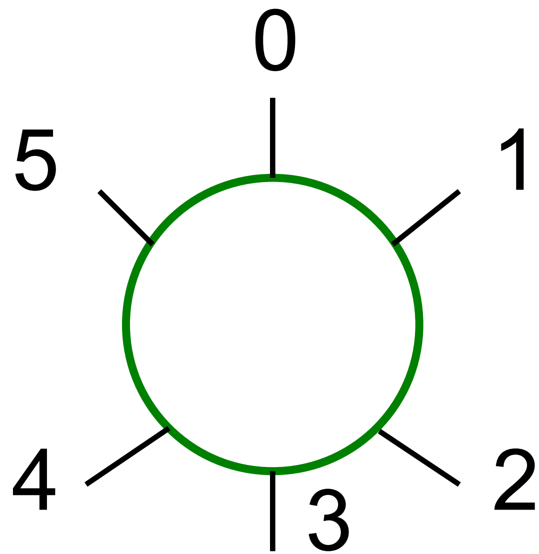
Public Key Crypto and RSA

Modular Arithmetic

Modular addition



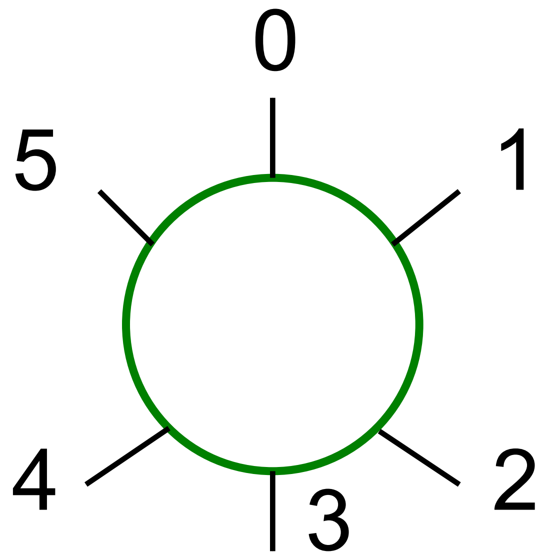
Mod 6

- Modular addition:

  ((a **mod** n) + ((b **mod** n)) **mod** n = (a + b) **mod** n

((4 mod 6) + (5 mod 6)) = ? mod 6

Modular multiplication



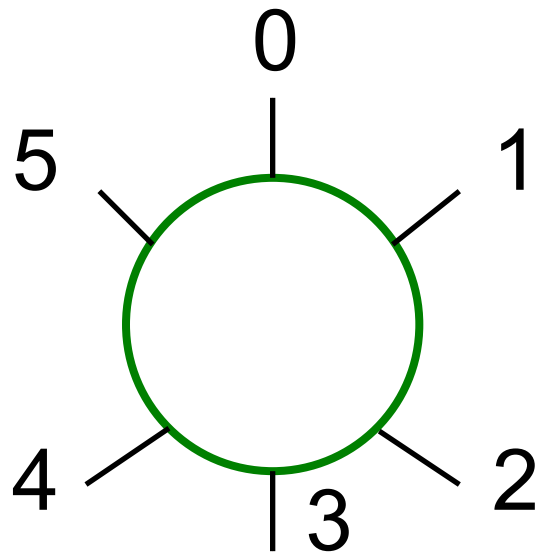
Mod 6

- Modular multiplication:

((a **mod** n) x ((b **mod** n)) **mod** n = (a b) **mod** n

((4 mod 6) x (5 mod 6)) = ? mod 6

Modular additive inverse

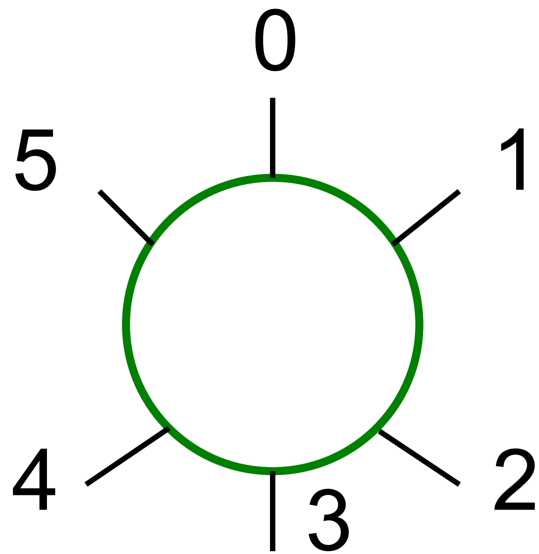


Mod 6

- Modular additive inverse of x: -x mod n
- Result: It is the number a such that $(a + x) \equiv 0 \pmod{n}$

