Modular Arithmetic in Finite Groups, Rings, Fields

OWO lecture notes, Martin Otto 2021

Contents: groups, rings, fields as algebraic structures; divisibility, primality and gcd over the integers, Euclid's algorithm; congruence modulo n and modular arithmetic; the finite rings $\mathbb{Z}_n$ and the finite fields $\mathbb{Z}_p$; outlook towards applications in cryptography.

1 Algebraic structures: groups, rings, fields

Algebraic structures consist of a non-empty domain together with a number of specified operations (functions of specified arities over that domain) and constants (designated elements of the domain). Common examples are the usual number domains like $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ with arithmetical operations like $+, \cdot$ (binary) and/or additive inversion $x \mapsto -x$ (unary, but not available over $\mathbb{N}$ (!)), and possibly constants like $0, 1$ for numbers which play a special rôle w.r.t. such arithmetical operations.

We collect some properties of interest for operations like these.

Definition 1 [properties of binary operations]
Consider a binary operation $\circ\colon A \times A \to A$ over a non-empty domain A.

(i) $\circ$ is *associative* if, for all $a, b, c \in A$,
$a \circ (b \circ c) = (a \circ b) \circ c$;

(ii) $\circ$ is *commutative* if, for all $a, b \in A$,
$a \circ b = b \circ a$;

(iii) $\circ$ has $e \in A$ as a *neutral element* if, for all $a \in A$,
$a \circ e = e \circ a = a$;

(iv) $\circ$ has *inverses* (w.r.t. the neutral element $e \in A$) if
for all $a \in A$ there is some $b \in A$ such that $a \circ b = b \circ a = e$;

(v) $\circ$ *distributes* (from the left) over a second binary operation $*\colon A \times A \to A$ if,
for all $a, b, c \in A$, $a \circ (b * c) = (a \circ b) * (a \circ c)$.

Definition 2 [groups]
A structure $(A, \circ, e)$ with a binary operation $\circ$ and constant $e \in A$ is a *group* [Gruppe] if $\circ$ is associative and has inverses w.r.t. the neutral element e. A group $(A, \circ, e)$ is called *commutative* or *Abelian* if the group operation $\circ$ is commutative.

Definition 3 [rings and fields]
A structure $(A, +, \cdot, 0, 1)$ with two binary operations $+, \cdot$ over A that are both associative and commutative (referred to as addition and multiplication) and constants $0, 1 \in A$ is a *ring* [Ring] if

(i) $(A, +, 0)$ is a commutative group (the additive group of the ring);
(ii) 1 is a neutral element for the multiplication operation $\cdot$;
(iii) multiplication $\cdot$ distributes over addition $+$, and $0 \cdot a = 0$ for all $a \in A$.

A ring $(A, +, \cdot, 0, 1)$ is a *field* [Körper] if
$1 \neq 0$ and all $a \neq 0$ have inverses w.r.t. multiplication, i.e. if the restriction of $\cdot$ to elements other than 0 forms another group (the multiplicative group of the field).

Familiar examples. The standard number fields $(\mathbb{R}, +, \cdot, 0, 1)$ and $(\mathbb{Q}, +, \cdot, 0, 1)$ are fields (multiplicative inverses through division by non-zero numbers). The ring of integers $(\mathbb{Z}, +, \cdot, 0, 1)$ is not a field, as division is not available; the additive group $(\mathbb{Z}, +, 0)$ has inverses, unlike $(\mathbb{Z} \setminus \{0\}, \cdot, 1)$ for multiplication. Over the natural numbers $\mathbb{N}$ both addition and multiplication are associative and commutative with neutral elements 0 and 1, respectively, but inverses are not generally available.

We shall come to examples of *finite* number domains that form groups, rings and fields with suitably adapted addition and multiplication operations in Section 3 below.

Example 4 Natural examples of finite and infinite groups that are not usually commutative arise on sets of *symmetries* (structure-preserving invertible transformations) of mathematical structures, or e.g. geometric patterns, with composition of transformations as the binary operation, and with the trivial identity transformation, which fixes everything point-wise, as the neutral element w.r.t. composition.

