# COMP3260
# Data Security

# Lecture 2

Prof Ljiljana Brankovic

# Lecture Overview

1. Chinese Remainder Theorem
2. Euclid's algorithm for computing gcd
3. Extended Euclid's algorithm to compute multiplicative inverses
4. Euler's totient function
5. Solving general equations
6. Example
7. Introduction to Information Theory
8. Entropy

# Number Theory and Finite Fields

- Text: Chapter 2 Introduction to Number Theory“ [1]
- Cryptography and Data Security” by D. Denning [2]

Note that in-text references and quotes are omitted for clarity of the slides. When you write as essay or a report it is very important that you use both in-text references and quotes where appropriate.

# Chinese Remainder Theorem

***In 3rd century AD, the Chinese mathematician Sun Tzu (or Sun Zi) asked the following question in his book Sun Tzu Suan Ching (literally, "Sun Tzu's Calculation Classic"):***

"We have a number of things, but we do not know exactly how many. If we count them by threes we have two left over. If we count them by fives we have three left over. If we count them by sevens we have two left over. How many things are there?"

# Chinese Remainder Theorem

***Chinese Remainder Theorem:*** Let $d_1, \ldots, d_t$ be pairwise relatively prime, and let $n=d_1 d_2 \ldots d_t$. Then the system of equations

$(x \bmod d_i) = x_i \ (i = 1, \ldots, t)$

has a common solution $x$ in the range $[0,n-1]$.

Proof Outline: The common solution is

$$x = [\sum_{i=1}^{t} \frac{n}{d_i} y_i x_i] \bmod n$$

where $y_i$ is a solution of $\frac{n}{d_i} y_i \bmod d_i = 1$ (note that $\frac{n}{d_i}$ is relatively prime to $d_i$, thus there is always a solution).

# Chinese Remainder Theorem

***Example 1*:** Solve $3x \bmod 10 = 1$ (in other words, find a mulplicative inverse of $3$ modulo $10$).

$10 = 2 \times 5$ so $d_1 = 2$ and $d_2 = 5$.

We first find solutions $x_1$ and $x_2$:

$3x \bmod 2 = 1 \rightarrow x_1 = 1$

$3x \bmod 5 = 1 \rightarrow x_2 = 2$

We now apply Chinese reminder theorem to find a common solution to the equations

$x \bmod 2 = x_1 = 1$

$x \bmod 5 = x_2 = 2$

# Chinese Remainder Theorem

First find $y_1$ and $y_2$ such that

$$\frac{10}{2} y_1 \bmod 2 = 1 \;\rightarrow\; y_1 = 1$$

$$\frac{10}{5} y_2 \bmod 5 = 1 \rightarrow y_2 = 3$$

We now have

$$x = \left(\frac{10}{2} y_1 x_1 + \frac{10}{5} y_2 x_2\right) \bmod 10$$

$$= \;(5 \times 1 \times 1 + 2 \times 3 \times 2) \bmod 10 = 7$$

Thus *7* is the multiplicative inverse of *3* modulo *10*.

# Chinese Remainder Theorem

***Example 2:*** An old woman goes to market and a horse steps on her basket and crashes the eggs. The rider offers to pay for the damages and asks her how many eggs she had brought. She does not remember the exact number, but when she had taken them out two at a time, there was one egg left. The same happened when she picked them out three, four, five, and six at a time, but when she took them seven at a time they came out even. What is the smallest number of eggs she could have had?

# Chinese Remainder Theorem

$x \bmod 2 = 1$
$x \bmod 3 = 1$
$x \bmod 4 = 1$
$x \bmod 5 = 1$
$x \bmod 6 = 1$
$x \bmod 7 = 0$

Remember that in order to apply the Chinese remainder Theorem, we need $d_1, \ldots, d_t$ to be relatively prime. Is that the case here? Which ones should we keep?

