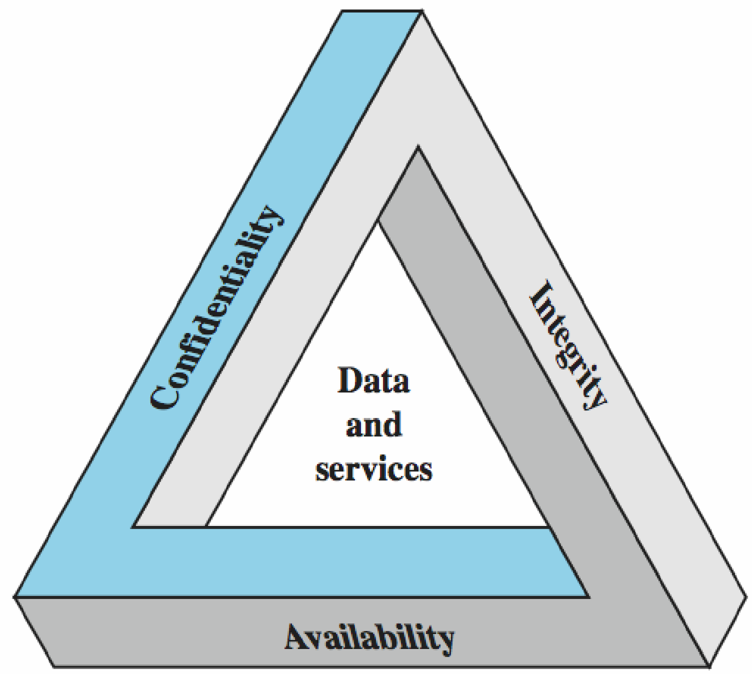
Number Theory and RSA Public-Key Encryption

Dr. Natarajan Meghanathan
Associate Professor of Computer Science
Jackson State University
E-mail: natarajan.meghanathan@jsums.edu

CIA Triad: Three Fundamental Concepts of Information Security

- Confidentiality – Preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information
- Integrity – Guarding against improper information modification or destruction, and includes ensuring
  - information non-repudiation (actions of an entity are to be traced back uniquely to that entity)
  - authenticity (verifying that users are who they say they are and that each input arriving at the system came from a trusted source)
- Availability – Ensuring timely and reliable access to and use of information.



Source: Figure 1.2 from William Stallings – Cryptography and Network Security, 5th Edition

Cryptography Algorithms in Use

- Confidentiality – Public-key encryption algorithms to exchange a secret key and Symmetric key algorithms for encrypting the actual data.
- Integrity – Hashing algorithms to compute a hash value of the message and public-key encryption algorithms to encrypt the hash value with the private key (to form a digital signature).
- Non-repudiation – Public-key encryption algorithms used to digitally sign a message with the sender's private key.
- Authentication – Passwords, {Public-key certificates and digital signatures} and Biometrics are typically preferred for authentication. Symmetric encryption is also OK; but, not preferred.

