COM307000 - Cryptography
Public Key Crypto:

Dr. Anca Jurcut
E-mail: `anca.jurcut@ucd.ie`

School of Computer Science and Informatics
University College Dublin,
Ireland

Public Key Cryptography (PKC)

- Probably most significant advance in the history of cryptography

- Developed to address two main issues:
  - **Key Distribution** – how to have secure communications in general without having to trust a Key Distribution Centre with your secret key
  - **Digital Signatures** – how to verify a message comes intact from the claimed sender

Public Key Cryptography(PKC)

- **Two keys, one to encrypt, another to decrypt**
  - Alice uses Bob's **public key** to encrypt
  - Only Bob's **private key** decrypts the message
- **Based on "trap door, one way function"**
  - "One way" means easy to compute in one direction, but hard to compute in other direction
  - Example: Given p and q, product N = pq easy to compute, but hard to find p and q from N
  - "Trap door" is used when creating key pairs

More Examples of “Trap Door, One Way Functions” : Integer Factorization

Factoring: Given n, find all its prime factors

Example: $f(p,q) = n = pq$

- Easy compute: $n = pq$
- Hard: factoring pq into p and q

Example: $f(p,q,e,y) = y^e \bmod pq$

- Easy: $y^e \bmod pq$
- Hard: given pq, e, and $y^e \bmod pq$, compute y’ such that $y'^e = y^e \bmod pq$

Public Key Cryptography (PKC)

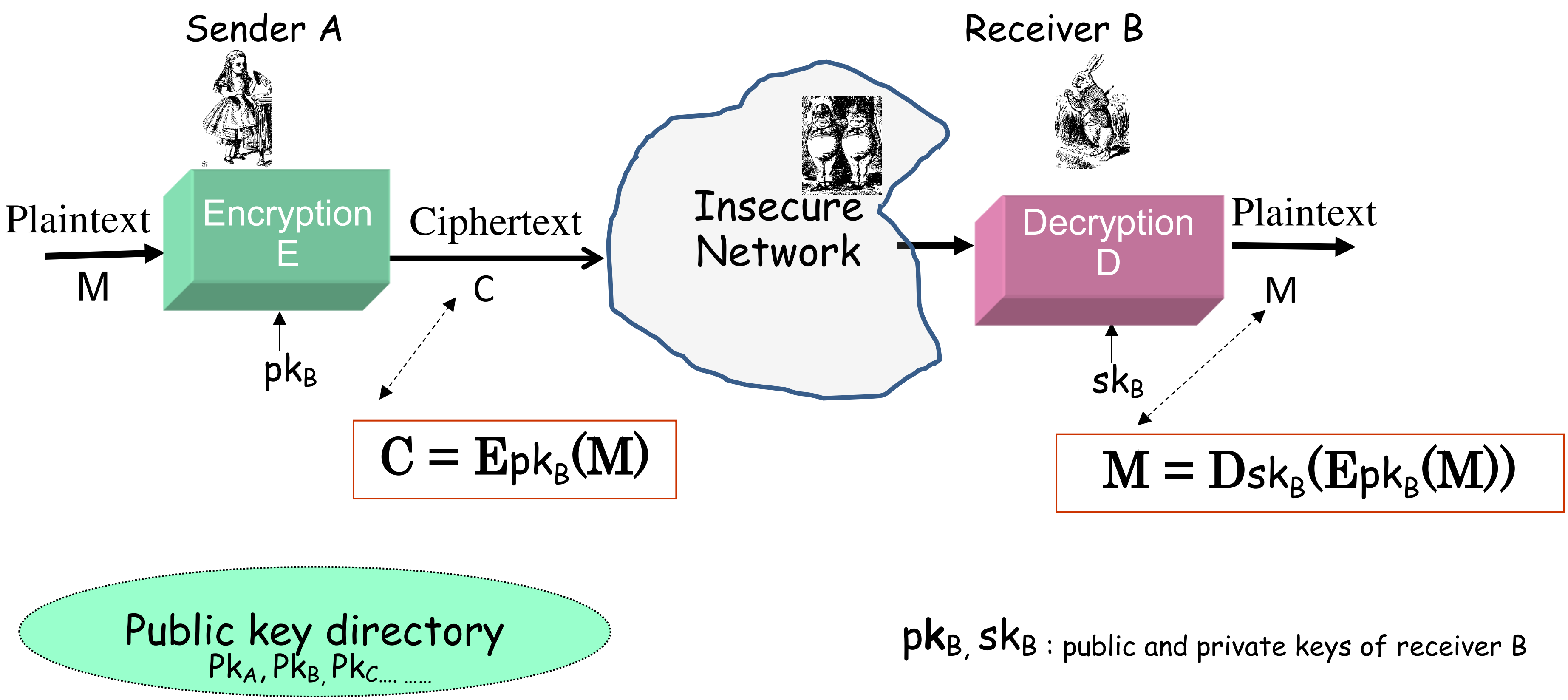
- Each user has 2 keys, public key (**pk**) and private key **(sk)**
  - **Public key pk**
    - ✓ used to **encrypt messages**
    - ✓ used to **verify signatures**
  - **Private key sk**
    - ✓ only known by the owner
    - ✓ used to **decrypt messages**
    - ✓ used to **create signatures**