# Chinese Remainder Theorem

Note that *x mod 6 = 1* implies *x mod 2 = 1* but not the other way around! However, *x mod 2 = 1* and *x mod 3 = 1* together imply *x mod 6 = 1* (by the Chinese Remainder Theorem :-)

| x | x mod 2 | x mod 3 | x mod 6 |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 2 | 0 | 2 | 2 |
| 3 | 1 | 0 | 3 |
| 4 | 0 | 1 | 4 |
| 5 | 1 | 2 | 5 |
| 6 | 0 | 0 | 0 |
| 7 | 1 | 1 | 1 |
| 8 | 0 | 2 | 2 |
| 9 | 1 | 0 | 3 |
| 10 | 0 | 1 | 4 |
| 11 | 1 | 2 | 5 |
| 12 | 0 | 0 | 0 |

# Chinese Remainder Theorem

$x \bmod 3 = 1$
$x \bmod 4 = 1$
$x \bmod 5 = 1$
$x \bmod 7 = 0$

$N = 3 \times 4 \times 5 \times 7$

$140\, y_1 \bmod 3 = 1 \rightarrow 2\, y_1 \bmod 3 = 1 \rightarrow y_1 = 2$
$105 y_2 \bmod 4 = 1 \rightarrow \quad y_2 \bmod 4 = 1 \rightarrow y_2 = 1$
$84\, y_3 \bmod 5 = 1 \rightarrow 4\, y_3 \bmod 5 = 1 \rightarrow y_3 = 4$
$60 y_4 \bmod 7 = 1 \rightarrow 4 y_4 \bmod 7 = 1 \rightarrow y_4 = 2$

$x = (140 \times 2 \times 1 + 105 \times 1 \times 1 + 84 \times 4 \times 1 + 60 \times 2 \times 0) \bmod 420$
$\quad = 721 \bmod 420 = 301$

# Euclid's algorithm for computing gcd

Euclud's algorithms is based on the fact that if $a$ and $b$, $a > b$, are both divisible by $c$, so is $a \bmod b$.

Indeed, if we write $a$ as

$$a=kb+r$$

and if both a and b are divisible by $c$, we have

$$hc=klc + r$$

then $r=(h-kl)c$ and thus $r$ is also divisible by $c$.

# Euclid's algorithm for computing gcd

For example, *100* and *22* are both divisible by *2*, and so is *100* mod *22 = 12*.

Therefore, instead of looking for the greatest common divisor (gcd) of *100* and *22*, we can look for the greatest common divisor of *22* and 12; and so on…

*100, 22*

*22, 12*

*12, 10*

*10, 2*

*2, 0* ← when we hit "*0* ", the previous value is the *gcd(a.b)*

# Euclid's algorithm for computing gcd

```
Algorithm gcd(a,n)
//n ≥ a

begin
   g0 := n;
   g1 := a;
   i := 1;
   while gi ≠ 0 do
      begin
         gi+1 := gi-1 mod gi;
         i := i + 1
      end;
   gcd := gi-1
end
```

# Euclid's algorithm extended to compute inverses

```
Algorithm inv(a,n)
begin
   g0 := n; g1 := a; u0 = 1;  v0 := 0;  u1 := 0;  v1 := 1; i := 1;
   while gi ≠ 0 do "gi = ui n + vi a"
     begin
       y := gi-1 div gi;   gi+1 := gi-1 - y × gi; //y:=10 div 4 = 2;
                                                  //gi+1 := 10 - 2×4=2
       ui+1 := ui-1 - y × ui;  vi+1 := vi-1 - y × vi;
       i := i + 1
     end;
   x := vi-1
   if x ≥ 0 then inv := x else inv := x+n
end
```

# Euclid's extended algorithm: $3x \bmod 10 = 1$

```
Algorithm inv(a,n)
begin
    g0 := n; g1 := a; u0 = 1;  v0 := 0;  u1 := 0;

    v1 := 1; i := 1;

    while gi ≠ 0 do "gi = ui n + vi a"
      begin
         y := gi-1 div gi;   gi+1 := gi-1 - y *gi;

         ui+1 := ui-1 - y *ui;  vi+1 := vi-1 - y *vi;

         i := i + 1
      end;

    x := vi-1

    if x ≥ 0 then inv := x else inv := x+n
end
```

| i | y | g | u | v |
|---|---|---|---|---|
| 0 | | 10 | 1 | 0 |
| 1 | | 3 | 0 | 1 |
| 2 | 3 | 1 | 1 | -3 |
| 3 | 3 | 0 | | |

# Euler's Totient Function

***Definitions:***

A positive integer $p$ is a prime number iff $p>1$ and the only positive integer divisors of $p$ are $1$ and $p$.

