Definitions and Review

- **Field.** A field is a set $F$ equipped with two operations $+, \times$ such that $S$ forms an abelian (commutative) group under $+$, and $F \setminus \{0\}$ forms an abelian group under $\times$ where '0' is the identity of $+$ (a.k.a the additive identity). Also, multiplication should distribute over addition : $\forall x, y, z \in F, x \times (y + z) = x \times y + x \times z$

- **Polynomial.** Given a field $F$, we can construct the set of polynomials over $F$, denoted by $F[x]$. This is simply the set of expressions of the form $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$, where the $a_i$s are elements of the field $F$.

- **RSA Crash Course.** Let’s say I want to send my credit card number $c$ to Amazon. Amazon creates an asymmetric pair of keys as follows:

  1. Pick two random, distinct large primes $p$ and $q$.
  2. Let $N = pq$. Compute $\phi(N)$ using the formula $\phi(pq) = (p - 1)(q - 1)$.
  3. Pick a random element from $\mathbb{Z}_{\phi(N)}^*$, call it $e$.
  4. Publish $(N, e)$ on the internet. This is the **public key**.
  5. Compute the inverse of $e$ modulo $\mathbb{Z}_{\phi(N)}^*$, and call it $d$. $d$ is Amazon’s **private key**.

Given this setup, I can send $M = c^e \mod N$ to Amazon, and Amazon can recover $c$ using this equality : $M^d \equiv_N c^{ed} \equiv_N c^1$

**RSA Fundamentals**

(a) In step (3), why did we pick $e$ from $\mathbb{Z}_{\phi(N)}^*$?

(b) What prevents an attacker from computing the inverse of $e$ in $\mathbb{Z}_{\phi(N)}^*$ themselves?

(c) Do we know for sure that $c \in \mathbb{Z}_{N}^*$? What if it’s not?

**Inverting RSA**

Suppose I’m communicating with an untrusted server that claims to be Amazon. I want the server to prove that it is indeed Amazon. Come up with a ‘digital signature’ scheme (based on RSA) that will let me verify Amazon’s identity. Note that the underlying assumption is that I trust Amazon’s public key indeed belongs to Amazon and not some imposter.

---

1To see a real world private key, run `cat ~/.ssh/id_rsa` on a Unix system
Interpolation

Find the unique degree-2 polynomial \( f \) over \( \mathbb{Z}_7 \) that satisfies the following:

\[
f(1) = 5, f(2) = 3, f(4) = 1
\]

Fields are Meta

Let \( F \) be \( \mathbb{Z}_7 \) - this is the unique field of size 7, up to isomorphism. Let \( S \) be the set of polynomials over \( F \) with degree at most 2.

(a) What is the size of \( S \)?

(b) Verify that \( S \) is a field under addition and multiplication modulo \( x^3 - 2 \).

#Hashing

A length-compressing hashing function is a function \( f : \{0, 1\}^n \to \{0, 1\}^m \), where \( m < n \). Note that such a function cannot be injective (by the pigeonhole principle), so it has collisions (i.e. \( \exists x, y \in \{0, 1\}^n \) such that \( f(x) = f(y) \)).

Let \( p \) be an \( n \)-bit prime and let \( g \in \mathbb{Z}_p^* \) be a generator of this group. Fix some \( y \in \mathbb{Z}_p^* \). Consider the following hashing function \( h : \{0, 1\}^{n+1} \to \{0, 1\}^n \), given by \( h_{p, g, y}(x, b) = y^b g^x \mod p \). Note that \( x \in \{0, 1\}^n \) and \( b \in \{0, 1\} \), and we interpret \( x \) as a number (an element of \( \mathbb{Z}_p^* \)).

Prove that the problem of efficiently finding collisions for this hash function is at least as hard as the discrete log problem\(^2\).

---

\(^2\)If we can find collisions in \( h_{p, g, y} \) for arbitrary \( p, g, y \), then we can find the discrete log (base \( g \)) of arbitrary elements of \( \mathbb{Z}_p^* \).