2 Integer divisibility and division with remainder

Background terminology and basic facts. An integer $a \in \mathbb{Z}$ is *divisible* by the non-zero natural number (or integer) d if $a = k \cdot d$ for some $k \in \mathbb{Z}$; note that 0 is divisible by any non-zero d. Common notation for "d divides a" is $d|a$. Any non-zero number is trivially divisible by 1 and by itself. A natural number $p > 1$ that is divisible only by 1 and itself is called a *prime* (note that 1 is not considered prime). It is well known that any natural number $n > 1$ can be decomposed into a product of primes, which is unique up to the arrangement of the factors (commutativity!). A related feature about divisibility, which is also not proved here, is the following.

Fact 5 *A prime p divides an integer $m \in \mathbb{Z}$ if, and only if, p is a factor in the prime decomposition of m; it follows that p divides a product $m \cdot n$ of integers if, and only if, it divides at least one of the factors m, n.*

2.1 Division with remainder: a mod b

Integers can always be divided *with remainder* (for any non-zero divisor, that is). A *remainder* w.r.t. divison by some natural number $n > 0$ can be any integer between 0 and $n - 1$ (inclusive).

Definition 6 [remainders and congruence modulo n]
The *remainder* of $a \in \mathbb{Z}$ w.r.t. divison by non-zero $n \in \mathbb{N}$ is the unique r in the range $0 \leqslant r < n$ for which $a - r$ is divisible by n, i.e. such that there is some $k \in \mathbb{Z}$ for which

$$a = k \cdot n \ + \ r \qquad (0 \leqslant r < n)$$

We write $a \bmod n$ (read: "a modulo n", or just "a mod n") for this remainder of $a \in \mathbb{Z}$ w.r.t. divison by n. For fixed $n > 0$, two integers a and b are called *congruent modulo* n, denoted as

$$a \equiv_n b$$

if their remainders w.r.t. division by n are identical, i.e. if $a \bmod n = b \bmod n$.

It is easy to check that $a \equiv_n b$ if, and only if, $n|(a - b)$: their difference is an integer multiple of n. Division with remainder is closely connected with *rounding* (to the largest integer below some rational). For $a, b \in \mathbb{N}$ where $b > 0$ we write $\lfloor a/b \rfloor$ for

$$\lfloor a/b \rfloor := \max\big\{d \in \mathbb{N} \colon d \cdot b \leqslant a\big\}$$

and, **Exercise**, check by simple arithmetic that $a \bmod b = a - \lfloor a/b \rfloor b$. ([1])

The relation $\equiv_n$, congruence modulo n, is an example of an *equivalence relation*. Such relations can be seen as variants of equality, based on a possibly much coarser level of classification as to when two objects are to be considered indistinguishable. Clusters of mutually equivalent objects form so-called *equivalence classes*, and each member of such an equivalence class becomes a *representative* of its entire equivalence class. In our case, of $\equiv_n$ as an equivalence relation over $\mathbb{Z}$, the equivalence classes modulo n are the

$$[a]_n := \{b \in \mathbb{Z} \colon a \equiv_n b\}, \text{ the equivalence class of } a \in \mathbb{Z},$$

each of them infinite. There are just exactly n many of them, which can be listed as $[0]_n, \ldots, [n-1]_n$, so that the equivalence class $[a]_n$ of a is represented by its remainder modulo n:

$$[a]_n = [a \bmod n]_n,$$

and all $\mathbb{Z}$ is the disjoint union of these n-many classes. For an intuitive image, think of $\mathbb{Z}$ wrapped around in a cycle of length n. The set of classes is denoted as

$$\mathbb{Z}_n := \{[a]_n \colon a \in \mathbb{Z}\} \ ([2])$$

and can (via our preferred choice of representatives) also be represented just by the set $\{0, \ldots, n-1\}$ of possible remainders. Interestingly, these finite domains $\mathbb{Z}_n$ inherit natural arithmetical operations from those on $\mathbb{Z}$, see Section 3.

[1] We now often drop the explicit notation $\cdot$ for multiplication, as in writing just $\lfloor a/b \rfloor b$ for $\lfloor a/b \rfloor \cdot b$.
[2] Another suggestive notation for this set is $\mathbb{Z}/\equiv_n$ to be read as "$\mathbb{Z}$ modulo n."