Public Key Cryptography (PKC)

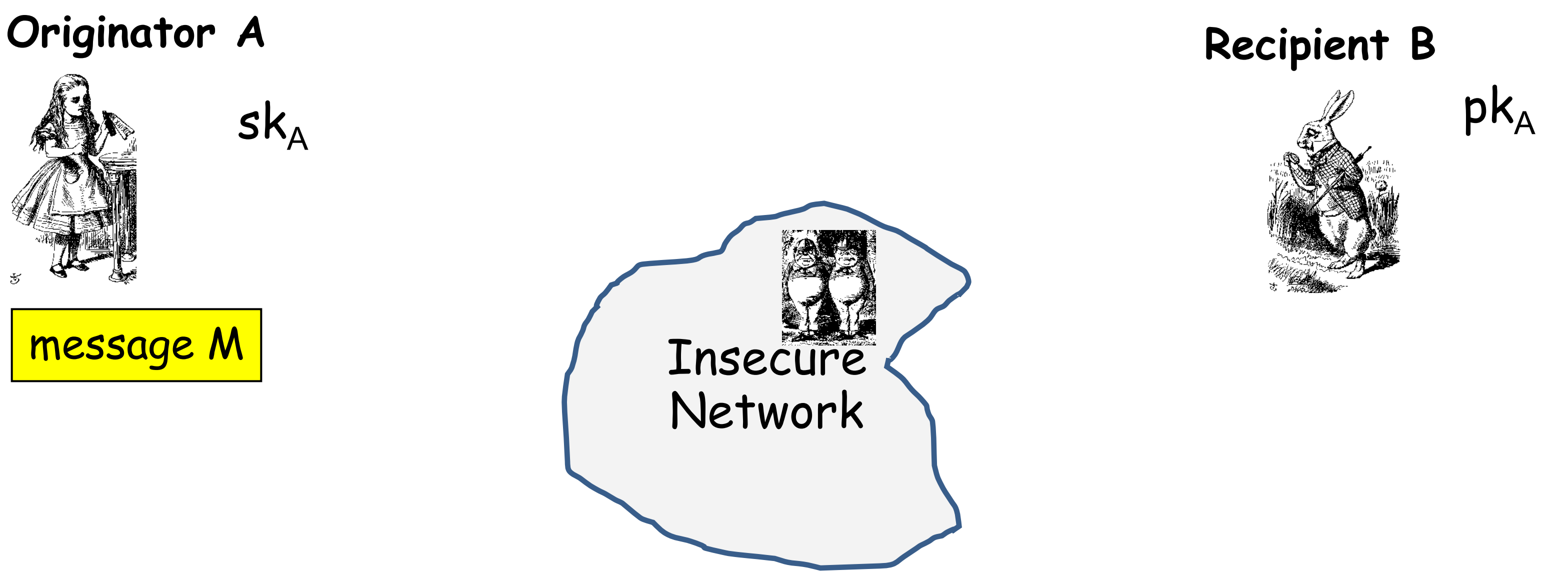
- Encryption
  - Suppose we **encrypt** M with Bob's public key
  - Bob's private key can **decrypt** C to recover M
- Digital Signature
  - Bob **signs** by "encrypting" with his private key
  - Anyone can **verify** signature by "decrypting" with Bob's public key
  - But only Bob could have signed
  - Like a handwritten signature, but much better…