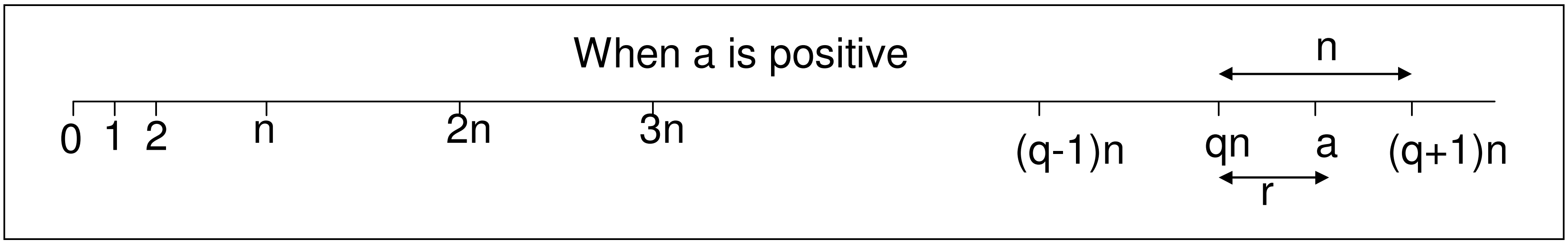
Public Key Encryption

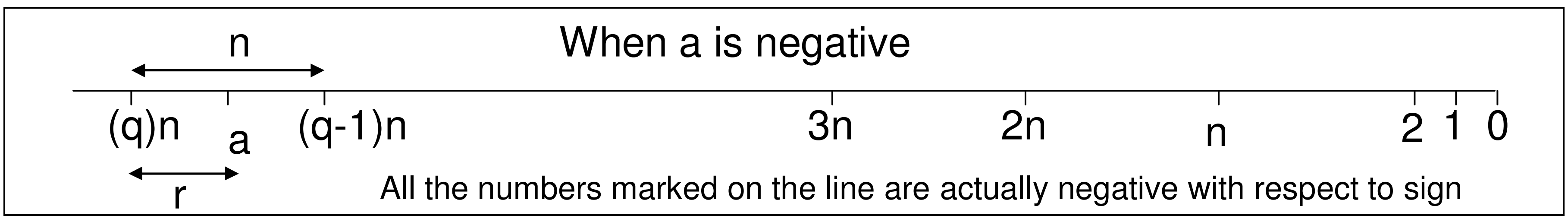
- Motivation: Key distribution problem of symmetric encryption system
- Let K_{PRIV} and K_{PUB} be the private key and public key of a user. Then,
  - $P = D(K_{PRIV}, E(K_{PUB}, P))$
  - With some, public key encryption algorithms like RSA, the following is also true: $P = D(K_{PUB}, E(K_{PRIV}, P))$
- In a system of n users, the number of secret keys for point-to-point communication is $n(n-1)/2 = O(n^2)$. With the public key encryption system, we need 2 keys (one public and one private key) per user. Hence, the total number of keys needed is $2n = O(n)$.

	Secret Key (Symmetric)	**Public Key (Asymmetric)**
Number of Keys	1	2
Protection of Key	Must be secret	One key must be secret; the key can be publicly exposed
Best uses	Cryptographic workhorse; secrecy and integrity of data	Key exchange, authentication
Key distribution	Must be out-of-band	Public key can be used to distribute other keys
Speed	Fast	Slow

Modular Arithmetic

- Given any positive integer n and any integer a, if we divide a by n, we get a quotient q and a remainder r that obey the following relationship:
  - a = q * n + r, $0 \le r < n$ and r is the remainder, q is the quotient





  - Example:
    - a = 59; n = 7;   59 = (8)*7 + 3   r = 3; q = 8
    - a = -59; n = 7; -59 = (-9)*7 + 4   r = 4; q = -9
    - 59 mod 7 = 3
    - -59 mod 7 = 4

Modular Arithmetic

- Two integers a and b are said to be congruent modulo n, if a mod n = b mod n. This is written as $a \equiv b \bmod n$.
  - We say "a and b are equivalent to each other in class modulo n"

- Example:
  - $73 \equiv 4 \bmod 23$, because 73 mod 23 = 4 = 4 mod 23
  - $21 \equiv -9 \bmod 10$, because 21 mod 10 = 1 = -9 mod 10

- Properties of the Modulo Operator
  - If $a \equiv b \bmod n$, then $(a - b) \bmod n = 0$
  - If $a \equiv b \bmod n$, then $b \equiv a \bmod n$
  - If $a \equiv b \bmod n$ and $b \equiv c \bmod n$, then $a \equiv c \bmod n$

- Example:
  - $73 \equiv 4 \bmod 23$, then $(73 - 4) \bmod 23 = 0$
  - $73 \equiv 4 \bmod 23$, then $4 \equiv 73 \bmod 23$, because 4 mod 23 = 73 mod 23
  - $73 \equiv 4 \bmod 23$ and $4 \equiv 96 \bmod 23$, then $73 \equiv 96 \bmod 23$.

Modular Arithmetic

- Now, that we have studied the meaning of "<u>equivalency</u>" or "congruent modulo n", it is see that the "<u>mod n</u>" operator maps "all integers" (negative and positive) into the set of integers [0, 1, …., n-1].
- Example: Class of modulo 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
-60	-59	-58	-57	-56	-55	-54	-53	-52	-51	-50	-49	-48	-47	-46
-45	-44	-43	-42	-41	-40	-39	-38	-37	-36	-35	-34	-33	-32	-31
-30	-29	-28	-27	-26	-25	-24	-23	-22	-21	-20	-19	-18	-17	-16
-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