2.2 Greatest common divisors and Euclid's algorithm

For non-negative integers a, b, not both equal to 0,

$$\gcd(a,b) = \max\{d \in \mathbb{N} \colon d|a \text{ and } d|b\}$$

is the *greatest common divisor* [größter gemeinsamer Teiler, ggT] of a and b. One can show (essentially based on Fact 5; do you see how?) that the above definition always yields a unique result for $\gcd(a,b)$. We usually assume the convention to list the arguments for gcd such that $a \geqslant b$. For $b = 0$ we get $\gcd(a,b) = a$, for every $a > 0$. Numbers a and b are called *relatively prime* [teilerfremd] if $\gcd(a,b) = 1$, i.e. if they share no prime factors.

Fact 7 *For all* $a, b > 0$*:* $\quad \gcd(a,b) = \gcd(b, a \bmod b)$.

Exercise. To prove this fact, use elementary arithmetic to show directly that d divides both a and b if, and only if, d divides both $a \bmod b$ and b.

Fact 7 supports an elegant algorithm for a straightforward (and computationally feasible) gcd-computation. Euclid's algorithm was known to (though possibly not invented by) Euclid, in the 4th century BC, and may be regarded as one of the earliest non-trivial mathematical algorithms.

EUCLID(a,b) assume $0 \neq a \geqslant b \geqslant 0$

```
1   IF b = 0  THEN return a
2             ELSE return EUCLID(b, a mod b)
```

Correctness is shown on the basis of Fact 7 above. An extended form of this algorithm also computes not just $\gcd(a,b)$ but also values $k, \ell \in \mathbb{Z}$ for which

$$\gcd(a,b) = ka + \ell b.$$

EXTENDED-EUCLID(a,b) assume $0 \neq a \geqslant b \geqslant 0$

```
1   IF b = 0  THEN return (a, 1, 0)
2             ELSE    DO
3                     (d,x,y) := EXTENDED-EUCLID(b, a mod b)
4                     return (d, y, x − ⌊a/b⌋y)
5                     OD
```

EXTENDED-EUCLID has the same recursive structure as EUCLID. It performs just like EUCLID w.r.t. to variable (and output) d. For its correctness it just remains to prove the equation

$$d = \gcd(a,b) = ka + lb \qquad (*)$$

for the output (d, k, l) of EXTENDED-EUCLID(a,b). We do so by *induction*, namely by establishing that

(i) (base case) equation $(*)$ is true in case of termination in line (1);

(ii) (induction step) assuming that line (3) returns d, x, y satisfying the claim $(*)$ for Extended-Euclid$(b, a \bmod b)$, so does $(d, y, x - \lfloor a/b \rfloor y)$ w.r.t. for Extended-Euclid(a, b).

Of these, (i) is obvious, and (ii) follows by simple arithmetic. Indeed,

$$d = \gcd(b, a \bmod b) = xb + y(a \bmod b)$$

implies that $d = \gcd(a, b)$ and that

$$ya + (x - \lfloor a/b \rfloor y)b = xb + y(a - \lfloor a/b \rfloor b) = xb + y(a \bmod b) = d$$

as required. As an **Exercise** you should check the arithmetic behind the last equality and think about how and why (i) and (ii) together do indeed establish the claim for all runs of this algorithm, i.e. for all legitimate arguments a, b.

The fact that $\gcd(a, b)$ is an *integer linear combination* of the arguments a and b as in $(*)$ will be useful for solving simple equations in modular arithmetic and plays a key role in some cryptographic applications to be discussed in Section **??**. It is also essential for a proof of the following characterisation of the gcd, which we omit here.

Proposition 8 *For all $a, b > 0$:*

(i) $\gcd(a, b)$ *is the smallest positive number of the form $ka + lb$ for any $k, l \in \mathbb{Z}$.*

(ii) The numbers of the form $ka+lb$, for $k, l \in \mathbb{Z}$, are precisely the multiples of $\gcd(a, b)$.