Using PKC: Confidentiality/Secrecy

- Sender encrypts the message using the receiver’s public key

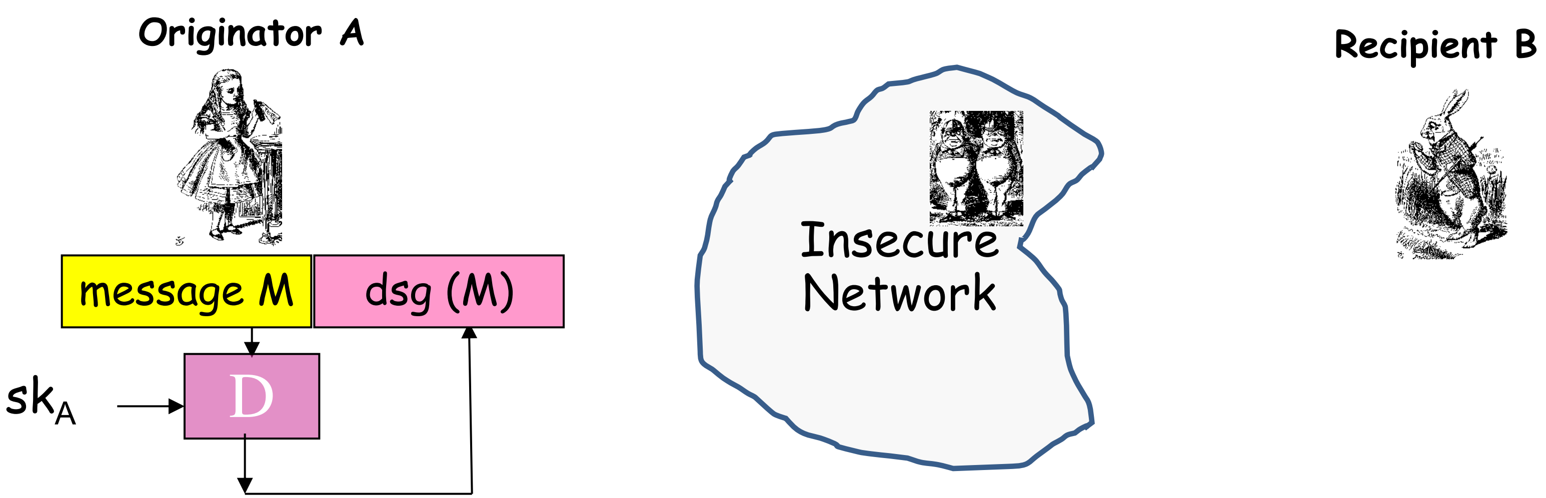


Using PKC: Message Authentication using Digital Signatures



- A wishes to send a signed message M to B
- On receipt B wants to verify the integrity and the originator of the message M

Using PKC: Message Authentication using Digital Signatures



$$dsg(M) = Dsk_A(M)$$

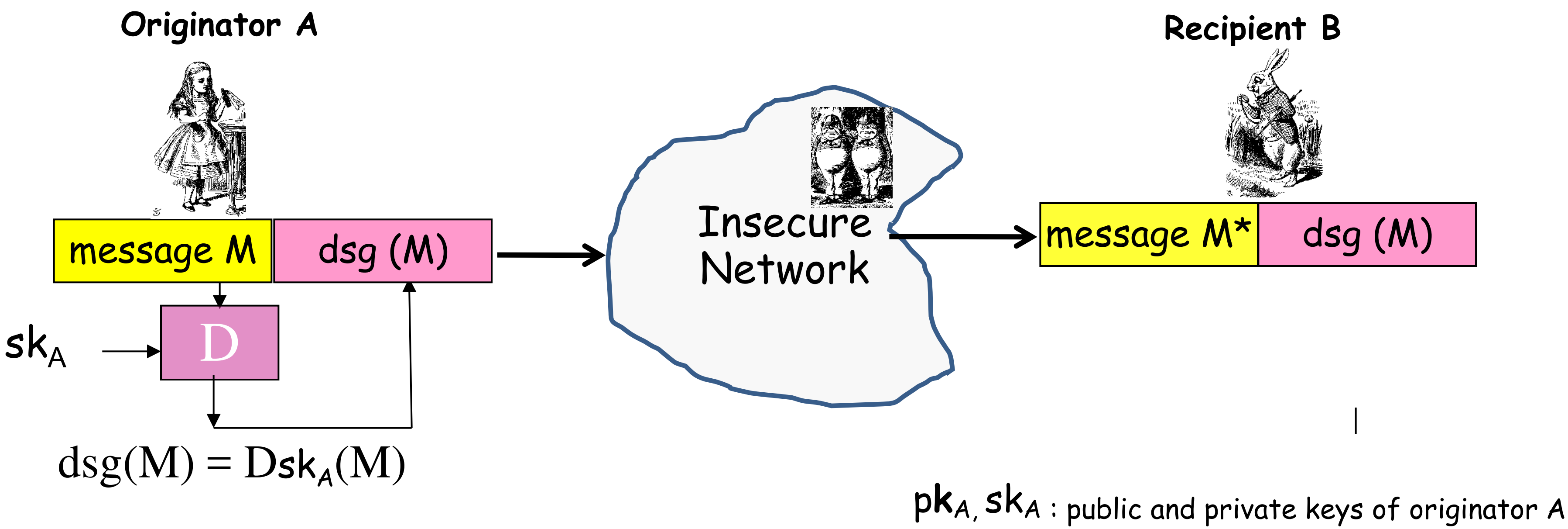
pk_A, sk_A : public and private keys of originator A