- 4 mod 6 = ? mod 6

Modular multiplicative inverse



Mod 6

- Modular multiplicative inverse of x: x^{-1} mod n.
- Result: It is the number *a* such that $ax \equiv 1 \pmod{n}$

$5^{-1} \bmod 6 = ? \bmod 6$

Existence of multiplicative inverse

- Two numbers are relatively prime if they have no common factor other than 1
  - Are 2 and 3 relatively prime?
  - Are 5 and 10 relatively prime?

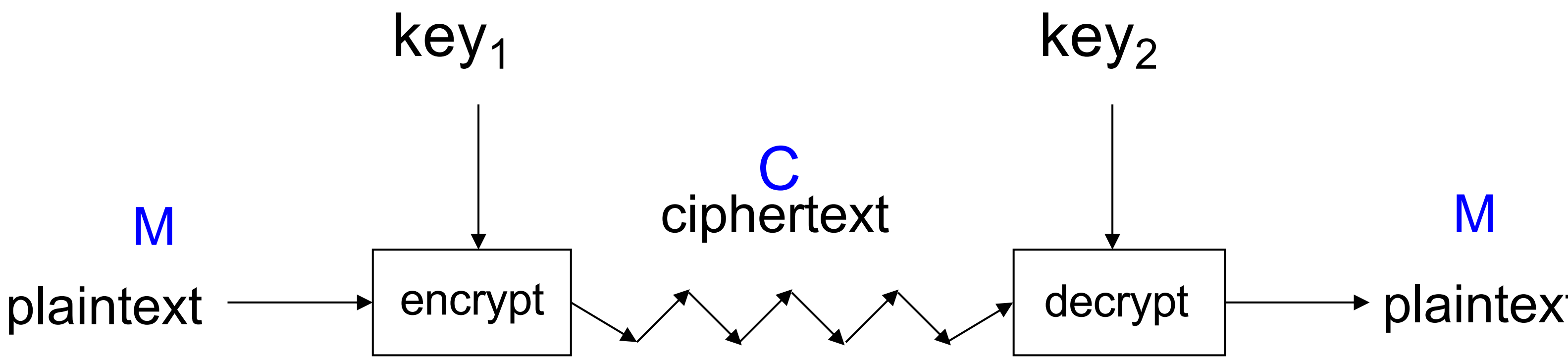
- X^{-1} mod y exists if and only if x and y are relatively prime

Public Key Cryptography

Public key crypto - History

- Invented by cryptographers working for GCHQ (the British equivalent of NSA) in the late 1960s and early 1970s
- Public key crypto was pushed into the limelight by the academicians, and thus revolutionized the crypto field ever since
- Relatively few public key systems are known, and even fewer are widely used
- Each public key system is based on a special mathematical structure, making it extraordinarily difficult to develop new systems

Public Key Cryptography



- A key pair is used in public key cryptography

  <Public Key K_{pub}, Private key K_{pri}>

- Encrypt with key$_1$ = K_{pub}, decrypt with key$_2$ = K_{pri}
- Encrypt with key$_1$ = K_{pri}, decrypt with key$_2$ = K_{pub}

BINGHAMTON UNIVERSITY STATE UNIVERSITY OF NEW YORK

Crux of public key cryptography

- "trap door one way function"
  - "**One way**" means easy to compute in one direction, but hard to compute in other direction
  - "Trap door" used to create key pairs

For instance…

- **Question 1**: Can you factor 247 into the product of two prime numbers quickly?

- **Question 2**: Given two prime numbers 13 and 19, what's their product?

- Which question is more difficult to solve?