3 Modular arithmetic

Recall from Section 2.1 that $\mathbb{Z}_n$ stands for the set of equivalence classes of integers w.r.t. the equivalence relation of *congruence modulo n*:

$$a \equiv_n b \quad \text{if, and only if,} \quad a \bmod n = a \bmod n \quad \text{if, and only if,} \quad n|(a - b).$$

In other words, we identify numbers having the same remainder w.r.t. division by n. The minimal non-negative member in the class $[a]_n$ is just the remainder $a \bmod n$. As $0 \leqslant a \bmod n < n$, we may represent $\mathbb{Z}_n$ by the n-element set $\{0, \ldots, n-1\}$, which formally is a complete set of representatives for $\mathbb{Z}_n$.

Modular addition and multiplication. Addition and multiplication on $\mathbb{Z}$ carry over to $\mathbb{Z}_n$ by putting

$$[a]_n + [b]_n := [a + b]_n \quad \text{and} \quad [a]_n \cdot [b]_n := [ab]_n.$$

One needs to check that these operations are well-defined over $\mathbb{Z}_n$ (!), i.e. that the equivalence classes stipulated as values on the right-hand sides do *not* depend on the chosen representatives a for $[a]_n$ and b for $[b]_n$. We leave this as an **Exercise**.

Proposition 9

(i) For all $n > 0$: $(\mathbb{Z}_n, +, [0]_n)$ is a commutative group, and $(\mathbb{Z}_n, +, \cdot, [0]_n, [1]_n)$ a ring.

(ii) For primes p: $(\mathbb{Z}_p, +, \cdot, [0]_p, [1]_p)$ is a field.

Proof For (i) one checks that the required equalities for associativity, commutativity, distributivity and neutral elements carry over from corresponding equalities for representatives in $\mathbb{Z}$. For additive inverses, we note that $[a]_n + [n-a]_n = [n]_n = [0]_n$.

The crucial observation towards (ii) then concerns the existence of multiplicative inverses (which we do not have in $\mathbb{Z}$ (!)). For a prime p and any a in the range $1, \ldots, p-1$, the numbers a and p are relatively prime, i.e. $\gcd(p, a) = 1$ (why?). From Section 2.2 we therefore know that there are $k, \ell \in \mathbb{Z}$ such that $1 = \gcd(p, a) = kp + \ell a$. This implies that $[1]_p = [kp + \ell a]_p = [kp]_p + [\ell a]_p = [\ell a]_p = [\ell]_p \cdot [a]_p = [a]_p \cdot [\ell]_p$ so that $[\ell]_p$ is the desired inverse for $[a]_p$ in $(\mathbb{Z}_p \setminus \{[0]_p\}, \cdot, [1]_p)$. □

Theorem 10 (Fermat's little theorem)
For all primes p and all $a \not\equiv_p 0$: $a^{p-1} \equiv_p 1$.

Note that the claim implies that $a^p \equiv_p a$ for all $a \in \mathbb{Z}_p$, i.e. that $x \mapsto x^p$ is the identity transformation on $\mathbb{Z}_p$.

Proof For the proof we consider *subgroups* of the multiplicative group of (in this case: the field) $\mathbb{Z}_p$, which itself has precisely $p-1$ elements represented by $1, \ldots, p-1$. For fixed $a \not\equiv_p 0$, we look at the subgroup *generated* by a whose domain is the subset $\{[a]_p^k \colon k \in \mathbb{Z}\} \subseteq \mathbb{Z}_p$. Here $[a]^{-n}$ for $n \in \mathbb{N}$ stands for the n-th power of the multiplicative inverse of $[a]_p$, i.e. of $[a]_p^{-1}$. (To see that these elements form a group w.r.t. multiplication within $\mathbb{Z}_p$, we need to check that the set is closed under multiplication and has multiplicative inverses for each of its elements, which is clear from its definition.) In any group, any subset with these closure properties forms a group as a subgroup of the given group. As the supply of elements in the multiplicative group of $\mathbb{Z}_p$ is finte (there are exactly $p-1$ many),

$$\{[a]_p^k \colon k \in \mathbb{Z}\} = \{[a]_p^k \colon 0 \leqslant k < m\}$$

for some $m \in \mathbb{N}$, and for the minimal such m we must have $[a]_p^m = [1]_p$ (why?), and this minimal m is the number of elements in the subgroup. By a general fact about finite groups, which we leave as a slightly more challenging **Exercise**, the size of any subgroup divides the size of the given finite group. Hence $m|(p-1)$, i.e. $p - 1 = \ell m$ for some $\ell \in \mathbb{N}$. It follows that $[a]_p^{p-1} = [a]_p^{m\ell} = ([a]_p^m)^\ell = [1]_p^\ell = [1]_p$ as claimed. □

