are equal to the same identity element:

Generic | Multiplicative | Additive | All

Generic

$$\exists\, E \in \mathbb{G}\; \forall\, A \in \mathbb{G}\; \exists\, B \in \mathbb{G}\; A \circ E \overset{\text{G3}}{=} A \land A \circ B \overset{\text{G4}}{=} E \land$$
$$\big(A \circ A \overset{\text{IP}}{=} A \implies A \overset{\text{G3}}{=} A \circ E \overset{\text{G4}}{=} A \circ (A \circ B) \overset{\text{G2}}{=} (A \circ A) \circ B \overset{\text{IP}}{=} A \circ B \overset{\text{G4}}{=} E\big)$$

Multiplicative

$$\exists\, I \in \mathbb{G}\; \forall\, A \in \mathbb{G}\; \exists\, B \in \mathbb{G}\; A \cdot I \overset{\text{G3}}{=} A \land A \cdot B \overset{\text{G4}}{=} I \land$$
$$\big(A \cdot A \overset{\text{IP}}{=} A \implies A \overset{\text{G3}}{=} A \cdot I \overset{\text{G4}}{=} A \cdot (A \cdot B) \overset{\text{G2}}{=} (A \cdot A) \cdot B \overset{\text{IP}}{=} A \cdot B \overset{\text{G4}}{=} I\big)$$

Additive

$$\exists\, O \in \mathbb{G}\; \forall\, A \in \mathbb{G}\; \exists\, B \in \mathbb{G}\; A + O \overset{\text{G3}}{=} A \land A + B \overset{\text{G4}}{=} O \land$$
$$\big(A + A \overset{\text{IP}}{=} A \implies A \overset{\text{G3}}{=} A + O \overset{\text{G4}}{=} A + (A + B) \overset{\text{G2}}{=} (A + A) + B \overset{\text{IP}}{=} A + B \overset{\text{G4}}{=} O\big)$$

Since an identity element is an identity for all elements, including itself, it is idempotent. Thus, all identity elements are the same. As a consequence, we no longer need to quantify the identity element. Depending on the notation, E, O, or I simply refers to the unique identity element from now on. The identity element can also be seen as the output of a nullary function.

▼ **Unique identity in commutative groups**

If the operation is commutative, it's much easier to see why any two identities are the same:

Generic | Multiplicative | Additive | All

Generic

$$E_1 \overset{\text{G3}}{=} E_1 \circ E_2 \overset{\text{G5}}{=} E_2 \circ E_1 \overset{\text{G3}}{=} E_2$$

Multiplicative

$$I_1 \overset{\text{G3}}{=} I_1 \cdot I_2 \overset{\text{G5}}{=} I_2 \cdot I_1 \overset{\text{G3}}{=} I_2$$

Additive

$$O_1 \overset{\text{G3}}{=} O_1 + O_2 \overset{\text{G5}}{=} O_2 + O_1 \overset{\text{G3}}{=} O_2$$

Right inverses are left inverses (G9)

We show that when we apply any right inverse from the left, the resulting element is idempotent and thus equal to the only identity:

Generic | Multiplicative | Additive | All

Generic

$$\forall\, A, B \in \mathbb{G}\; A \circ B \overset{\text{G4}}{=} E$$

$$\Downarrow$$

$$(B \circ A) \circ (B \circ A) \overset{G7}{=} (B \circ (A \circ B)) \circ A \overset{G4}{=} (B \circ E) \circ A \overset{G3}{=} B \circ A \overset{G8}{=} E$$

Multiplicative

$$\forall\, A, B \in \mathbb{G}\ A \cdot B \overset{G4}{=} I$$

$$\Downarrow$$

$$(B \cdot A) \cdot (B \cdot A) \overset{G7}{=} (B \cdot (A \cdot B)) \cdot A \overset{G4}{=} (B \cdot I) \cdot A \overset{G3}{=} B \cdot A \overset{G8}{=} I$$

Additive

$$\forall\, A, B \in \mathbb{G}\ A + B \overset{G4}{=} O$$

$$\Downarrow$$

$$(B + A) + (B + A) \overset{G7}{=} (B + (A + B)) + A \overset{G4}{=} (B + O) + A \overset{G3}{=} B + A \overset{G8}{=} O$$