- **Reason**: Given two prime numbers p and q, product $N = pq$ easy to compute, but given N, it's hard to find p and q

Quiz - Check all answers that apply

- Which of the following are correct about DES?
  - DES is a Feistel cipher
  - DES is a block cipher
  - DES is a stream cipher
  - The security of DES mainly depends on its S-boxes because of their non-linear transformation

Quiz

- What is the primary advantage of using the Feistel structure in the design of block ciphers?
  - It requires the decryption process to use a different algorithm than encryption.
  - In every round in Feistel cipher, the subkey is exactly the same as the key of the cipher
  - The similar structure of encryption and decryption makes Feistel cipher efficiently implementable
  - Feistel cipher refers to a specific cipher designed by Horst Feistel

Quiz

- Which of the following is correct about the key size of 3DES?
  - The same as the key size of DES
  - Two times the key size of DES
  - Three times the key size of DES
  - Four times the key size of DES

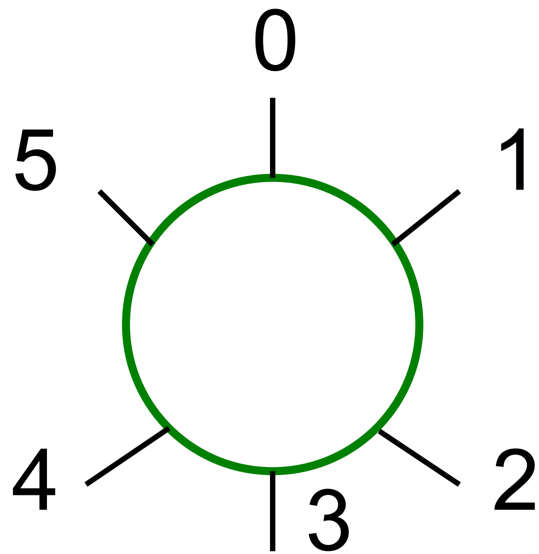
Quiz

- Which is true about CBC decryption?
  - P[i+1] = D(C[i+1])
  - P[i+1] = D(C[i+1] XOR C[i])
  - P[i+1] = D(C[i+1]) XOR C[i]
  - P[i+1] = D(C[i+1]) XOR P[i]

Quiz - Check all answers that apply

- In a block cipher with CBC mode, suppose C[1] is garbled to G, which plaintext block is lost?
  - P[0]
  - P[1]
  - P[2]
  - P[3]

Recall - Modular multiplicative inverse



Mod 6

- Modular multiplicative inverse of x: x^{-1} mod n.
- Result: It is the number *a* such that $a \cdot x \equiv 1 \pmod{n}$
- “≡” is not “=“: `a ≡ b mod n` means `a mod n = b mod n`
- $(x \cdot x^{-1}) \bmod n = 1 \bmod n$

Knapsack Cryptosystem

Knapsack Problem

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

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

  Example

  - Weights $(62, 93, 26, 52, 166, 48, 91, 141)$
  - Problem: Find subset that sums to $S = 302$
  - Answer: $62 + 26 + 166 + 48 = 302$

Knapsack Problem

- **General knapsack** (GK) is hard to solve
- But **superincreasing knapsack** (SIK) is easy
- SIK: each weight is **greater than** the *sum of all previous weights*

$$W_0, W_1, \ldots, W_i, \ldots, W_{n-1}$$

Superincreasing: $W_0 + \ldots + W_{i-1} < W_i$

SIK Example

- Weights (2, 3, 7, 14, 30, 57, 120, 251)
- Problem: Find subset that sums to S=186
- Solution?
  - **251** must not be selected because 251 > 186
  - **120** has to be selected because 120 < 186
  - **57** has to be selected because 57 < 66 (186 – 120)
  - **30** can’t be selected because 30 > 9 (66 – 57)
  - **14** can’t be selected because 14 > 9
  - **7** has to be selected because 7 < 9
  - **3** can’t be selected because 3 > 2 (9 – 7)
  - **2** is selected because 2 = 2
- Final answer: 120+57+7+2=186

Knapsack Cryptosystem

1. Generate superincreasing knapsack (SIK)
2. Convert SIK into "general" knapsack (GK)
3. **Public Key:** GK
4. **Private Key:** SIK + conversion factor

- Ideally…
  - Easy to encrypt with GK
  - With private key, easy to decrypt (convert ciphertext to SIK problem)
  - Without private key, must solve GK

Knapsack Keys

- 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: **modular multiplication**