Efficient exponentiation in $\mathbb{Z}_n$. Addition and multiplication in $\mathbb{Z}_n$ are computationally feasible because they reduce to corresponding algorithms for integers. But in $\mathbb{Z}_n$, even exponentiation is feasibly computable. In contrast, ordinary integer exponentiation $a, b \mapsto a^b$ involves an exponential increase in the lengths of the digital representations (e.g. in binary). But if the results are capped as in modular arithmetic over $\mathbb{Z}_n$, we obtain a feasible exponentiation algorithm, based on *repeated squaring*.

Modular-Exponentiation(a, b, n) assume $a, b > 0$

let $b_k \dots b_0$ be the binary expansion of b: $b = b_0 + 2b_1 + \cdots + 2^k b_k$

```
1  d := 1
2  c := a
3  FOR i = 0, ..., k  DO
4                      IF b_i = 1 THEN d := (d · c) mod n
5                      c := c^2 mod n
6                      OD
7  return d
```

For correctness observe that (in $\mathbb{Z}_n$ as well as in $\mathbb{Z}$):

$$a^b = a^{b_0} \cdot a^{2b_1} \cdots a^{2^k b_k} = a^{b_0} \cdot (a^2)^{b_1} \cdots (a^{2^k})^{b_k}.$$

The variable c successively gets set to a, a^2, a^4, …, $a^{2^k} \bmod n$ in lines 3 and 5, and the corresponding factors are multiplied into d for those b_i that are not 0.

Exercise. Show that $3^{444} + 4^{333}$ is divisible by 5, and that $2^{999} + 5^{999}$ as well as $5^{222} - 2^{222}$ are divisible by 7 (without ever so much as thinking of 3-digit decimal numbers for the intermediate results!).

4 Outlook/excursion: cryptographic applications

Cryptography is the science of the use of *encryption* methods that protect communication. In the simplest setting, consider the problem of transmitting a message M from B (Bob) to A (Alice) over some insecure channel in such a way that even though someone might intercept the message, its contents is not revealed to the eavesdropper. To achieve this, B has to *encrypt* the message and only submit its encrypted form to the insecure channel; the necessary assumptions being that (a) a potential eavesdropper who intercepts the encrypted message cannot (or at least not easily) retrieve the original message; but that the legitimate recipient, A, can *decrypt* the encrypted message to get at the original message.

Encryption and decryption are therefore based on transformations of M which require specific extra information, so-called *keys*, to be (efficiently) computed. More precisely, encryption and decryption are performed by means of matching pairs of transformations, such that the decryption transformation inverts the encryption transformation.

With simple symmetric schemes the same key is necessary for encryption and matching decryption, which raises the problem of making the key used by B (for encryption) available to A (for decryption). This makes a large scale use of such systems impractical, as one either would have to rely on the same insecure channels for the distribution of keys or has to go to the extra trouble of (pre-)arranging key choices by other means (trusted messengers). While the first option clearly compromises security, the second

is not feasible at the scale required for secure everyday communication (outside secret service activities, say).

Surprisingly this *key exchange problem* can be eliminated. A first milestones in the development of modern era cryptography is the discovery of a key exchange protocol by Diffie, Hellman and Merkle that allows Alice and Bob to generate a common secret key interactively without actually transmitting the key itself. A second and even more important milestone occurred with the invention of *public key cryptography* by Rivest, Shamir and Adleman, whose RSA cryptosystem is a widely used standard today.[3]

We generally assume here that the messages in question have already been encoded as bitstrings of some fixed length or numbers in a certain fixed range. (This also means that longer messages have to be cut up and transmitted in blocks of suitable lengths.) In this sense, our *message spaces* will either be domains of fixed length bitstrings $\{0,1\}^n$ or fixed initial segments of the natural numbers $\mathbb{Z}_n = \{0,\ldots,n-1\}$.