- From the above table, we could say things like
  - $-38 \equiv 22 \bmod 15$ $\qquad$ $24 \equiv 54 \bmod 15$
  - $-38 \bmod 15 = 7$ $[-38 = (-3)*15 + 7]$ $\qquad$ $24 \bmod 15 = 9$ $[24 = (1)*15 + 9]$
  - $22 \bmod 15 = 7$ $[22 = (1)*15 + 7]$ $\qquad$ $54 \bmod 15 = 9$ $[54 = (3)*15 + 9]$

Modular Arithmetic

- Properties:
  - (x + y) mod n = (x mod n + y mod n) mod n
  - Example:
    - Compute: (54 + 49) mod 15
      - (54 + 49) mod 15 = 103 mod 15 = <u>13</u>
      - 54 mod 15 = 9
      - 49 mod 15 = 4
      - (54 mod 15 + 49 mod 15) = 9 + 4 = 13
      - (54 mod 15 + 49 mod 15) mod 15 = 13 mod 15 = <u>13</u>
  - Example:
    - Compute (42 + 52) mod 15
      - (42 + 52) mod 15 = 94 mod 15 = <u>4</u>
      - 42 mod 15 = 12
      - 52 mod 15 = 7
      - (42 mod 15 + 52 mod 15) = 12 + 7 = 19
      - (42 mod 15 + 52 mod 15) mod 15 = 19 mod 15 = <u>4</u>

Modular Arithmetic

- Properties:
  - (x * y) mod n = (x mod n * y mod n) mod n
  - Example:
    - Compute: (54 * 49) mod 15
      - (54 * 49) mod 15 = 2646 mod 15 = <u>6</u>
      - 54 mod 15 = 9
      - 49 mod 15 = 4
      - (54 mod 15 * 49 mod 15) = 9 * 4 = 36
      - (54 mod 15 * 49 mod 15) mod 15 = 36 mod 15 = <u>6</u>
  - Example:
    - Compute (42 * 52) mod 15
      - (42 * 52) mod 15 = 2184 mod 15 = <u>9</u>
      - 42 mod 15 = 12
      - 52 mod 15 = 7
      - (42 mod 15 * 52 mod 15) = 12 * 7 = 84
      - (42 mod 15 * 52 mod 15) mod 15 = 84 mod 15 = <u>9</u>

Modular Arithmetic

- Properties:
  – (a * b * c) mod n = ((a mod n) * (b mod n) * (c mod n)) mod n
  – (a * b * c) mod n = ((((a mod n) * (b mod n)) mod n) * (c mod n)) mod n
  – (a * b * c * d) mod n = ((a mod n) * (b mod n) * (c mod n) * (d mod n)) mod n
  – Similarly, (a * b * c * d * e) mod n….

  – Example:
    - Compute (42 * 56 * 98 * 108) mod 15
    - Straightforward approach: (42 * 56 * 98 * 108) mod 15 = (24893568) mod 15 = 3
    - <u>Optimum approach 1</u> <u>Optimum approach 2</u>

Optimum approach 1

- 42 mod 15 = 12
- 56 mod 15 = 11
- 98 mod 15 = 8
- 108 mod 15 = 3
- (42 * 56 * 98 * 108) mod 15
  = (12 * 11 * 8 * 3) mod 15
  = (3168) mod 15 = 3

Optimum approach 2

- First Compute (42 * 56) mod 15
- (42 * 56) mod 15 = (12 * 11) mod 15 = 12
- Then, compute (42 * 56 * 98) mod 15
- (42 * 56 * 98) mod 15 = (12 * 98) mod 15 = (12 * 8) mod 15 = 6
- Now, compute (42 * 56 * 98 * 108) mod 15
- (42 * 56 * 98 * 108) mod 15 = (6 * 108) mod 15 = (6 * 3) mod 15 = 3

Modular Arithmetic

- Modular Exponentiation
  - The Right-to-Left Binary Algorithm

To compute b^e mod n

First, write the exponent e in binary notation.

$$e = \sum_{i=0}^{m-1} a_i 2^i$$

In this notation, the length of e is m bits. For any i, such that $0 \leq i < m-1$, the a_i take the value of 0 or 1. By definition, $a_{m-1} = 1$.

$$b^e = b^{\left(\sum_{i=0}^{m-1} a_i 2^i\right)} = \prod_{i=0}^{m-1} \left(b^{2^i}\right)^{a_i}$$