$2 \cdot 41 \bmod 491 = 82$

$3 \cdot 41 \bmod 491 = 123$

$7 \cdot 41 \bmod 491 = 287$

$14 \cdot 41 \bmod 491 = 83$

$30 \cdot 41 \bmod 491 = 248$

$57 \cdot 41 \bmod 491 = 373$

$120 \cdot 41 \bmod 491 = 10$

$251 \cdot 41 \bmod 491 = 471$

- “General” knapsack: $(82, 123, 287, 83, 248, 373, 10, 471)$

Knapsack Cryptosystem

- **Private key:**
  - (2, 3, 7, 14, 30, 57, 120, 251)
  - Multiplier m = 41
  - Modulus n = 491
- **Public key:**
  - (82, 123, 287, 83, 248, 373, 10, 471)

Knapsack Crypto Example

Encrypt with public key, decrypt with private key

Public key: (82, 123, 287, 83, 248, 373, 10, 471)

- Example: Encrypt plaintext 10010110

Ciphertext: 82 + 83 + 373 + 10 = **548**

(**82**, 123, 287, **83**, 248, **373**, **10**, 471)

$\uparrow$ 1 0 0 $\uparrow$ 1 0 $\uparrow$ 1 $\uparrow$ 1 0

Knapsack Crypto Example

Encrypt with public key, decrypt with private key

Ciphertext: 548

Private key:

(2, 3, 7, 14, 30, 57, 120, 251)
Multiplier m = 41
Modulus n = 491

- To decrypt,
  - $548 \cdot 41^{-1} = 548 \cdot 12 = 193 \bmod 491$
  - Solve (easy) SIK with S = 193

(**2**, 3, 7, **14**, 30, **57**, **120**, 251)

↑ ↑ ↑ ↑

1 0 0 1 0 1 1 0

Why this works?

- $548 \cdot 41^{-1} \bmod 491$

$$= (82 + 83 + 373 + 10) \cdot 41^{-1} \bmod 491$$

$$= (2 \cdot 41 + 14 \cdot 41 + 57 \cdot 41 + 120 \cdot 41) \cdot 41^{-1} \bmod 491$$

$$= (2 + 14 + 57 + 120) \bmod 491$$

$$= 193 \bmod 491$$

Hence, modular multiplicative inverse (multiplying by $m^{-1} \bmod n$) converts the general knapsack problem back to the superincreasing knapsack problem, which is easily solvable

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**
- <u>"General knapsack" converted from SIK is not general enough</u>, and this special knapsack is easy to solve with lattice reduction!

RSA

RSA

- Was developed by **Clifford Cocks** (GCHQ – equivalent of NSA) in 1973 but was not declassified until 1997. But even in the classified crypto circle, it was only a curiosity rather than a practical system
- Was publicly described by **R**ivest, **S**hamir, and **A**dleman (MIT) in 1977
- RSA is the ***gold standard*** in public key crypto

Trapdoor key generation

- Let p and q be two large prime numbers
  - A **prime** number has no positive divisors other than 1 and itself
- Let $N = p \cdot q$ be the **modulus**
- Choose e **relatively prime** to $(p-1) \cdot (q-1)$
- Find d such that $\boldsymbol{d \cdot e = 1 \bmod (p-1) \cdot (q-1)}$
  - So d is the *multiplicative inverse* of e in the *ring* of integers modulo (p-1) ·(q-1). Recall d must exist!
- **Public key** is (N, e)
- **Private key** is d

BINGHAMTON UNIVERSITY STATE UNIVERSITY OF NEW YORK

RSA encryption and decryption

- Message M is treated as a *number* in [0, N)

- To encrypt M with public key we compute

  $C = M^e \bmod N$

- To decrypt ciphertext C with private key compute

  $M = C^d \bmod N$

Public key is ***(N, e)***, **private key** is ***d***

Simple RSA Example – Find Keys

- Example of RSA
  - Select "large" primes $p = 11,\ q = 3$
  - Then $N = \ p \cdot q = 33$ and $(p-1) \cdot (q-1) = 20$
  - Choose $e = 3$ (relatively prime to 20)
  - Find d such that $e \cdot d = 1 \bmod 20$
    - We find that $d = 7$ works
- **Public key:** $(N, e) = (33, 3)$
- **Private key:** $d = 7$