Two integers $n$ and $m$ are relatively prime iff $gcd(n,m)=1$.

***Fundamental Theorem of Number Theory:***

Every positive integer $n > 1$ can be written uniquely in the form

$$n = p_1^{e_1} p_2^{e_2} \dots p_t^{e_t}$$

where $p_i$ is a prime number and $p_1 < p_2 < \dots < p_t$.

# Euler's Totient Function

- For every integer $n$, the Euler's totient function $\phi(n)$ is the number of positive integers less than or equal to $n$ which are relatively prime to $n$.

- If $n$ is prime then $\phi(n) = n-1$.

- If $n$ is the product of two primes, $n = p \times q$ then $\phi(n) = (p-1) \times (q-1)$.

# Euler's Totient Function

- In general,
  1. $$\phi(1) = 1$$
  2. For $n > 1$, if $$n = p_1^{e_1} p_2^{e_2} \ldots p_t^{e_t}$$

     then $$\phi(n) = \prod_{i=1}^{t} p_i^{e_i - 1}(p_i - 1).$$

Examples:

- $\phi(5) = 5 - 1 = 4$
- $\phi(10) = \phi(5 \times 2) = (5-1) \times (2-1) = 4$
- $\phi(40) = \phi(5 \times 2^3) = (5-1) \times 2^2 \times (2-1) = 4 \times 4 \times 1 = 16$

# Euler's Totient Function

***Fermat's Little Theorem***: Let $p$ be a prime. Then for every $a$ such that $gcd(a,p) = 1$, $a^{p-1} \bmod p = 1$.

***Euler's generalization***: For every $a$ and $n$ such that $gcd(a,n)=1$, $a^{\phi(n)} \bmod n = 1$.

Euler's generalization gives us an algorithm for finding multiplicative inverses, that is, for solving equations of the type $ax \bmod n = 1$. The inverse ($x$) is given by

$$x = a^{\phi(n)-1} \bmod n.$$

$$ax = a\, a^{\phi(n)-1} \bmod n = a^{\phi(n)-1+1} \bmod n = a^{\phi(n)} \bmod n = 1$$

# General equations

***Solving general equations of the form $ax \bmod n = b$:***

When $\gcd(a,n) = 1$, find solution $x_0$ to $ax \bmod n = 1$;
$ax_0 \bmod n = 1$ implies $abx_0 \bmod n = b$ and $x = bx_0 \bmod n$.

When $\gcd(a,n) = g$:

- If $g$ divides $b$, that is, $b \bmod g = 0$, $ax \bmod n = b$ has $g$ solutions of the form
  $x = ((b/g)x_0 + t(n/g)) \bmod n$, for $t=0,1,\ldots,g-1$,
  where $x_0$ is the solution to $(a/g)x \bmod (n/g) = 1$.
- If $g$ does not divide $b$ then there are no solutions.

# General equations

***Example:*** Solve $6x \bmod 10 = 4$.

$g = \gcd(6,10) = 2$ and $2$ divides $4$ so there are $2$ solutions.

Compute $x_0$ from $(6/2)x \bmod (10/2) = 1$. Get $x_0 = 2$.

Now calculate the two solutions:

$t=0$ gives $x = 2(4/2) + 0\,(10/2) \bmod 10 = 4$

$t=1$ gives $x = 2(4/2) + 1(10/2) \bmod 10 = 9$

Check:

$6 \times 4 \bmod 10 = 24 \bmod 10 = 4$

$6 \times 9 \bmod 10 = 54 \bmod 10 = 4$

# Example

Solve $31x \bmod 42 = 1$.

We shall use 3 different techniques to solve this equation and find multiplicative inverse of $31$ modulo $42$.