Solution for b^e mod n = $\prod_{i=0}^{m-1} \left(b^{2^i}\right)^{a_i} \bmod n$

Example for Modular Exponentiation

- To compute $5^{41} \bmod 9$
  - Straightforward approach:
    - $5^{41} \bmod 9$ = (45474735088646411895751953125) mod 9 = 2
    - Number of multiplications - 40
  - Using the Right-to-Left Binary Algorithm
    - Write 41 in binary: 101001
    - $5^{41} = 5^{32} * 5^{8} * 5^{1}$

32	16	8	4	2	1
1	0	1	0	0	1

$5^1 \bmod 9 = 5 \bmod 9 = 5$

$5^2 \bmod 9 = (5^1 * 5^1) \bmod 9 = (5 \bmod 9 * 5 \bmod 9) \bmod 9 = (5 * 5) \bmod 9 = 25 \bmod 9 = 7$

$5^4 \bmod 9 = (5^2 * 5^2) \bmod 9 = (5^2 \bmod 9 * 5^2 \bmod 9) \bmod 9 = (7 * 7) \bmod 9 = 49 \bmod 9 = 4$

$5^8 \bmod 9 = (5^4 * 5^4) \bmod 9 = (5^4 \bmod 9 * 5^4 \bmod 9) \bmod 9 = (4 * 4) \bmod 9 = 16 \bmod 9 = 7$

$5^{16} \bmod 9 = (5^8 * 5^8) \bmod 9 = (5^8 \bmod 9 * 5^8 \bmod 9) \bmod 9 = (7 * 7) \bmod 9 = 49 \bmod 9 = 4$

$5^{32} \bmod 9 = (5^{16} * 5^{16}) \bmod 9 = (5^{16} \bmod 9 * 5^{16} \bmod 9) \bmod 9 = (4 * 4) \bmod 9 = 16 \bmod 9 = 7$

$$
\begin{aligned}
5^{41} \bmod 9 &= (5^{32} * 5^8 * 5^1) \bmod 9 \\
&= (7 * 7 * 5) \bmod 9 \\
&= ((49 \bmod 9) * (5 \bmod 9)) \bmod 9 \\
&= (4 * 5) \bmod 9 \\
&= 20 \bmod 9 \\
&= 2
\end{aligned}
$$

Number of multiplications: 5 + 2 = 7

Example for Modular Exponentiation

- To compute 3^{61} mod 8
  - Straightforward approach:
    - 3^{61} mod 8 = (127173478256486195428832996O3) mod 8 = 3
    - Number of multiplications - 60
  - Using the Right-to-Left Binary Algorithm
    - <u>Write 61 in binary: </u>111101
    - $3^{41} = 3^{32} * 3^{16} * 3^{8} * 3^{4} * 3^{1}$

32	16	8	4	2	1
1	1	1	1	0	1

$3^1 \bmod 8 = 3 \bmod 8 = 3$

$3^2 \bmod 8 = (3^1 * 3^1) \bmod 8 = (3 \bmod 8 * 3 \bmod 8) \bmod 8 = (3 * 3) \bmod 8 = 9 \bmod 8 = 1$

$3^4 \bmod 8 = (3^2 * 3^2) \bmod 8 = (3^2 \bmod 8 * 3^2 \bmod 8) \bmod 8 = (1 * 1) \bmod 8 = 1 \bmod 8 = 1$

$3^8 \bmod 8 = (3^4 * 3^4) \bmod 8 = (3^4 \bmod 8 * 3^4 \bmod 8) \bmod 8 = (1 * 1) \bmod 8 = 1 \bmod 8 = 1$

$3^{16} \bmod 8 = (3^8 * 3^8) \bmod 8 = (3^8 \bmod 8 * 3^8 \bmod 8) \bmod 8 = (1 * 1) \bmod 8 = 1 \bmod 8 = 1$

$3^{32} \bmod 8 = (3^{16} * 3^{16}) \bmod 8 = (3^{16} \bmod 8 * 3^{16} \bmod 8) \bmod 8 = (1 * 1) \bmod 8 = 1 \bmod 8 = 1$

$$
\begin{aligned}
3^{61} \bmod 8 &= (3^{32} * 3^{16} * 3^{8} * 3^{4} * 3^{1}) \bmod 8 \\
&= (1 * 1 * 1 * 1 * 3) \bmod 8 \\
&= ((1 \bmod 8) * (1 * 1 * 3 \bmod 9)) \bmod 8 \\
&= ((1 * 1) \bmod 8 * (1 * 3)) \bmod 8 \\
&= ((1 * 1) \bmod 8 * (3)) \bmod 8 \\
&= (1 * 3) \bmod 8 \\
&= 3 \bmod 8 = 3
\end{aligned}
$$