4.1 Conventional symmetric cryptography

One of the most simple and generic encryption/decryption schemes of the conventional symmetric key variant uses bit-wise addition $\bmod 2$ (or bitwise exclusive OR, XOR). To use it on a message space of bitstrings of length n, $M = m_1 \ldots m_n \in \{0,1\}^n$, Alice and Bob share the same *key*, which can be used for encryption and decryption (hence "symmetric"). For their key they can use any fixed (and agreed) bitstring of length n, $K = k_1 \ldots k_n \in \{0,1\}^n$. To encrypt a message M, Alice would compute the string M'

$$M' := M \oplus K \quad \text{where } m'_i = (m_i + k_i) \bmod 2.$$

To decrypt, Bob converts M' back into M using the same transformation

$$M := M' \oplus K \quad \text{where } m_i = (m'_i + k_i) \bmod 2.$$

Security relies on the fact that only the encrypted message M' is passed across the insecure channel, and on the assumption that only A and B know their key K.

Of course, in order to be able to communicate in this way, Alice and Bob have to share the information about K: the key exchange problem.

4.2 An interactive key exchange protocol

The gist of the key exchange problem seems to be that in order to communicate securely, Alice and Bob must already share an exclusive secret (their key) beforehand. Can this constraint be avoided? Is this constraint unavoidable, or can keys somehow be generated between A and B, in such a way that only they share the full information about the key, and without transmitting the key between A and B? Unlikely though it seemed at the

[3] Historically it now appears that both these major scientific innovations had previously (and in reverse order) been found but not published by researchers for the British Government, who were not permitted to publish them. These and many other intriguing aspects of the history of cryptography are treated in Simon Singh's books [The code book, 1999] and [The Science of Secrecy, 2000].

time of this discovery, there are protocols that allow A and B to create their common key interactively in a process in which they only exchange auxiliary information from which the resulting key is not feasibly computable.

Such phenomena rest on the existence of functions that are easy to compute but (assumed to be) hard to invert, so-called *one-way functions*. An important example is provided by modular exponentiation (easy, see above) with the discrete logarithm as its inverse. Though not a proven fact, there is good reason to assume that the discrete logarithm is not feasibly computable. The discrete logarithm to base g in $\mathbb{Z}_n$ solves the modular exponential equation

$$g^x \equiv_n y$$

for x, given y (and fixed g and n). It is thus an inverse to modular exponentiation $x \mapsto g^x \bmod n$. The underlying complexity theoretic assumption is that x is not feasibly computable from y, while y is of course feasibly computable from x.

This can be used for a key exchange protocol idea due to Diffie, Hellman and Merkle (1976) as follows.

Alice and Bob agree (publicly) on suitable numbers n and g. In a session in which they want to generate a common key for cryptographic purposes, Alice and Bob independently choose numbers a and b. They do not reveal the numbers a and b to each other, however. Instead Alice sends to Bob the number $g^a \bmod n$ and Bob send to Alice the number $g^b \bmod n$. According to the crucial complexity assumption, a and b are not accessible even to someone who intercepts these messages. Now Alice chooses for her key the number $(g^b \bmod n)^a \bmod n = g^{ab} \bmod n$, which she can compute from the number received from Bob using her own secret a. Bob computes the same number according to $g^{ab} \bmod n = (g^a \bmod n)^b \bmod n$ from the number received from Alice using his secret b.

Now Alice and Bob share the key $g^{ab} \bmod n$, without having communicated any information from which this key could be extracted (if the discrete logarithm is indeed not feasibly computable).

4.3 RSA public key cryptography

In public key cryptography the key exchange problem does not even arise as such, as we are dealing with asymmetric schemes in which A and B need not share the same key. That alone, however, does not solve any of the underlying problems, as still A and B have to establish a situation in which they are in possession of matching keys for the decryption of messages encrypted by the other. Public key cryptosystems like RSA solve this problem by letting each participant have two keys: one public to be used for encryption by anyone who wants to communicate to the owner of this public key; and one secret known only to its owner which is necessary to decrypt any message thus encrypted.