▼ **Alternative proof**

We can also prove that a right inverse is also a left inverse as follows:

Generic | Multiplicative | Additive | All

Generic

$$\forall\, A, B, C \in \mathbb{G}\ A \circ B \overset{G4}{=} E \land\ B \circ C \overset{G4}{=} E$$

$$\Downarrow$$

$$B \circ A \overset{G3}{=} (B \circ A) \circ E \overset{G4}{=} (B \circ A) \circ (B \circ C) \overset{G7}{=} (B \circ (A \circ B)) \circ C \overset{G4}{=} (B \circ E) \circ C \overset{G3}{=} B \circ C \overset{G4}{=} E$$

Multiplicative

$$\forall\, A, B, C \in \mathbb{G}\ A \cdot B \overset{G4}{=} I \land\ B \cdot C \overset{G4}{=} I$$

$$\Downarrow$$

$$B \cdot A \overset{G3}{=} (B \cdot A) \cdot I \overset{G4}{=} (B \cdot A) \cdot (B \cdot C) \overset{G7}{=} (B \cdot (A \cdot B)) \cdot C \overset{G4}{=} (B \cdot I) \cdot C \overset{G3}{=} B \cdot C \overset{G4}{=} I$$

Additive

$$\forall\, A, B, C \in \mathbb{G}\ A + B \overset{G4}{=} O \land\ B + C \overset{G4}{=} O$$

$$\Downarrow$$

$$B + A \overset{G3}{=} (B + A) + O \overset{G4}{=} (B + A) + (B + C) \overset{G7}{=} (B + (A + B)) + C \overset{G4}{=} (B + O) + C \overset{G3}{=} B + C \overset{G4}{=} O$$

Right identity is left identity (G10)

The unique right identity is also an identity element when applied from the left:

Generic | Multiplicative | Additive | All

Generic

$$\forall\, A, B \in \mathbb{G}\ A \circ B \overset{G4}{=} E$$

$$\Downarrow$$

$$E \circ A \overset{G4}{=} (A \circ B) \circ A \overset{G2}{=} A \circ (B \circ A) \overset{G9}{=} A \circ E \overset{G3}{=} A$$

Multiplicative

$$\forall\, A, B \in \mathbb{G}\ A \cdot B \overset{G4}{=} I$$

$$\Downarrow$$

$$I \cdot A \overset{G4}{=} (A \cdot B) \cdot A \overset{G2}{=} A \cdot (B \cdot A) \overset{G9}{=} A \cdot I \overset{G3}{=} A$$

Additive

$$\forall\, A, B \in \mathbb{G}\ A + B \overset{G4}{=} O$$

$$\Downarrow$$

$$O + A \overset{G4}{=} (A + B) + A \overset{G2}{=} A + (B + A) \overset{G9}{=} A + O \overset{G3}{=} A$$

Using the same reasoning as above, the left identity is also unique.

Uniqueness of inverses (G11)

It follows that any two inverses of the same element are the same:

Generic | Multiplicative | Additive | All

Generic

$$\forall\, A, B_1, B_2 \in \mathbb{G}\ A \circ B_1 \overset{\text{G4}}{=} E \land A \circ B_2 \overset{\text{G4}}{=} E$$
$$\Downarrow$$
$$B_1 \overset{\text{G3}}{=} B_1 \circ E \overset{\text{G4}}{=} B_1 \circ (A \circ B_2) \overset{\text{G2}}{=} (B_1 \circ A) \circ B_2 \overset{\text{G9}}{=} E \circ B_2 \overset{\text{G10}}{=} B_2$$

Multiplicative

$$\forall\, A, B_1, B_2 \in \mathbb{G}\ A \cdot B_1 \overset{\text{G4}}{=} I \land A \cdot B_2 \overset{\text{G4}}{=} I$$
$$\Downarrow$$
$$B_1 \overset{\text{G3}}{=} B_1 \cdot I \overset{\text{G4}}{=} B_1 \cdot (A \cdot B_2) \overset{\text{G2}}{=} (B_1 \cdot A) \cdot B_2 \overset{\text{G9}}{=} I \cdot B_2 \overset{\text{G10}}{=} B_2$$