Number of multiplications: 5 + 4 = 9

Multiplicative Inverse Modulo n

- If (a * b) modulo n = 1, then
  - a is said to be the multiplicative inverse of b in class modulo n
  - b is said to be the multiplicative inverse of a in class modulo n
- Example:
  - Find the multiplicative inverse of 7 in class modulo 15
  - Straightforward approach:
    - Multiply 7 with all the integers [0, 1, …, 14] in class modulo 15
    - There will be only one integer x for which (7*x) modulo 15 = 1

X	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
(7 * X) modulo 15	0	7	14	6	13	5	12	4	11	3	10	2	9	1	8

  - Find the multiplicative inverse of 9 in class modulo 13
    - Multiply 9 with all the integers [0, 1, …, 12] in class modulo 13
    - There will be only one integer x for which (9*x) modulo 13 = 1

X	0	1	2	3	4	5	6	7	8	9	10	11	12
(9 * X) modulo 13	0	9	5	1	10	6	2	11	7	3	12	8	4

- A more efficient approach to find multiplicative inverse in class modulo n is to use the Extended Euclid Algorithm

Euclid's Algorithm to find the GCD

- Given two integers m and n (say m > n), then
  - GCD (m, n) = GCD (n, m mod n)
  - One can continue using the above recursion until the second term becomes 0. The GCD (m, n) will be then the value of the first term, because GCD (k, 0) = k

- Example: GCD (120, 45)
  - GCD (120, 45) = GCD (45, 30) = GCD (30, 15) = GCD (15, 0) = 15
- Example: GCD (45, 12)
  - GCD (45, 12) = GCD (12, 9) = GCD (9, 3) = GCD (3, 0) = 3
- Example: GCD (53, 30)
  - GCD (53, 30) = GCD (30, 23) = GCD (23, 7) = GCD (7, 2) = GCD (2, 1) = GCD (1, 0) = 1

- Note: Two numbers m and n are said to be relatively prime if
  - GCD (m, n) = 1.

Property of GCD

- For any two integers m and n,
  - We can write m * x + n * y = GCD (m, n)
    - x and y are also integers
    - We find x and y through the Extended Euclid algorithm

- If m and n are relatively prime, then
  - there exists two integers x and y such that m * x + n * y = 1
    - x is the multiplicative inverse of m modulo n
    - y is the multiplicative inverse of n modulo m
    - We could find x and y through the Extended Euclid algorithm

Extended Euclid Algorithm

- Theorem Statement
  - Let m and n be positive integers. Define
    - a[0] = m, a[1] = n
    - x[0] = 1, x[1] = 0, y[0] = 0, y[1] = 1,
    - q[k] = Floor(a[k-1]/ a[k]) for k > 0
    - a[k] = a[k-2] – (a[k-1]*q[k-1]) for k > 1
    - x[k] = x[k-2] – (q[k-1] * x[k-1]) for k > 1
    - y[k] = y[k-2] – (q[k-1] * y[k-1]) for k > 1
  - If a[p] is the last non-zero a[k], then
    - a[p] = GCD (m, n) = x[p] * m + y[p] * n
    - x[p] is the multiplicative inverse of m modulo n
    - y[p] is the multiplicative inverse of n modulo m

Example for Extended Euclid Algorithm

- Find the multiplicative inverse of 30 modulo 53
  - The larger of the two numbers is our m and the smaller is n
  - Initial Setup of the computation table

We want to find the x and y such that 53x + 30y = 1

	a	q	x	y
m →	53	-	1	0
n →	30		0	1

Iteration 1

a	q	x	y
53	-	1	0
30	1	0	1

a	q	x	y
53	-	1	0
30	1	0	1
23			

a	q	x	y
53	-	1	0
30	1	0	1
23		1	

a	q	x	y
53	-	1	0
30	1	0	1
23		1	-1

Iteration 2

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7			

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7		-1	

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7		-1	2

Example for Extended Euclid Algorithm

Iteration 3

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7	3	-1	2

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7	3	-1	2
2			

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7	3	-1	2
2		4	

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7	3	-1	2
2		4	-7

Iteration 4

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7	3	-1	2
2	3	4	-7

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7	3	-1	2
2	3	4	-7
1			

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7	3	-1	2
2	3	4	-7
1		-13	

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7	3	-1	2
2	3	4	-7
1		-13	23

Iteration 5

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7	3	-1	2
2	3	4	-7
1	2	-13	23

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7	3	-1	2
2	3	4	-7
1	2	-13	23
0			

a	q	x	y
53	-	1	0
30	1	0	1
23	1	1	-1
7	3	-1	2
2	3	4	-7
1	2	-13	23

STOP!

$-13*53+30*23 = 1 = GCD$

23 is the multiplicative inverse of 30 modulo 53

-13 ≡ 17 is the Multiplicative inverse of 53 modulo 30

Example for Extended Euclid Algorithm

- Find the multiplicative inverse of 17 modulo 89
  - The larger of the two numbers is our m and the smaller is n
  - <u>Initial Setup</u> of the computation table

We want to find the x and y such that 89x + 17y = 1

	a	q	x	y
m →	89	-	1	0
n →	17		0	1

Iteration 1

a	q	x	y
89	-	1	0
17	5	0	1

a	q	x	y
89	-	1	0
17	5	0	1
4			

a	q	x	y
89	-	1	0
17	5	0	1
4		1	

a	q	x	y
89	-	1	0
17	5	0	1
4		1	-5

Iteration 2

a	q	x	y
89	-	1	0
17	5	0	1
4	4	1	-5

a	q	x	y
89	-	1	0
17	5	0	1
4	4	1	-5
1			

a	q	x	y
89	-	1	0
17	5	0	1
4	4	1	-5
1			

a	q	x	y
89	-	1	0
17	5	0	1
4	4	1	-5
1		-4	21

Example for Extended Euclid Algorithm

Iteration 3

a	q	x	y
89	-	1	0
17	5	0	1
4	4	1	-5
1	4	-4	21

a	q	x	y
89	-	1	0
17	5	0	1
4	4	1	-5
1	4	-4	21
0			

a	q	x	y
89	-	1	0
17	5	0	1
4	4	1	-5
1	4	-4	21

STOP!

-4*89 + 21*17 = 1 = GCD

21 is the multiplicative inverse of 17 modulo 89

- 4 ≡ 13 is the multiplicative inverse of 89 modulo 17

RSA Algorithm

- The RSA algorithm uses two keys, *d* and *e*, which work in pairs, for decryption and encryption, respectively.
- A plaintext message P is encrypted to ciphertext by:
  – $C = P^e \bmod n$
- The plaintext is recovered by:
  – $P = C^d \bmod n$
- Because of symmetry in modular arithmetic, encryption and decryption are mutual inverses and commutative. Therefore,
  – $P = C^d \bmod n = (P^e)^d \bmod n = (P^d)^e \bmod n$
- Thus, one can apply the encrypting transformation first and then the decrypting one, or the decrypting transformation first followed by the encrypting one.

- <u>On the complexity of RSA:</u> It is very difficult to factorize a large integer into two prime factors. The number of prime numbers between 2 and *n* is $(n/(\ln n))$.
- <u>Euler's Phi Function for Positive Prime Integers:</u> For any positive prime integer p, (p-1) is the number of positive integers less than p and relatively prime to p.

Key Choice for RSA Algorithm

- The encryption key consists of the pair of integers (e, n) and the decryption key consists of the pair of integers (d, n).
- Finding the value of n:
  - Choose two large prime numbers p and q (approximately at least 100 digits each)
  - The value of n is p * q, and hence n is also very large (approximately at least 200 digits).
  - Trump card of RSA: `A large value of n inhibits us to find the prime factors p and q.`
- Choosing e:
  - Choose e to be a very large integer that is relatively prime to (p-1)*(q-1).
  - To guarantee the above requirement, choose e to be greater than both p-1 and q-1
- Choosing d:
  - Select d such that (e * d) mod ((p-1)*(q-1)) = 1
  - In other words, d is the multiplicative inverse of e in class modulo (p-1)*(q-1)

Example for RSA Algorithm

- Let p = 11 and q = 13. Find the encryption and decryption keys. Choose your encryption key to be at least 10. Show the encryption and decryption for Plaintext 7

a	q	x	y
120	-	1	0
11	10	0	1
10	1	1	-10
1	10	-1	11
0			

Solution:

- The value of n = p*q = 11*13 = 143
- (p-1)*(q-1) = 10*12 = 120
- Choose the encryption key e = 11, which is relatively prime to 120 = (p-1)*(q-1).
- The decryption key d is the multiplicative inverse of 11 modulo 120.
- Run the Extended Euclid algorithm with m = 120 and n = 11.
- We find the decryption key d to be also 11 (the multiplicative inverse of 11 in class modulo 120)

- The encryption key is (11, 143)
- The decryption key is (11, 143)

Example for RSA Algorithm

- Encryption for Plaintext P = 7
- Ciphertext C = P^e mod n

  = 7^{11} mod 143

8	4	2	1
1	0	1	1

7^1 mod 143 = 7 mod 143 = 7

7^2 mod 143 = (7^1 * 7^1) mod 143 = (7 mod 143 * 7 mod 143) mod 143 = (7 * 7) mod 143 = 49 mod 143 = 49

7^4 mod 143 = (7^2 * 7^2) mod 143 = (7^2 mod 143 * 7^2 mod 143) mod 143 = (49 * 49) mod 143 = 2401 mod 143 = 113

7^8 mod 143 = (7^4 * 7^4) mod 143 = (7^4 mod 143 * 7^4 mod 143) mod 143 = (113 * 113) mod 143 = 12769 mod 143 = 42

7^{11} mod 143 = (7^8 * 7^2 * 7^1) mod 143
= (42 * 49 * 7) mod 143
= (((42 * 49) mod 143) * (7)) mod 143
= (((2058) mod 143) * (7)) mod 143
= ((56) * (7)) mod 143
= (392) mod 143
= 106

Ciphertext is 106

Example for RSA Algorithm

- Decryption for Ciphertext C = 106
- Plaintext P = C^d mod n

  $= 106^{11} \bmod 143$

8	4	2	1
1	0	1	1

$106^1 \bmod 143 = 106 \bmod 143 = 106$

$106^2 \bmod 143 = (106^1 * 106^1) \bmod 143 = (106 \bmod 143 * 106 \bmod 143) \bmod 143 = (106 * 106) \bmod 143 = 49 \bmod 143 = 82$

$106^4 \bmod 143 = (106^2 * 106^2) \bmod 143 = (106^2 \bmod 143 * 106^2 \bmod 143) \bmod 143 = (82 * 82) \bmod 143 = 6724 \bmod 143 = 3$

$106^8 \bmod 143 = (106^4 * 106^4) \bmod 143 = (106^4 \bmod 143 * 106^4 \bmod 143) \bmod 143 = (3 * 3) \bmod 143 = 9 \bmod 143 = 9$

$106^{11} \bmod 143 = (106^8 * 106^2 * 106^1) \bmod 143$

$= (9 * 82 * 106) \bmod 143$

$= (((9 * 82) \bmod 143) * (106)) \bmod 143$

$= (((738) \bmod 143) * (106)) \bmod 143$

$= ((23) * (106)) \bmod 143$

$= (2438) \bmod 143$

$= 7$

Plaintext is 7

Another Example for RSA Algorithm

- Let p = 17 and q = 23. Find the encryption and decryption keys. Choose your encryption key to be at least 10. Show the encryption and decryption for Plaintext 127

a	q	x	y
352	-	1	0
13	27	0	1
1	13	1	-27
0			

<u>Solution:</u>

- The value of n = p*q = 17*23 = 391
- (p-1)*(q-1) = 16*22 = 352
- Choose the encryption key e = 13, which is relatively prime to 352 = (p-1)*(q-1).
- The decryption key d is the multiplicative inverse of 13 modulo 352.
- Run the Extended Euclid algorithm with m = 352 and n = 13.
- The multiplicative inverse is -27 ≡ (-27 + 352) = 325
- We find the decryption key d to be 325 (the multiplicative inverse of 13 in class modulo 352)

- The encryption key is (13, 391)
- The decryption key is (325, 391)

Another Example for RSA Algorithm

- Encryption for Plaintext P = 127
- Ciphertext C = P^e mod n

  = 127^{13} mod 391

8	4	2	1
1	1	0	1

127^1 mod 391 = 127 mod 391 = 127

127^2 mod 391 = (127^1 * 127^1) mod 391 = (127 mod 391 * 127 mod 391) mod 391 = (127 * 127) mod 391 = 16129 mod 391 = 98

127^4 mod 391 = (127^2 * 127^2) mod 391 = (127^2 mod 391 * 127^2 mod 391) mod 391 = (98 * 98) mod 391 = 9604 mod 391 = 220

127^8 mod 391 = (127^4 * 127^4) mod 391 = (127^4 mod 391 * 127^4 mod 391) mod 391 = (220 * 220) mod 391 = 48400 mod 391 = 307

