Linear Congruences

A congruence of the form \( ax \equiv b \pmod{m} \) where \( a, b, m \) are integers and \( x \) a variable is called a linear congruence.

Given such a linear congruence, often need to answer:

1. Are there any solutions?
2. What are the solutions?

Example: Does \( 8x \equiv 2 \pmod{4} \) have any solutions?

Example: Does \( 8x \equiv 2 \pmod{7} \) have any solutions?

Question: Is there a systematic way to solve linear congruences?

Determining Existence of Solutions

Theorem: The linear congruence \( ax \equiv b \pmod{m} \) has solutions iff \( \gcd(a, m) | b \).

Proof, Part I

If \( ax \equiv b \pmod{m} \) has solutions, then \( \gcd(a, m) | b \).

Proof involves two steps:
1. If \( ax \equiv b \pmod{m} \) has solutions, then \( \gcd(a, m) | b \).
2. If \( \gcd(a, m) | b \), then \( ax \equiv b \pmod{m} \) has solutions.

First prove (1), then (2).

Proof, Part II

If \( \gcd(a, m) | b \), then \( ax \equiv b \pmod{m} \) has solutions.

Let \( d = \gcd(a, m) \) and suppose \( d | b \).

Then, there is a \( k \) such that \( b = dk \).

By earlier theorem, there exist \( s, t \) such that \( d = s \cdot a + t \cdot m \).

Multiply both sides by \( k \): \( dk = a \cdot (sk) + m \cdot (tk) \).

Since \( b = dk \), we have \( b = a \cdot (sk) + m \cdot tk \).

Thus, \( b \equiv a \cdot (sk) \pmod{m} \).

Hence, \( sk \) is a solution.

Examples

- Does \( 5x \equiv 7 \pmod{15} \) have any solutions?
- Does \( 3x \equiv 4 \pmod{7} \) have any solutions?
Finding Solutions

- Can determine existence of solutions, but how to find them?
- **Theorem:** Let \(d = \gcd(a, m) = sa + tm\). If \(d|b\), then the solutions to \(ax \equiv b \pmod{m}\) are given by:
  \[x = \frac{sb}{d} + \frac{m}{d} u \quad \text{where } u \in \mathbb{Z}\]

Example

Let \(d = \gcd(a, m) = sa + tm\). If \(d|b\), then the solutions to \(ax \equiv b \pmod{m}\) are given by:
\[x = \frac{sb}{d} + \frac{m}{d} u \quad \text{where } u \in \mathbb{Z}\]

What are the solutions to the linear congruence \(3x \equiv 4 \pmod{7}\)?

Another Example

Let \(d = \gcd(a, m) = sa + tm\). If \(d|b\), then the solutions to \(ax \equiv b \pmod{m}\) are given by:
\[x = \frac{sb}{d} + \frac{m}{d} u \quad \text{where } u \in \mathbb{Z}\]

What are the solutions to the linear congruence \(3x \equiv 1 \pmod{7}\)?

Inverse Modulo \(m\)

- The inverse of \(a\) modulo \(m\), written \(a\) has the property:
  \[a \equiv 1 \pmod{m}\]
- **Theorem:** Inverse of \(a\) modulo \(m\) exists if and only if \(a\) and \(m\) are relatively prime.
- **Proof:** Inverse must satisfy \(ax \equiv 1 \pmod{m}\)
- Does 3 have an inverse modulo 7?

Example

- Find an inverse of 3 modulo 7.
- An inverse is any solution to \(3x \equiv 1 \pmod{7}\)
- Earlier, we already computed solutions for this equation as:
  \[x = -2 + 7u\]
- Thus, \(-2\) is an inverse of 3 modulo 7
- 5, 12, -9, ... are also inverses

Cryptography

- Cryptography is the study of techniques for secure transmission of information in the presence of adversaries
- How can Alice send secret messages to Bob without Eve being able to read them?
Private vs. Public Crypto Systems

- Two different kinds of cryptography systems:
  1. Private key cryptography (also known as symmetric)
  2. Public key cryptography (asymmetric)

- In private key cryptography, sender and receiver agree on secret key that both use to encrypt/decrypt the message
- In public key cryptography, a public key is used to encrypt the message, and private key is used to decrypt the message

Private Key Cryptography

- Private key crypto is classical method, used since antiquity
- Caesar’s cipher is an example of private key cryptography
- Caesar’s cipher is shift cipher where \( f(p) = (p + k) \mod 26 \)
- Both receiver and sender need to know \( k \) to encrypt/decrypt
- Modern symmetric algorithms: RC4, DES, AES, ...
- Main problem: How do you exchange secret key in a secure way?

Public Key Cryptography

- Public key cryptography is the modern method: different keys are used to encrypt vs. decrypt message
- Most commonly used public key system is RSA
- Great application of number theory and things we’ve learned

RSA History

- Named after its inventors Rivest, Shamir, and Adleman, all researchers at MIT (1978)
- Actually, similar system invented earlier by British researcher Clifford Cocks, but classified – unknown until 90’s

RSA Overview

- Bob has two keys: public and private
- Everyone knows Bob’s public key, but only he knows his private key
- Alice encrypts message using Bob’s public key
- Bob decrypts message using private key
- Since public key cannot decrypt, noone can read message accept Bob

High Level Math Behind RSA

- In the RSA system, private key consists of two very large prime numbers \( p, q \)
- Public key consists of a number \( n \), which is the product of \( p, q \) and another number \( e \), which is relatively prime with \( (p - 1)(q - 1) \)
- Encrypt messages using \( n, e \), but to decrypt, must know \( p, q \)
- In theory, can extract \( p, q \) from \( n \) using prime factorization, but this is intractable for very large numbers
- Security of RSA relies on inherent computational difficulty of prime factorization
Encryption in RSA

- To send message to Bob, Alice first represents message as a sequence of numbers.
- Call this number representing message \( M \).
- Alice then uses Bob’s public key \( n, e \) to perform encryption as:
  \[
  C = M^e \pmod{n}
  \]
- \( C \) is called the ciphertext.

RSA Decryption

- Decryption key \( d \) is the inverse of \( e \) modulo \((p - 1)(q - 1)\):
  \[
  d \cdot e \equiv 1 \pmod{(p - 1)(q - 1)}
  \]
- Decryption function: \( C^d \pmod{n} \).
- As we saw earlier, \( d \) can be computed reasonably efficiently if we know \((p - 1)(q - 1)\).
- However, since adversaries do not know \( p, q \), they cannot compute \( d \) with reasonable computational effort!

Security of RSA

- The encryption function used in RSA is a trapdoor function.
- Trapdoor function is easy to compute in one direction, but very difficult in reverse direction without additional knowledge.
- Decryption without private key is very hard because requires prime factorization (which is intractable for large enough numbers).
- Interesting fact: There are efficient (poly-time) prime factorization algorithms for quantum computers (e.g., Shor’s algorithm).
- If we could build quantum computers with sufficient “qubits”, RSA would no longer be secure!