Simple RSA Example – Encryption, Decryption

- **Public key:** $(N, e) = (33, 3)$
- **Private key:** $d = 7$
- Suppose message $M = 8$
- Ciphertext C is computed as

$$C = M^e \bmod N = 8^3 = 512 = \mathbf{17} \bmod 33$$

- Decrypt C to recover the message M by

$$M = C^d \bmod N = 17^7$$

$$= 410{,}338{,}673$$

$$= 12{,}434{,}505 * 33 + 8 = \mathbf{8} \bmod 33$$

Primes should be large

- In the example, the two prime numbers p and q are not large enough. In the real world, N is typically at least 1024 bits, with 2048-bit or large modulus often used.

Real-world example: OpenSSL

```
~
❯ openssl genrsa -out key.pem 1024

~
❯ cat key.pem
```

(PEM: Privacy Enhanced Mail, base64**)**


```
-----BEGIN PRIVATE KEY-----
MIICdwIBADANBgkqhkiG9w0BAQEFAASCAmEwggJdAgEAAoGBAMxoHmD2qYPVfAx3
f9zrrvEb1uylREJHa8ncNSrcpX1Kdlxkev0s3n1SS0XDVq7chcW20JflF5T5RKW0
RONNYvlhIeghqp/jco2dARDRVBhSJZz5ossNg0P9Tid5aIigLfpapeIwrljXxqkB
oWy/yqVzLHedFCWzKX5yOI080IkHAgMBAAECgYABPpUIxBeuHMufi860ep7bCu9Z
C3yJ5sNqPDP6qdM8Gwrzbw0so2xLWQfqSdEqSV9rH7zPX+6v0oCvfZR5ycvTdA+6
WtF4p25PcP6MAhs6njqSRNsocUrXdQ2qUXE+p58tCD8JyuDgujElQToa3gCuanAC
o5DXjUl4LKRk43olKQJBAPcup92H9w63/vfJAuP270+zFizIpNecajTK62UNYBU7
owd9ts0e4bfQnTgX2dPoucbbhaT3zei4MkmW31TejS0CQQDTstLX0N9uGFA23t6r
xKKe4JAmZxxMLPI7R/gAGu+GkGXPlTeiAex+W8ZBSVpkJvIDTQvnwxWfcQBgulyB
RFeDAkEAn8obXkxM28j6HDhnk/LH1V/SD/VNCszko2giL9srp847n9YW3BcAl5FW
cTKJ8EFcBz9V78T56V1ZtNTBXt3XqQJAK5ueiw5fuBISE/t86u0qgofHqeF7lsV7
cHK2x27FAHcmQch/GURELxNAl5pAoHjVSZDJbwhkn99rMIGzJH2reQJBAPGG6vwp
1ZDU9WuekKAG0gT4zDwmWMb5F1Kh0cOHdR+BcBNMmVI9yh607aa2TXE3KsLnQuyw
GoGMCgBOU6lLAkk=
-----END PRIVATE KEY-----
```


BINGHAMTON
UNIVERSITY
STATE UNIVERSITY OF NEW YORK

What are the public key: (N, e) and private key: d?

```
~
❯ openssl rsa -in key.pem -text -noout | head -n 34
Private-Key: (1024 bit, 2 primes)
```

N

```
modulus:
    00:cc:68:1e:60:f6:a9:83:d5:7c:0c:77:7f:dc:eb:
    ae:f1:1b:d6:ec:a5:44:42:47:6b:c9:dc:35:2a:dc:
    a5:7d:4a:76:5c:64:7a:fd:2c:de:7d:52:4b:45:c3:
    56:ae:dc:85:c5:b6:d0:97:e5:17:94:f9:44:a5:b4:
    44:e3:4d:62:f9:61:21:e8:21:aa:9f:e3:72:8d:9d:
    01:10:d1:54:18:52:25:9c:f9:a2:cb:0d:83:43:fd:
    4e:27:79:68:88:a0:2d:fa:5a:a5:e2:30:ae:58:d7:
    c6:a9:01:a1:6c:bf:ca:a5:73:2c:77:9d:14:25:b3:
    29:7e:72:38:8d:3c:d0:89:07
```