So each participant is responsible for his or her own pair of matching keys, only one of which is made available to the rest of the world. The theoretical problem to be solved, once this idea is formulated, concerns a mechanism according to which such key pairs can be generated such that one key cannot feasibly be computed from the other. Rivest, Shamir and Adleman (1977) devised the following scheme.

We consider simple one-way communication from B to A. Let P_A and S_A be the two keys maintained (and generated) by Alice (A): P_A is made public, S_A is kept secret (remains known only to A). The crucial property is that only with knowledge of S_A can one decrypt a message encrypted with P_A. B obtains the publicly available key P_A to encrypt M and sends this encrypted message to A. Then A, and only A, can restore the original message through decryption with S_A.

Assume that the message is (encoded as) a natural number $M < n$, with a fixed agreed bound n. So the message space is $Z_n = \{0, \ldots n-1\} = \mathbb{Z}_n$. Let T_P and T_S be one-one mappings of $\mathbb{Z}_n$ into itself, associated with keys P and S. In our scenario think of A's public and secret keys $P = P_A$, $S = S_A$. These need to give rise to transformations $T_P, T_S \colon \mathbb{Z}_n \to \mathbb{Z}_n$, which form a matching encryption/decryption pair such that for all M

$$T_S(T_P(M)) = M.$$

According to the above protocol, if A has chosen $S_A = S$ and $P_A = P$, B sends $T_P(M)$ to A; A applies T_S to regain $T_S(T_P(M)) = M$. To ensure secrecy and practicality, one has to find families of matching pairs of transformations T_S and T_P based on key pairs S and P such that

- on the one hand, M cannot be feasibly computed from $T_P(M)$ and P alone,
- on the other hand, both T_P and T_S are feasibly computable with knowledge of the respective keys.

RSA is based on modular exponentiation. With a pair of two large primes p and q (randomly generated by A) associate $n = pq$ and $N := (p-1)(q-1)$. A also chooses some integer e relatively prime to N, i.e., one such that $\gcd(e, N) = 1$. A then computes a solution d to the modular equation

$$ex \equiv_N 1,$$

essentially using EXTENDED EUCLID. A's keys, which are good for messages $M \in \mathbb{Z}_n$, are the pairs of numbers $P = (n, e)$ (the public key) and $S = (n, d)$ (the secret key). The associated transformations in the message space $\mathbb{Z}_n$ are

- T_P based on the *public key* $P = (n, e)$: $\quad T_P(M) = M^e \bmod n$
- T_S based on the *secret key* $S = (n, d)$: $\quad T_S(M) = M^d \bmod n$

Correctness of RSA. To show that $T_S(T_P(M)) = M$, it suffices to show that

$$M^{ed} \equiv_n M.$$

The proof uses Fermat's little theorem, Theorem 10 above, plus some simple modular arithmetic. We only give a sketch here. From $ed \equiv_N 1$ and $N = (p-1)(q-1)$ we find that $ed = 1 + k(p-1)(q-1)$ for suitable $k \in \mathbb{Z}$. Therefore,

$$M^{ed} \equiv_n M M^{(p-1)(q-1)k}.$$

Fermat's theorem, applied for both primes p and q, respectively, implies that

$$M^{ed} \equiv_p M \quad \text{and} \quad M^{ed} \equiv_q M.$$

Thus $M^{ed} = M + k_1 p = M + k_2 q$ for suitable $k_i \in \mathbb{Z}$. Now $k_1 p = k_2 q$ for distinct primes p and q implies that $p|k_2$ and $q|k_1$. Using for instance that $k_2 = k_3 p$ for some $k_3 \in \mathbb{Z}$ we find that

$$M^{ed} \bmod n = (M + k_2 q) \bmod n = (M + k_3 pq) \bmod n = (M + k_3 n) \bmod n = M \bmod n.$$

Remark. The security of RSA rests on the assumption that T_S is not efficiently computable without knowledge of d and in particular that d is not efficiently computable from the publicly available (n, e).

Note that d is computable from n and e to the extent that *factorisation* of composite numbers into their prime factors is computable. From the prime factors p and q of n and knowledge of e, one would obtain N and d just as effectively as Alice.

There is no known polynomial time factorisation algorithm, and factorisation is currently *believed* to be too hard to admit a feasible systematic attack on RSA encryption.