1. Euclid's extended algorithm for gcd
2. Chinese Remainder Theorem
3. Euler Totient Function

# Euclid's extended algorithm for gcd

```
Algorithm inv(a,n)
begin
    g0 := n; g1 := a; u0 = 1;  v0 := 0;  u1 := 0;

    v1 := 1; i := 1;

    while gi ≠ 0 do "gi = uin + via"
      begin
        y := gi-1 div gi;   gi+1 := gi-1 - y *gi;

        ui+1 := ui-1 - y *ui;  vi+1 := vi-1 - y *vi;

        i := i + 1
      end;

    x := vi-1

    if x ≥ 0 then inv := x else inv := x+n
end
```

| i | y | g | u | v |
|---|---|---|---|---|
| 0 | | 42 | 1 | 0 |
| 1 | | 31 | 0 | 1 |
| 2 | 1 | 11 | 1 | -1 |
| 3 | 2 | 9 | -2 | 3 |
| 4 | 1 | 2 | 3 | -4 |
| 5 | 4 | 1 | -14 | 19 |
| 6 | 2 | 0 | | |

# Chinese Remainder Theorem

$42 = 2 \times 3 \times 7$ so $d_1 = 2$, $d_2 = 3$ and $d_3 = 7$.

We first find solutions $x_1$ and $x_2$:

$31x \bmod 2 = 1 \rightarrow x_1 = 1$

$31x \bmod 3 = 1 \rightarrow x_2 = 1$

$31x \bmod 7 = 1 \rightarrow x_3 = 5$

We now apply Chinese reminder theorem to find a common solution to the equations

$x \bmod 2 = x_1 = 1$

$x \bmod 5 = x_2 = 1$

$x \bmod 7 = x_3 = 5$

# Chinese Remainder Theorem

First find $y_1$ and $y_2$ such that

$(42/2)y_1 \bmod 2 = 1 \rightarrow y_1 = 1$
$(42/3)y_2 \bmod 3 = 1 \rightarrow y_2 = 2$
$(42/7)y_3 \bmod 7 = 1 \rightarrow y_3 = 6$

We now have

$x = (42/2)y_1x_1 + (42/3)y_2x_2 + (42/7)\, y_3x_3 \bmod 42 =$
$= 21 \times 1 \times 1 + 14 \times 1 \times 2 + 6 \times 5 \times 6 \bmod 42 = 19$

Thus *19* is the multiplicative inverse of *31* modulo *42*.

# Euler Totient function

- $42 = 2 \times 3 \times 7$
- $\phi(42) = 1 \times 2 \times 6 = 12$
- $x = a^{\phi(n)-1} \bmod n = 31^{11} \bmod 42$

# Fast Exponentiation

```
Algorithm fastexp(a, z, n)
begin "return x = a^z mod n"
  a1:= a; z1 :=z;  x := 1;
  while z1 ≠ 0 do
    begin
      while z1 mod 2 = 0 do
        begin "square a1 while
                z1 is even"
          z1 := z1 div 2; a1 :=
                (a1*a1) mod n;
        end;
      z1 := z1 - 1; x := (x*a1)
                mod n;
    end;
  fastexp := x;
end
```

| | a | z | x |
|---|---|---|---|
| 0 | 31 | 11 (1011) | 1 |
| 1 | 31 | 10 (1010) | 31 |
| 2 | 37 | 5 (101) | 31 |
| 3 | 37 | 4 (100) | 13 |
| 4 | 25 | 2 (10) | 13 |
| 5 | 37 | 1 (1) | 13 |
| 6 | 37 | 0 (0) | 19 |

# Introduction to Information Theory Article on Claude Shannon

**Claude Shannon, father of information theory, dies at 84**

Murray Hill, N.J. (Feb. 26, 2001) -- Claude Elwood Shannon, the mathematician who laid the foundation of modern information theory while working at Bell labs in the 1940s, died on Saturday. He was 84.

Shannon's theories are as relevant today as they were when he first formulated them. "It was truly visionary thinking," said Arun Netravali, president of Lucent Technologies' Bell labs. "As if assuming that inexpensive, high-speed processing would come to pass, Shannon figured out the upper limits on communication rates. First in telephone channels, then in optical communications, and now in wireless, Shannon has had the utmost value in defining the engineering limits we face."

In 1948 Shannon published his landmark paper
*"A mathematical theory of communication."*

He begins this pioneering paper on information theory by observing that "the fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point." He then proceeds to so thoroughly establish the foundations of information theory that his framework and terminology remain standard.

Shannon's theory was an immediate success with communications engineers and stimulated the technology which led to today's information age.

Another example is Shannon's 1949 paper entitled "*Communication theory of secrecy systems*." This work is now generally credited with transforming cryptography from an art to a science.

Shannon was born in Petoskey, Michigan, on April 30, 1916. He graduated from the University of Michigan in 1936 with bachelor's degrees in mathematics and electrical engineering. In 1940 he earned both a master's degree in electrical engineering and a Ph.D. in mathematics from the Massachusetts Institute of Technology (MIT).

Shannon joined the mathematics department at Bell Labs in 1941 and remained affiliated with the labs until 1972. He became a visiting professor at MIT in 1956, a permanent member of the faculty in 1958, and a professor emeritus in 1978.

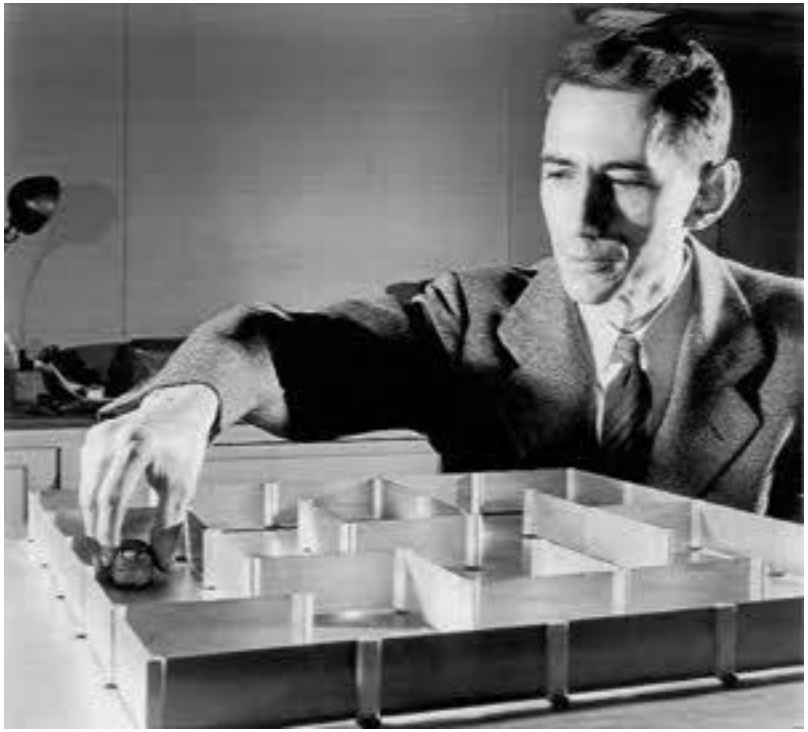
Shannon was renowned for his eclectic interests and capabilities. A favourite story describes him juggling while riding a unicycle down the halls of Bell labs.

He designed and built chess-playing, maze-solving, juggling and mind-reading machines. These activities bear out Shannon's claim that he was more motivated by curiosity than usefulness.

In his words "I just wondered how things were put together."

# Claude Shannon

Claude Shannon's clever electromechanical mouse, which he called Theseus, was one of the earliest attempts to "teach" a machine to "learn" and one of the first experiments in artificial intelligence.



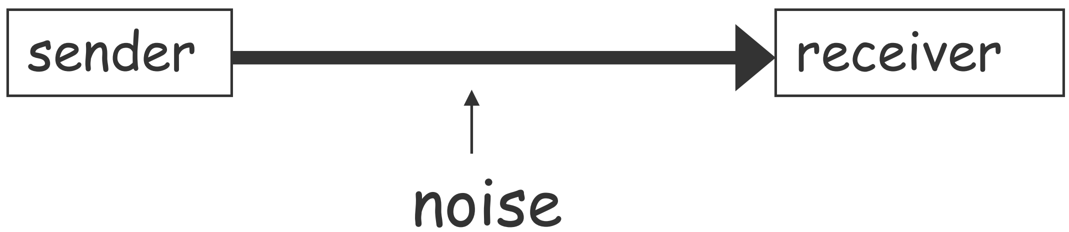
# Information Theory

In 1948 Shannon published a paper "*A* mathematical theory of communication" where he provided a mathematical theory to measure:

1. The **uncertainty** of the receiver of a message passed through a ***noisy channel***

2. The ***secrecy*** of a message passed through an ***encryption channel***

1. The **uncertainty** of the receiver of a message passed through a ***noisy channel***




2. The **secrecy** of a message passed through an ***encryption channel***

The two concepts are closely related.

Both need a measure of the **amount of information** or **entropy** or **randomness** of a message.

# Noisy Channel Problem

In a noisy channel problem, a sender transmits a message M over a noisy channel to a receiver.

If a distorted message M′ is received, the receiver would like to recover the true message M.

To make this possible, the sender adds redundant bits called error control codes to M in such a way that transmission errors can be corrected, or at least detected so that the receiver can request retransmission.

# Analogy Between Noisy Channel and Encryption

- The enciphering transformation corresponds to the noise.
- The ciphertext corresponds to the received message M′.
- The role of the cryptanalyst is similar to the role of the receiver in the noisy channel problem.

# Analogy Between Noisy Channel and Encryption

The role of the sender in the two problems is very different:

- In the noisy channel the objective is to make M directly recoverable from M'.
- In the secrecy problem the objective is to make recovery of M from M' infeasible (without the knowledge of the deciphering key).

# Entropy

Information theory measures the amount of information in a message by the average number of bits needed to encode all possible messages in an optimal encoding.

# Entropy

The **entropy** of a given message *X* is defined by

$$H(X) = -\sum_{i=1}^{n} p(X_i)\log_2 p(X_i) =$$

$$= \sum_{i=1}^{n} p(X_i)\log_2 \frac{1}{p(X_i)}$$

where

- the sum is over all possible messages $X_1, X_2, \ldots, X_n$
- $p(X_i)$ is the probability that the message $X_i$ is sent.

# Entropy

In some sense, the entropy is the minimum size of a compressed version of X.

If the probability distribution is **uniform** then the entropy of the message is the **highest**; the message is random and contains much information.

If the probability distribution is **skewed**, then the entropy of the message is **low**; message is regular and predictable and doesn't tell us much new.

# Entropy

**Note:** each term $\log_2(1/p(X_i))$ represents the number of bits needed to encode message $X_i$ in optimal encoding. The weighted average $H(X)$ gives the expected number of bits in optimally encoded messages.

Because $\log_2(1/p(X_i))$ decreases as $p(X_i)$ increases, an optimal encoding uses short codes for frequently occurring messages and longer codes for infrequent messages. This principle is applied in Morse and Huffman coding.

# Examples

**Example 1.** The messages are about the weather and there are three possible messages: F=fine, S=showers, R=rain.

Each message has the probability of being sent: $p(F)=p(S)=0.25$, $p(R)=0.5$

Note that $\Sigma p(X)$ over all messages X is 1.

The entropy of this set of messages is

$$H(X) = 0.25 \times 2 + 0.25 \times 2 + 0.5 \times 1 = 1.5$$

An optimal encoding assigns 1-bit code to R and 2-bit codes to F and S.

# Example 1

For example, R is encoded as 0, S as 10 and F as 11.

Using this encoding, the 8-letter sequence RSRRFRSF is encoded as the 12-bit sequence 010001101011:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| R | S | R | R | F | R | S | F |
| 0 | 10 | 0 | 0 | 11 | 0 | 10 | 11 |

The average number of bits per letter is 12/8=1.5.

More precisely, we need to factor in the probability here.

So for RFS example we have:

$$0.25\text{x}2 + 0.25\text{x}2 + 0.5\text{X}1 = 1.5 \text{ bits}$$

# Summary

1. Euclid's algorithm for computing gcd

2. Finding multiplicative inverses:
    1. Chinese Remainder Theorem
    2. Extended Euclid's algorithm to compute multiplicative inverses
    3. Euler's totient function

3. Solving general equations
4. Introduction to Information Theory: Entropy

# Next Week

1. Galois fields GF(p) and GF(2n)
2. Entropy
3. Theoretical Secrecy
4. Rate of the Language
5. Redundancy
6. Equivocation
7. Perfect Secrecy
8. One-Time Pad

- Text: Chapter 5 Finite Fields [1]
- "Cryptography and Data Security" by D. Denning [2]

# References


1. W. Stallings. “Cryptography and Network Security”, Pearson, global edition, 2016.

2. D. Denning. “Cryptography and Data Security”, Addison Wesley, 1982.