e

```
publicExponent: 65537 (0x10001)
```

d

```
privateExponent:
    01:3e:95:08:c4:17:ae:1c:cb:9f:8b:ce:b4:7a:9e:
    db:0a:ef:59:0b:7c:89:e6:c3:6a:3c:33:fa:a9:d3:
    3c:1b:0a:f3:6f:0d:2c:a3:6c:4b:59:07:ea:49:d1:
    2a:49:5f:6b:1f:bc:cf:5f:ee:af:d2:80:af:7d:94:
    79:c9:cb:d3:74:0f:ba:5a:d1:78:a7:6e:4f:70:fe:
    8c:02:1b:3a:9e:3a:92:44:db:28:71:4a:d7:75:0d:
    aa:51:71:3e:a7:9f:2d:08:3f:09:ca:e0:e0:ba:31:
    25:41:3a:1a:de:00:ae:6a:70:02:a3:90:d7:8d:49:
    78:2c:a4:64:e3:7a:25:29
```

p

```
prime1:
    00:f7:2e:a7:dd:87:f7:0e:b7:fe:f7:c9:02:e3:f6:
    ef:4f:b3:16:2c:c8:a4:d7:9c:6a:34:ca:eb:65:0d:
    60:15:3b:a3:07:7d:b6:cd:1e:e1:b7:d0:9d:38:17:
    d9:d3:e8:b9:c6:db:85:a4:f7:cd:e8:b8:32:49:96:
    df:54:de:8d:2d
```

q

```
prime2:
    00:d3:b2:d2:d7:d0:df:6e:18:50:36:de:de:ab:c4:
    a2:9e:e0:90:26:67:1c:4c:2c:f2:3b:47:f8:00:1a:
    ef:86:90:65:cf:95:37:a2:01:ec:7e:5b:c6:41:49:
    5a:64:26:f2:03:4d:0b:e7:c3:15:9f:71:00:60:ba:
    5c:81:44:57:83
```

Primes should be large (cont.)

- In the simple example, the two prime numbers p and q are not large enough(11, 3). In the real world, N is typically at least 1024 bits, with 2048-bit or large modulus often used.

$$\text{Encryption: } C = M^e \bmod N$$
$$\text{Decryption: } M = C^d \bmod N$$

Efficient RSA

- Modular exponentiation example
  - $5^{20} = 95367431640625 = 25 \bmod 35$
- A better way: **repeated squaring**
  - $20 = 10100$ base 2
  - $(1, 10, 101, 1010, 10100) = (1, 2, 5, 10, 20)$
  - Note that $2 = 1 \cdot 2$, $5 = 2 \cdot 2 + 1$, $10 = 2 \cdot 5$, $20 = 2 \cdot 10$
  - $5^1 = 5^0 \cdot 5 \bmod 35$
  - $5^2 = (5^1)^2 = 5^2 = 25 \bmod 35$
  - $5^5 = (5^2)^2 \cdot 5^1 = 25^2 \cdot 5 = 3125 = 10 \bmod 35$
  - $5^{10} = (5^5)^2 = 10^2 = 100 = 30 \bmod 35$
  - $5^{20} = (5^{10})^2 = 30^2 = 900 = 25 \bmod 35$
- No huge numbers and it's efficient!

Factoring breaks RSA

- Suppose Trudy can factor $N = p \cdot q$:
  - With public key (N, e), she obtains p and q
  - She can use e to easily find private key d since

    $$e \cdot d = 1 \bmod (p-1) \cdot (q-1)$$
- How difficult is factoring?
  - Factoring a 768 bit number could be done within a span of two years using hundreds of machines

Forward search attack

- Suppose (e, N, C) is given to Trudy, who wishes to find M.

- If the *plaintext space* **M** = $\{M_1, M_2, \ldots, M_k\}$ is small or predictable, Trudy can decrypt C by simply encrypting all possible plaintext messages with the public key to get $C'_1, C'_2, \ldots,$ and C'_k, where

$$C'_i = M^e \pmod N$$

- Trudy checks if there is any i, $C'_i = C$. If so, he knows that $M = M_i$.

Solution to these attacks: padding

- Add some random bits to the beginning or the end of the message M before encryption, so the padded message M is large.

- After decryption, the padded bits are removed.

- There are standard padding schemes, which will be not covered in this course.
  - PKCS#1v1.5
  - Optimal Asymmetric Encryption Padding (OAEP)