# Security in Computing

## Chapter 2

## Elementary Cryptography (part 4)



# Chapter Outline





# AES Introduction

- DES:
  - too weak
  - 3DES too slow (esp in software)
  - theoretical attacks (linear, diff. cryptanalysis, weak keys, ...)
- NIST calls for replacement (1997)
- Requirements:
  - resistence to known cryptanalytic attacks
  - speed in hw and sw
  - limited size (*e.g.* smart cards)
  - resistence to attacks on implementations (timing, power, etc.)
  - instruction level parallelism potential
  - others

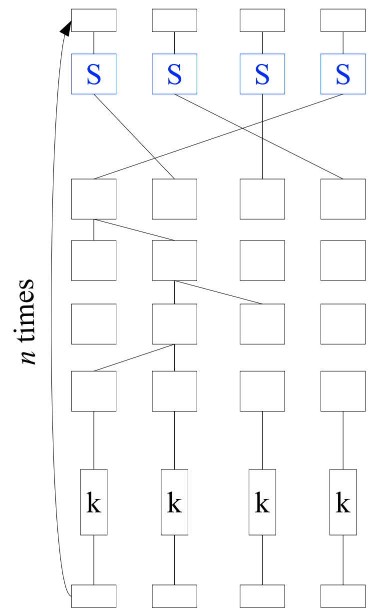


# AES

- winning algorithm: Rijndael (RINE dahl)
- no mathematical operators
  - endianness doesn't matter
- variable
  - block length
  - key size
- not Feistel structure

# Rijndael Cycle

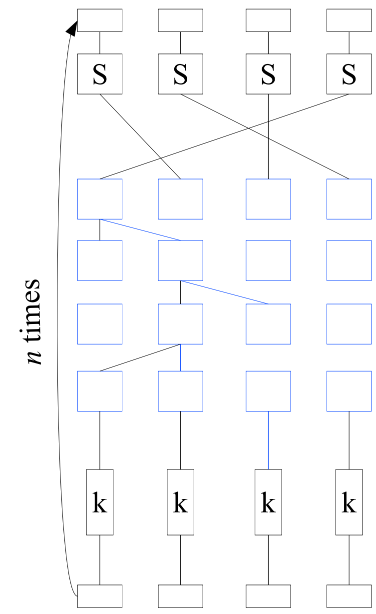
- 4 4-byte blocks
- Byte substitution
  - S-box
- Shift rows
  - simple permutation
- Mix columns
  - substitution
  - arithmetic over $GF(2^8)$
- Add round key
  - XOR with part of key






# Rijndael Cycle

- 4 4-byte blocks
- Byte substitution
  - S-box
- Shift rows
  - simple permutation
- Mix columns
  - substitution
  - arithmetic over $GF(2^8)$
- Add round key
  - XOR with part of key





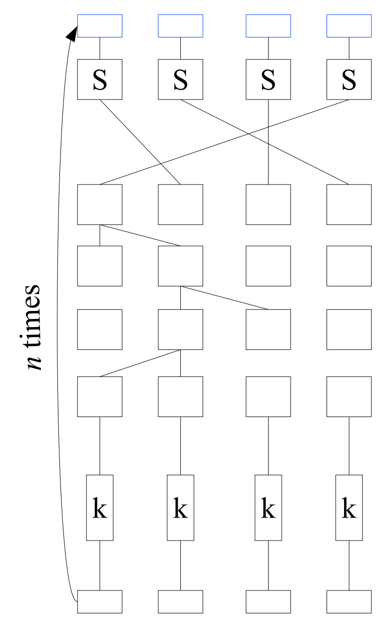

# Rijndael Overview

- Plaintext fed into state array (matrix)
- There are 9, 11, or 13 cycles
  - depends on whether 128, 192, or 256-bit keys are used
- Each cycle:
  - substitution
  - shift
  - mix column
  - XOR with subkey



# Rijndael Cycle

- 4 4-byte blocks
- Byte substitution
  - S-box
- Shift rows
  - simple permutation
- Mix columns
  - substitution
  - arithmetic over $GF(2^8)$
- Add round key
  - XOR with part of key



# Secret Key Encryption

- How many keys necessary for *n* people?