Additive

$$\forall\, A, B_1, B_2 \in \mathbb{G}\ A + B_1 \overset{\text{G4}}{=} O \land A + B_2 \overset{\text{G4}}{=} O$$
$$\Downarrow$$
$$B_1 \overset{\text{G3}}{=} B_1 + O \overset{\text{G4}}{=} B_1 + (A + B_2) \overset{\text{G2}}{=} (B_1 + A) + B_2 \overset{\text{G9}}{=} O + B_2 \overset{\text{G10}}{=} B_2$$

Given that every element of the group has an inverse and that the inverse is unique for each element, inversion is a unary operation on the set of elements. Instead of quantifying the inverse as above, we can use a much simpler notation for the inverse from now on:

Generic | Multiplicative | Additive | **All**

Generic

$$\forall\, A \in \mathbb{G}\ A \circ \overline{A} = \overline{A} \circ A = E$$

Multiplicative

$$\forall\, A \in \mathbb{G}\ A \cdot A^{-1} = A^{-1} \cdot A = I$$

Additive

$$\forall\, A \in \mathbb{G}\ A + (-A) = (-A) + A = O$$

Cancellation property (G12)

Equality is also preserved when you remove the same element from both sides, which is known as the cancellation property:

Generic | Multiplicative | Additive | All

Generic

$$\forall\, A, B, C \in \mathbb{G}\ A \circ C = B \circ C$$
$$\overset{\text{G6}}{\Longrightarrow}\ (A \circ C) \circ \overline{C} = (B \circ C) \circ \overline{C}$$
$$\overset{\text{G2}}{\Longrightarrow}\ A \circ (C \circ \overline{C}) = B \circ (C \circ \overline{C})$$
$$\overset{\text{G4}}{\Longrightarrow}\ A \circ E = B \circ E$$
$$\overset{\text{G3}}{\Longrightarrow}\ A = B$$

Multiplicative

$$\forall\, A, B, C \in \mathbb{G}\ A \cdot C = B \cdot C$$
$$\overset{\text{G6}}{\Longrightarrow}\ (A \cdot C) \cdot C^{-1} = (B \cdot C) \cdot C^{-1}$$
$$\overset{\text{G2}}{\Longrightarrow}\ A \cdot (C \cdot C^{-1}) = B \cdot (C \cdot C^{-1})$$
$$\overset{\text{G4}}{\Longrightarrow}\ A \cdot I = B \cdot I$$
$$\overset{\text{G3}}{\Longrightarrow}\ A = B$$

Additive

$$\forall\, A, B, C \in \mathbb{G}\ A + C = B + C$$
$$\overset{\text{G6}}{\Longrightarrow}\ (A + C) + (-C) = (B + C) + (-C)$$
$$\overset{\text{G2}}{\Longrightarrow}\ A + (C + (-C)) = B + (C + (-C))$$
$$\overset{\text{G4}}{\Longrightarrow}\ A + O = B + O$$
$$\overset{\text{G3}}{\Longrightarrow}\ A = B$$

Unique solution (G13)

As we saw earlier, the following equation has a unique solution in every group (also if the element to be determined is on the right):

Generic | Multiplicative | Additive | All

Generic

$$\begin{aligned} &\forall\, X, A, B \in \mathbb{G}\ X \circ A = B \\ &\overset{G6}{\implies} (X \circ A) \circ \overline{A} = B \circ \overline{A} \\ &\overset{G2}{\implies} X \circ (A \circ \overline{A}) = B \circ \overline{A} \\ &\overset{G4}{\implies} X \circ E = B \circ \overline{A} \\ &\overset{G3}{\implies} X = B \circ \overline{A} \end{aligned}$$