Signature Generation

$$\boxed{M,\ Dsk_A(M)}$$

- A signs message M using her private key sk_A

Using PKC: Message Authentication using Digital Signatures

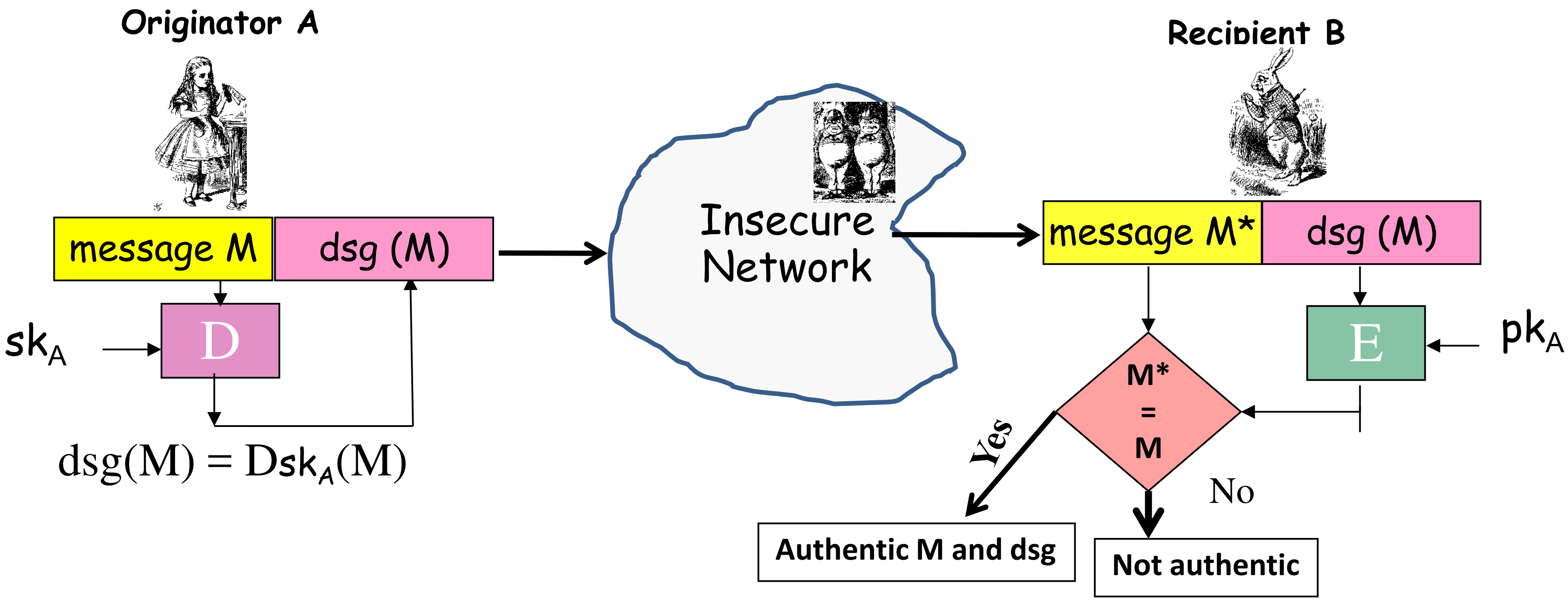


Signature Generation

$$M, \; Dsk_A(M)$$

- A sends signed message to B

Using PKC: Message Authentication using Digital Signatures



Signature Generation

$M, \; Dsk_A(M)$

Signature Verification

verify $M = M^*$,

where $M = Epk_A\,(Dsk_A\,(M))$

pk_A, sk_A : public and private keys of originator A

Public Key Algorithms

- **Knapsack**
- **RSA**
- **Diffie Hellman**
- **Elliptic Curve based Crypto (ECC)**

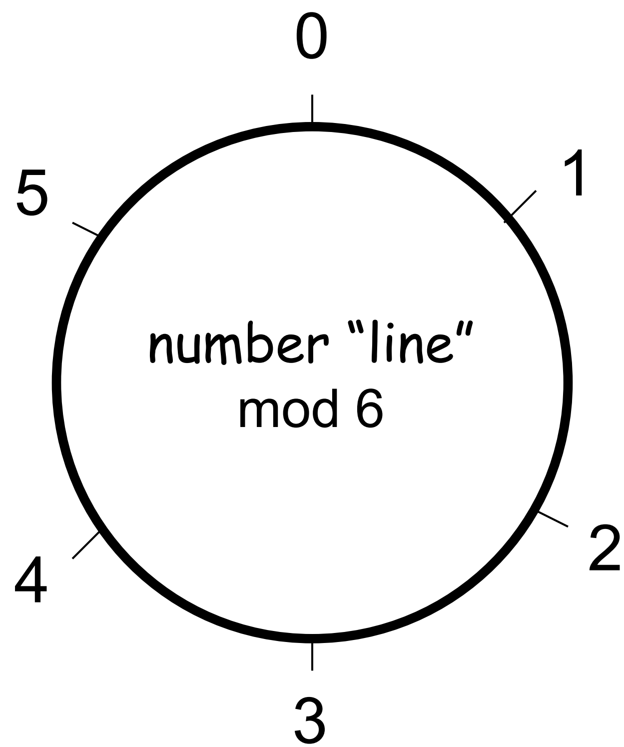
Others: Elgamal, Rabin, Goldwasser-Micali (probanilistic),Blum-Goldwasser(probalistic), Schnorr signature, Zero-Knowledge Algorithms (Fiat-Shamir, Ohta-Okamoto,…)

Appendix: Math Basics

Modular Arithmetic

Clock Arithmetic

- For integers x and n, "x mod n" is the remainder when we compute x ÷ n
  - We can also say "x modulo n"
- Examples
  - 33 mod 6 = 3