127^{13} mod 391 = (127^8 * 127^4 * 127^1) mod 391

= (307 * 220 * 127) mod 391

= (((307 * 220) mod 391) * (127)) mod 391

= (((67540) mod 391) * (127)) mod 391

= ((288) * (127)) mod 391

= (36576) mod 391

= 213

Ciphertext is 213

Another Example for RSA Algorithm

- Decryption for Ciphertext C = 213
- Plaintext P = C^d mod n

  = 213^{325} mod 391

256	128	64	32	16	8	4	2	1
1	0	1	0	0	0	1	0	1

$213^1 \bmod 391 = 213 \bmod 391 = 213$

$213^2 \bmod 391 = (213 * 213) \bmod 391 = 45369 \bmod 391 = 13$

$213^4 \bmod 391 = (13 * 13) \bmod 391 = 169 \bmod 391 = 169$

$213^8 \bmod 391 = (169 * 169) \bmod 391 = 28561 \bmod 391 = 18$

$213^{16} \bmod 391 = (18 * 18) \bmod 391 = 324 \bmod 391 = 324$

$213^{32} \bmod 391 = (324 * 324) \bmod 391 = 104976 \bmod 391 = 188$

$213^{64} \bmod 391 = (188 * 188) \bmod 391 = 35344 \bmod 391 = 154$

$213^{128} \bmod 391 = (154 * 154) \bmod 391 = 23716 \bmod 391 = 256$

$213^{256} \bmod 391 = (256 * 256) \bmod 391 = 65536 \bmod 391 = 239$

$$\begin{aligned} 213^{325} \bmod 391 &= (213^{256} * 213^{64} * 213^4 * 213^1) \bmod 391 \\ &= (239 * 154 * 169 * 213) \bmod 391 \\ &= (52 * 169 * 213) \bmod 391 \\ &= (186 * 213) \bmod 391 \\ &= 127 \end{aligned}$$

Plaintext is 127

Applications of Encryption

- Exchange of Shared Key using Asymmetric Encryption
  - Let K_{PUB-S}, K_{PRI-S} denote the public and private keys of Sender S. Similarly, let K_{PUB-R} and K_{PRI-R} be the public and private key of Receiver R. Let K be the secret key to be shared between only S and R.
  - S sends to R the following:
    - E (K_{PUB-R} E(K_{PRI-S}, K))
  - The inner encryption guarantees that the secret key K came from S and the outer encryption guarantees that only the receiver R could open the outer encryption of the message and get access to the inner encryption.

Applications of Encryption

- Diffie-Hellman Key Exchange
  - Used to allow two parties that have to establish a shared secret key over an insecure communication channel.
  - Alice and Bob agree on a field size n and a starting number g.
  - Alice generates a secret integer a and sends g^a mod n to Bob. Alice sends this encrypted using its private key, so that Bob can decrypt it using Alice's public key, thereby authenticating that the message came from Alice. $E(K_{PRI-ALICE}, g^a \bmod n)$
  - At the same time, Bob generates a secret integer b and sends g^b mod n to Alice. Bob sends this encrypted using its private key, thereby authenticating to Alice that the message came from Bob. $E(K_{PRI-Bob}, g^b \bmod n)$
  - When Bob gets Alice's message, it computes $(g^a)^b$ mod n and uses it as the secret key.
  - Similarly, when Alice gets Bob's message, it computes $(g^b)^a$ mod n and uses it as the secret key.
  - According to Modular arithmetic, $(g^a)^b \bmod n = (g^b)^a \bmod n$. Hence, both Alice and Bob have agreed on a shared secret key.

Applications of Encryption

- Digital Signatures
  - A digital signature is a protocol that produces the same effect as a real signature.
  - It is a mark that only the sender can make, but other people can easily recognize as it of being made by the sender.
  - Just like a real signature, a digital signature indicates the sender's agreement to the message.
  - Properties of a digital signature:
    - It must be unforgeable: If person P signs a message M with signature S(P, M), it is impossible for any one else to produce the pair [M, S(P, M)].
    - It must be authentic: If person R receives the pair [M, S(P, M)] from P, R can check that the signature is really from P. Only P could have created this signature, and the signature is firmly attached to M.
    - It is not alterable: After being transmitted, M cannot be changed by S, R or an interceptor.
    - It is not reusable: A previous message presented again will be instantly detected by R.
  - Public Key Protocol: S sends R $E(K_{PUB-R}\ E(K_{PRI-S}, M))$