Multiplicative

$$\begin{aligned} &\forall\, X, A, B \in \mathbb{G}\ X \cdot A = B \\ &\overset{G6}{\implies} (X \cdot A) \cdot A^{-1} = B \cdot A^{-1} \\ &\overset{G2}{\implies} X \cdot (A \cdot A^{-1}) = B \cdot A^{-1} \\ &\overset{G4}{\implies} X \cdot I = B \cdot A^{-1} \\ &\overset{G3}{\implies} X = B \cdot A^{-1} \end{aligned}$$

Additive

$$\begin{aligned} &\forall\, X, A, B \in \mathbb{G}\ X + A = B \\ &\overset{G6}{\implies} (X + A) + (-A) = B + (-A) \\ &\overset{G2}{\implies} X + (A + (-A)) = B + (-A) \\ &\overset{G4}{\implies} X + O = B + (-A) \\ &\overset{G3}{\implies} X = B + (-A) \end{aligned}$$

The solution is unique because any two solutions X_1 and X_2 are the same:

Generic | Multiplicative | Additive | All

Generic

$$\begin{aligned} &\forall\, X_1, X_2, A, B \in \mathbb{G}\ X_1 \circ A = B \land X_2 \circ A = B \\ &\overset{E3}{\implies} X_1 \circ A = X_2 \circ A \\ &\overset{G12}{\implies} X_1 = X_2 \end{aligned}$$

Multiplicative

$$\begin{aligned} &\forall\, X_1, X_2, A, B \in \mathbb{G}\ X_1 \cdot A = B \land X_2 \cdot A = B \\ &\overset{E3}{\implies} X_1 \cdot A = X_2 \cdot A \\ &\overset{G12}{\implies} X_1 = X_2 \end{aligned}$$

Additive

$$\begin{aligned} &\forall\, X_1, X_2, A, B \in \mathbb{G}\ X_1 + A = B \land X_2 + A = B \\ &\overset{E3}{\implies} X_1 + A = X_2 + A \\ &\overset{G12}{\implies} X_1 = X_2 \end{aligned}$$

Double inverse theorem (G14)

The inverse of the inverse is the original element again:

Generic | Multiplicative | Additive | All

Generic

$$\forall\, A \in \mathbb{G}\ A \circ \overline{A} = E \overset{G13}{\implies} A = E \circ \overline{\overline{A}} \overset{G10}{\implies} A = \overline{\overline{A}}$$

Multiplicative

$$\forall\, A \in \mathbb{G}\ A \cdot A^{-1} = I \overset{G13}{\implies} A = I \cdot (A^{-1})^{-1} \overset{G10}{\implies} A = (A^{-1})^{-1}$$

Additive

$$\forall\, A \in \mathbb{G}\ A + (-A) = O \overset{G13}{\implies} A = O + (-(-A)) \overset{G10}{\implies} A = -(-A)$$

Inversion of combination (G15)

We can invert a combination of two elements by combining their inverses in reverse order:

Generic | Multiplicative | Additive | All

Generic

$$\forall\, A \in \mathbb{G}\; (A \circ B) \circ (\overline{B} \circ \overline{A}) \overset{G7}{=} (A \circ (B \circ \overline{B})) \circ \overline{A} \overset{G4}{=} (A \circ E) \circ \overline{A} \overset{G3}{=} A \circ \overline{A} \overset{G4}{=} E$$

$$\Downarrow \text{G11}$$

$$\overline{A \circ B} = \overline{B} \circ \overline{A}$$

Multiplicative

$$\forall\, A \in \mathbb{G}\; (A \cdot B) \cdot (B^{-1} \cdot A^{-1}) \overset{G7}{=} (A \cdot (B \cdot B^{-1})) \cdot A^{-1} \overset{G4}{=} (A \cdot I) \cdot A^{-1} \overset{G3}{=} A \cdot A^{-1} \overset{G4}{=} I$$

$$\Downarrow \text{G11}$$

$$(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$$

Additive

$$\forall\, A \in \mathbb{G}\; (A + B) + ((-B) + (-A)) \overset{G7}{=} (A + (B + (-B))) + (-A) \overset{G4}{=} (A + O) + (-A) \overset{G3}{=} A + (-A) \overset{G4}{=} O$$

$$\Downarrow \text{G11}$$

$$-(A + B) = (-B) + (-A)$$