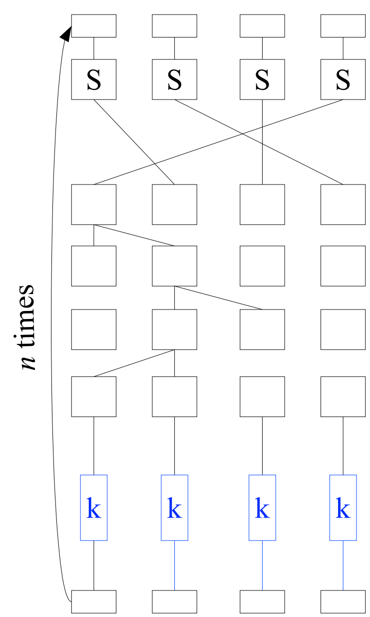
# Secret Key Encryption

- How many keys necessary for *n* people?
  - 1 person needs 0 keys



# Rijndael Cycle

- 4 4-byte blocks
- Byte substitution
  - S-box
- Shift rows
  - simple permutation
- Mix columns
  - substitution
  - arithmetic over GF($2^8$)
- Add round key
  - XOR with part of key

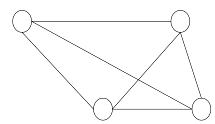





# Chapter Outline

# Secret Key Encryption

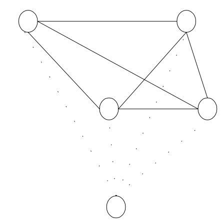
- How many keys necessary for *n* people?
  - 1 person needs 0 keys
  - 2 people need 1 key
  - 3 people need 3 keys
  - 4 people need 6 keys





# Secret Key Encryption

- How many keys necessary for *n* people?
  - 1 person needs 0 keys
  - 2 people need 1 key
  - 3 people need 3 keys
  - 4 people need 6 keys
  - 5 people need ?





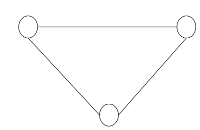
# Secret Key Encryption

- How many keys necessary for *n* people?
  - 1 person needs 0 keys
  - 2 people need 1 key



# Secret Key Encryption

- How many keys necessary for *n* people?
  - 1 person needs 0 keys
  - 2 people need 1 key
  - 3 people need 3 keys



## Secret Key Encryption Problems

- How many keys are necessary?
  - $O(n^2)$ keys
- How do you create and distribute the keys?



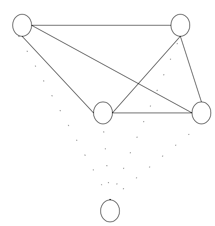
## Asymmetric Algorithm

- encryption, decryption keys different
- encryption key: $K_E$
- decryption key: $K_D$
  - $C = E(K_E, P)$
  - $P = D(K_D,C)$
  - $P = D(K_D, E(K_E, P))$



## Secret Key Encryption

- How many keys necessary for *n* people?
  - 1 person needs 0 keys
  - 2 people need 1 key
  - 3 people need 3 keys
  - 4 people need 6 keys
  - 5 people need 10 keys

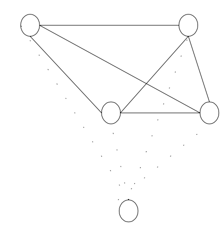


- for 5 people:
  - 6 + 4 = 10



## Secret Key Encryption

- How many keys necessary for *n* people?
  - 1 person needs 0 keys
  - 2 people need 1 key
  - 3 people need 3 keys
  - 4 people need 6 keys
  - 5 people need 10 keys



- for 5 people:
  - 6 + 4 = 10
- for *n* people?

$$\frac{(n)(n-1)}{2} = O(n^2)$$

## So What Can You Do With This?

- **Encryption** keep your data secret
- **Authentication** you are who you say you are
- **Integrity** the message hasn't been changed

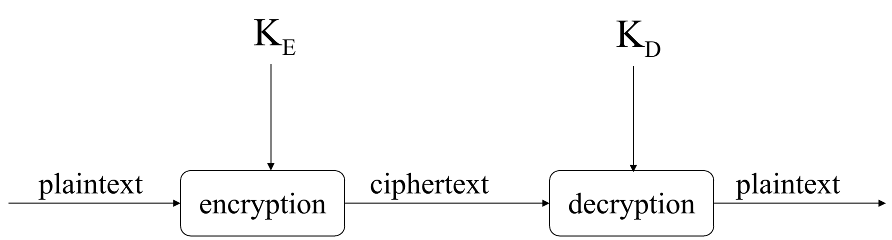


## Using Public Key Cryptography

- Who knows what?
  - Everyone can know your public key
  - Nobody should ever know your private key
- The keys are inverses of each other:
  - Anything encrypted with your *public* key can only be decrypted with your *private* key.
  - Anything encrypted with your *private* key can only be decrypted with your *public* key.

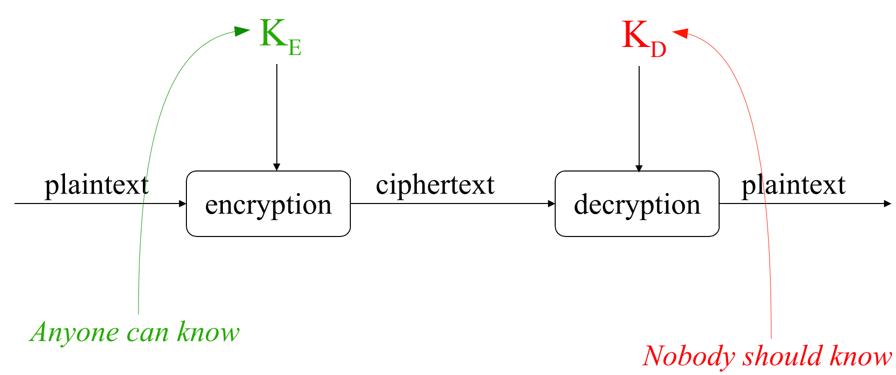


## Asymmetric Algorithm Diagram






## Asymmetric Algorithm Diagram



## Sending an Encrypted Message

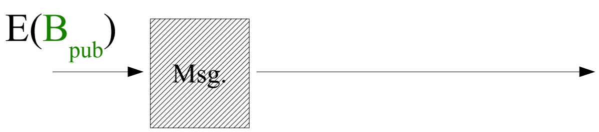
A

- A wants to send Msg. to B
- Only B should be able to read it.

B

A's private key

A's public key




B's public key

B's private key

- A encrypts Msg. with B's public key.
- B decrypts with B's private key.



## Digital Signatures

- Should work like handwritten signatures
  - Verify the sender of the document
- A sends a message to B
  - How can A prove that she really sent the message?



## Sending an Encrypted Message

A

Remember:
- Anyone can know your public key
- Nobody should ever know your private key

B

A's private key

A's public key

B's public key

B's private key



## Sending an Encrypted Message

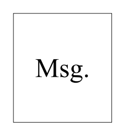
A

- A wants to send Msg. to B
- Only B should be able to read it.

B

A's private key

A's public key

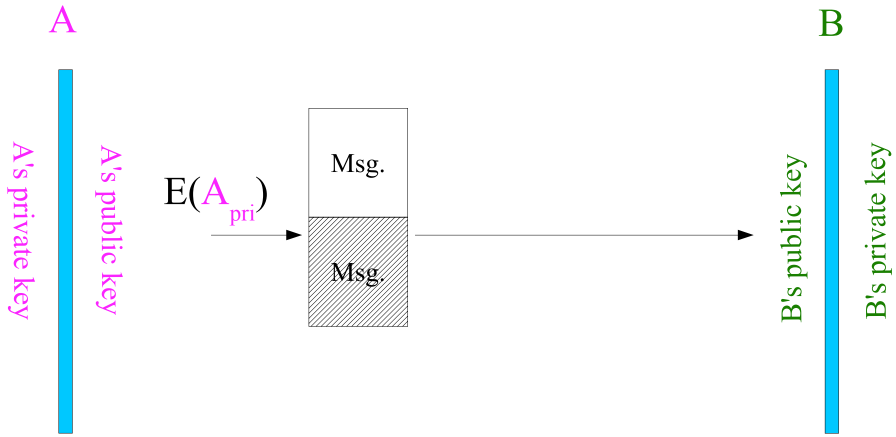



B's public key

B's private key

# Digitally Signed Messages

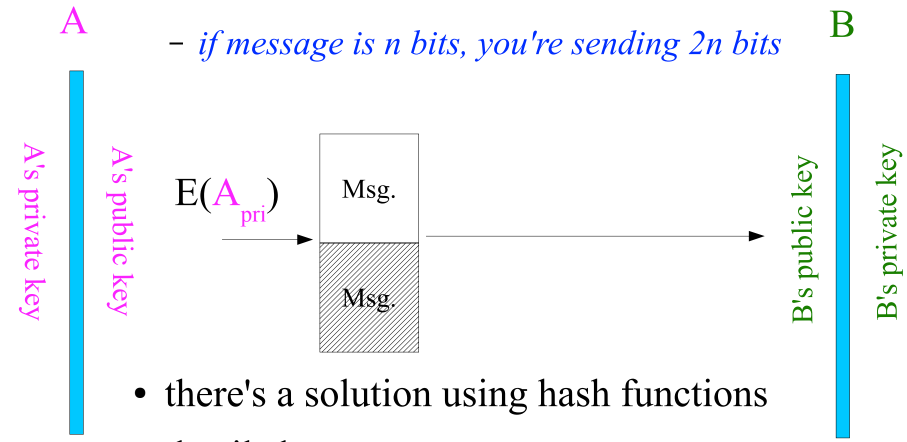
- Problem with this?






# Digitally Signed Messages

- Problem with this?
  - *if message is n bits, you're sending 2n bits*

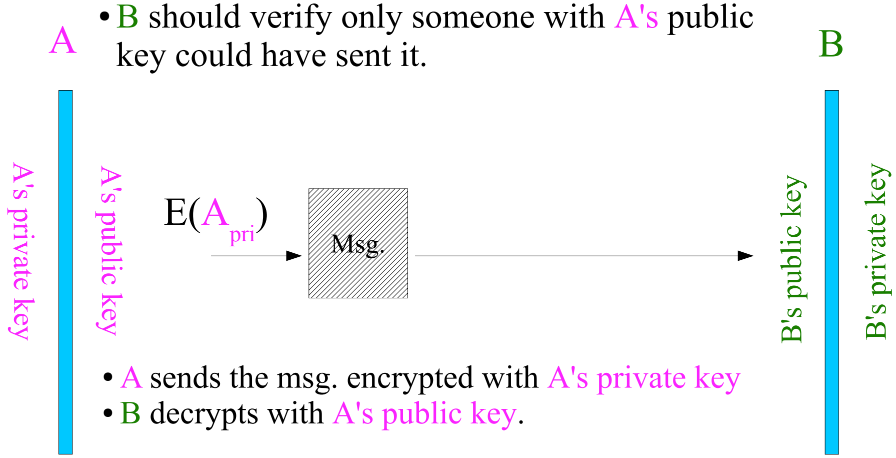



- there's a solution using hash functions
- details later



# Digitally Signed Messages

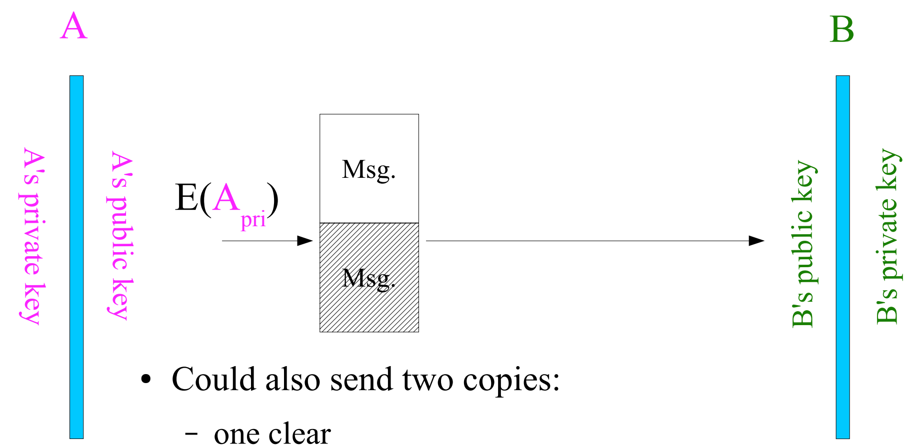
- A wants to send a message to B.
- B should verify only someone with A's public key could have sent it.




- A sends the msg. encrypted with A's private key
- B decrypts with A's public key.



# Digitally Signed Messages




- Could also send two copies:
  - one clear
  - one encrypted with A's private key

# Confidentiality and Authentication

- A both signs and encrypts the message
- Could either:
  - $E_{Apri}(E_{Bpub}(M))$ *-or-*
  - $E_{Bpub}(E_{Apri}(M))$



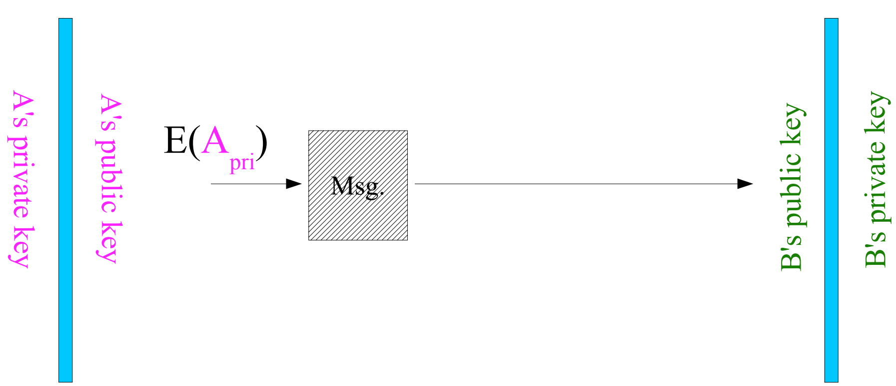
# public key algorithms

- first public algorithms in the 1970s
- we'll do details of RSA



# Digitally Signed Messages

A Question: does this provide confidentiality? B

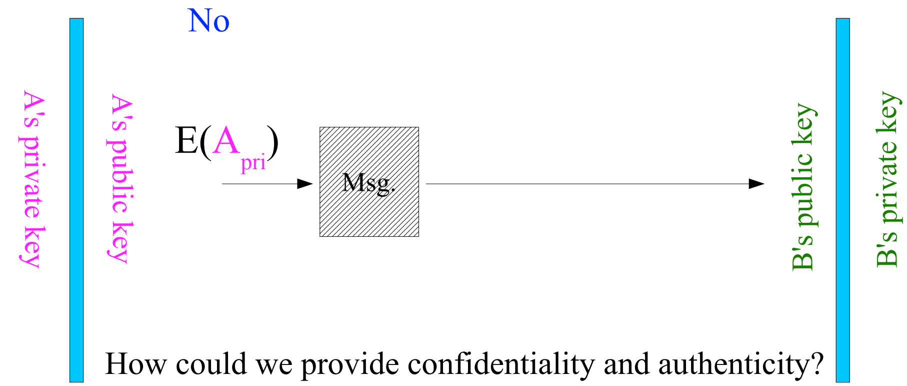





# Digitally Signed Messages

A Question: does this provide confidentiality? B

No




How could we provide confidentiality and authenticity?

## What's the answer to these?

- 7+2 mod 10 = ?
- 8+2 mod 10 = ?
- 6+5 mod 10 = ?
- 22 + 22 mod 10 = ?



## What's the answer to these?

- 7+2 mod 10 = 9
- 8+2 mod 10 = 0
- 6+5 mod 10 = 1
- 22 + 22 mod 10 = 4

- Can even make an addition table:



## RSA Key Generation

- choose large primes p, q; $p \neq q$
- calculate $n = pq$
- calculate $\varphi(n) = (p-1)(q-1)$
- choose e where $\gcd(e, \varphi(n))=1$; $1 < e < \varphi(n)$
- choose d where $d = e^{-1} \bmod \varphi(n)$;



## Modular Arithmetic Review

- Modular arithmetic review
- How RSA works

# Multiplication mod 10

| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **2** | 0 | 2 | 3 | 6 | 7 | 0 | 2 | 4 | 6 | 8 |
| **3** | 0 | 3 | 6 | 9 | 2 | 5 | 8 | 1 | 4 | 7 |
| **4** | 0 | 4 | 8 | 2 | 6 | 0 | 4 | 8 | 2 | 6 |
| **5** | 0 | 5 | 0 | 5 | 0 | 5 | 0 | 5 | 0 | 5 |
| **6** | 0 | 6 | 2 | 8 | 4 | 0 | 6 | 2 | 8 | 4 |
| **7** | 0 | 7 | 4 | 1 | 8 | 5 | 2 | 9 | 6 | 4 |
| **8** | 0 | 8 | 6 | 4 | 2 | 0 | 8 | 6 | 4 | 2 |
| **9** | 0 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |



# Is multiplication reversible?

- In regular arithmetic, x's multiplicative inverse is 1/x
- We multiply x by 1/x to get 1
- Can we use this for a cipher?
  - Multiply by some number k to encrypt
  - Multiply by 1/k to decrypt
- Look again at multiplication mod 10



# Addition mod 10

| + | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **1** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
| **2** | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
| **3** | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 |
| **4** | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 |
| **5** | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 |
| **6** | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 |
| **7** | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| **8** | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **9** | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |



# Addition is reversible

- In regular addition, we add -x to x to get 0
- -x is x's additive inverse
- Can we use this to do cryptography?
  - Add something to encrypt
  - Add an inverse to decrypt
- Yes, but it'd be lame.

# