7 RSA Cryptosystem

Addition and multiplication modulo n do not offer the computational difficulties needed to build a viable cryptographic system. We will see that exponentiation modulo n does.

Operations as functions. Recall that $+_n$ and $\cdot_n$ each read two integers and return a third integer. If we fix one of the two input integers, we get two functions. Specifically, fixing $a \in \mathbb{Z}_n$, we have functions $A : \mathbb{Z}_n \to \mathbb{Z}_n$ and $M : \mathbb{Z}_n \to \mathbb{Z}_n$ defined by

$$\begin{aligned} A(x) &= x +_n a; \\ M(x) &= x \cdot_n a; \end{aligned}$$

see Table 4. Clearly, A is injective for every choice of

x	0	1	2	3	4	5
$A(x)$	2	3	4	5	0	1
$M(x)$	0	2	4	0	2	4

Table 4: The function A defined by adding $a = 2$ modulo $n = 6$ is injective. In contrast, the function M defined by multiplying with $a = 2$ is not injective.

$n > 0$ and $a \in \mathbb{Z}_n$. On the other hand, M is injective iff $\gcd(a, n) = 1$. In particular, M is injective for every non-zero $a \in \mathbb{Z}_n$ if n is prime.

Exponentiation. Yet another function we may consider is taking a to the x-th power. Let $E : \mathbb{Z}_n \to \mathbb{Z}_n$ be defined by

$$\begin{aligned} E(x) &= a^x \bmod n \\ &= a \cdot_n a \cdot_n \ldots \cdot_n a, \end{aligned}$$

where we multiply x copies of a together. We see in Table 5 that for some values of a and n, the restriction of E to the non-zero integers is injective and for others it is not. Perhaps surprisingly, the last column of Table 5 consists of 1s only.

FERMAT'S LITTLE THEOREM. Let p be prime. Then $a^{p-1} \bmod p = 1$ for every non-zero $a \in \mathbb{Z}_p$.

PROOF. Since p is prime, multiplication with a gives an injective function for every non-zero $a \in \mathbb{Z}_p$. In other words, multiplying with a permutes the non-zero integers

a^x	0	1	2	3	4	5	6
1	1	1	1	1	1	1	1
2	1	2	4	1	2	4	1
3	1	3	2	6	4	5	1
4	1	4	2	1	4	2	1
5	1	5	4	6	2	3	1
6	1	6	1	6	1	6	1

Table 5: Exponentiation modulo $n = 7$. We write x from left to right and a from top to bottom.

in $\mathbb{Z}_p$. Hence,

$$\begin{aligned} X &= 1 \cdot_p 2 \cdot_p \ldots \cdot_p (p-1) \\ &= (1 \cdot_p a) \cdot_p (2 \cdot_p a) \cdot_p \ldots \cdot_p ((p-1) \cdot_p a) \\ &= X \cdot_p (a^{p-1} \bmod p). \end{aligned}$$

Multiplying with the inverse of X gives $a_{p-1} \bmod p = 1$. □

One-way functions. The RSA cryptosystem is based on the existence of *one-way functions* $f : \mathbb{Z}_n \to \mathbb{Z}_n$ defined by the following three properties:

- f is easy to compute;
- its inverse, $f^{-1} : \mathbb{Z}_n \to \mathbb{Z}_n$, exists;
- without extra information, f^{-1} is hard to compute.

The notions of 'easy' and 'hard' computation have to be made precise, but this is beyond the scope of this course. Roughly, it means that given x, computing $y = f(x)$ takes on the order of a few seconds while computing $f^{-1}(y)$ takes on the order of years. RSA uses the following recipe to construct one-way functions:

1. choose large primes p and q, and let $n = pq$;
2. choose $e \neq 1$ relative prime to $(p-1)(q-1)$ and let d be its multiplicative inverse modulo $(p-1)(q-1)$;
3. the one-way function is defined by $f(x) = x^e \bmod n$ and its inverse is defined by $g(y) = y^d \bmod n$.

According to the RSA protocol, Bob publishes e and n and keeps d private. To exchange a secret message, $x \in \mathbb{Z}_n$,

4. Alice computes $y = f(x)$ and publishes y;
5. Bob reads y and computes $z = g(y)$.

To show that RSA is secure, we would need to prove that without knowing p, q, d, it is hard to compute g. We

leave this to future generations of computer scientists. Indeed, nobody today can prove that computing p and q from $n = pq$ is hard, but then nobody knows to how factor large integers efficiently either.

Correctness. To show that RSA works, we need to prove that $z = x$. In other words, $g(y) = f^{-1}(y)$ for every $y \in \mathbb{Z}_n$. Recall that y is computed as $f(x) = x^e \bmod n$. We need $y^d \bmod n = x$ but we first prove a weaker result.

LEMMA. $y^d \bmod p = x \bmod p$ for every $x \in \mathbb{Z}_n$.

PROOF. Since d is the multiplicative inverse of e modulo $(p-1)(q-1)$, we can write $ed = (p-1)(q-1)k + 1$. Hence,

$$\begin{aligned} y^d \bmod p &= x^{ed} \bmod p \\ &= x^{k(p-1)(q-1)+1} \bmod p. \end{aligned}$$

Suppose first that $x^{k(q-1)} \bmod p \neq 0$. Then Fermat's Little Theorem implies $x^{k(p-1)(q-1)} \bmod p = 1$. But this implies $y^d \bmod p = x \bmod p$, as claimed. Suppose second that $x^{k(q-1)} \bmod p = 0$. Since p is prime, every power of a non-zero integer is non-zero. Hence, $x \bmod p = 0$. But this implies $y^d \bmod p = 0$ and thus $y^d \bmod p = x \bmod p$, as before. □

By symmetry, we also have $y^d \bmod q = x \bmod q$. Hence,

$$\begin{aligned} (y^d - x) \bmod p &= 0; \\ (y^d - x) \bmod q &= 0. \end{aligned}$$

By the Chinese Remainder Theorem, this system of two linear equations has a unique solution in $\mathbb{Z}_n$, where $n = pq$. Since $y^d - x = 0$ is a solution, there can be no other. Hence,

$$(y^d - x) \bmod n = 0.$$

The left hand side can be written as $((y^d \bmod n) - x) \bmod n$. This finally implies $y^d \bmod n = x$, as desired.

Summary. We talked about exponentiation modulo n and proved Fermat's Little Theorem. We then described how RSA uses exponentiation to construct one-way functions, and we proved it correct. A proof that RSA is secure would be nice but is beyond what is currently known.