  - 33 mod 5 = 3
  - 7 mod 6 = 1
  - 51 mod 17 = 0
  - 17 mod 6 = 5



Modular Addition

- Notation and fun facts
  - 7 mod 6 = 1
  - 7 = 13 = 1 mod 6
  - ((a mod n) + (b mod n)) mod n = (a + b) mod n
  - ((a mod n)(b mod n)) mod n = ab mod n
- Addition Examples
  - 3 + 5 = 2 mod 6
  - 2 + 4 = 0 mod 6
  - 3 + 3 = 0 mod 6
  - (7 + 12) mod 6 = 19 mod 6 = 1 mod 6
  - (7 + 12) mod 6 = (1 + 0) mod 6 = 1 mod 6

Modular Multiplication

- Multiplication Examples
  - $3 \cdot 4 = 0 \bmod 6$
  - $2 \cdot 4 = 2 \bmod 6$
  - $5 \cdot 5 = 1 \bmod 6$
  - $(7 \cdot 4) \bmod 6 = 28 \bmod 6 = 4 \bmod 6$
  - $(7 \cdot 4) \bmod 6 = (1 \cdot 4) \bmod 6 = 4 \bmod 6$

Modular Inverses

- *Additive inverse* of x mod n, denoted $-x \bmod n$, is the number that must be added to x to get 0 mod n
  - $-2 \bmod 6 = 4$, since $2 + 4 = 0 \bmod 6$
- *Multiplicative inverse* of x mod n, denoted $x^{-1} \bmod n$, is the number that must be multiplied by x to get 1 mod n
  - $3^{-1} \bmod 7 = 5$, since $3 \cdot 5 = 1 \bmod 7$

Modular Arithmetic Quiz

- Q: What is -3 mod 6?
- A: 3
- Q: What is -1 mod 6?
- A: 5
- Q: What is 5^{-1} mod 6?
- A: 5
- Q: What is 2^{-1} mod 6?
- A: No number works!
- Multiplicative inverse might not exist

Relative Primality

- x and y are **relatively prime** if they have no common factor other than 1
- x^{-1} mod y exists only when x and y are relatively prime
- If it exists, x^{-1} mod y is easy to compute using Euclidean Algorithm
  - We won't do the computation here
  - But, an efficient algorithm exists

Totient Function

- $\varphi(n)$ is "the number of numbers less than n that are relatively prime to n"
  - Here, "numbers" are positive integers
- Examples
  - $\varphi(4) = 2$ since 4 is relatively prime to 3 and 1
  - $\varphi(5) = 4$ since 5 is relatively prime to 1,2,3,4
  - $\varphi(12) = 4$
  - $\varphi(p) = p-1$ if p is prime
  - $\varphi(pq) = (p-1)(q-1)$ if p and q prime

Knapsack

Knapsack Problem

- Given a set of n weights $W_0, W_1, \ldots, W_{n-1}$ and a sum S, find $a_i \in \{0,1\}$ so that

  $$S = a_0 W_0 + a_1 W_1 + \ldots + a_{n-1} W_{n-1}$$

  (technically, this is the *subset sum* problem)
- **Example**
  - Weights (62,93,26,52,166,48,91,141)
  - Problem: Find a subset that sums to S = 302
  - Answer: 62 + 26 + 166 + 48 = 302
- The (general) knapsack is NP-complete

Knapsack Problem

- **General knapsack** (GK) is hard to solve
- But **superincreasing knapsack** (SIK) is easy
- SIK — each weight greater than the *sum of all previous weights*
- **Example**
  - Weights (2,3,7,14,30,57,120,251)
  - Problem: Find subset that sums to S = 186
  - Work from largest to smallest weight
  - Answer: 120 + 57 + 7 + 2 = 186

Knapsack Cryptosystem

1. Generate superincreasing knapsack (SIK)
2. Convert SIK to “general” knapsack (GK)
3. **Public Key:** GK
4. **Private Key:** SIK and conversion factor

- Goal…
  - Easy to encrypt with GK
  - With private key, easy to decrypt (solve SIK)
  - Without private key, Trudy has no choice but to try to solve GK

Example

- Start with (2,3,7,14,30,57,120,251) as the SIK
- Choose m = 41 and n = 491 (m, n relatively prime, n exceeds sum of elements in SIK)
- Compute "general" knapsack

  $2 \cdot 41 \bmod 491 = 82$

  $3 \cdot 41 \bmod 491 = 123$

  $7 \cdot 41 \bmod 491 = 287$

  $14 \cdot 41 \bmod 491 = 83$

  $30 \cdot 41 \bmod 491 = 248$

  $57 \cdot 41 \bmod 491 = 373$

  $120 \cdot 41 \bmod 491 = 10$

  $251 \cdot 41 \bmod 491 = 471$

- **"General" knapsack:**
  (82,123,287,83,248,373,10,471)

Knapsack Example

- **Private key:** (2,3,7,14,30,57,120,251)
  $m^{-1} \bmod n = 41^{-1} \bmod 491 = 12$
- **Public key:** (82,123,287,83,248,373,10,471), n=491
- Example: Encrypt 10010110
  $82 + 83 + 373 + 10 = 548$
- To decrypt, use private key…
  - $548 \cdot 12 = 193 \bmod 491$
  - Solve (easy) SIK with S = 193
  - Obtain plaintext 10010110

Knapsack Weakness

- **Trapdoor:** Convert SIK into "general" knapsack using modular arithmetic
- **One-way:** General knapsack easy to encrypt, hard to solve; SIK easy to solve
- This knapsack cryptosystem is **insecure**
  - Broken in 1983 with Apple II computer
  - The attack uses **lattice reduction**
- "General knapsack" is not general enough!
  - This special case of knapsack is easy to break

RSA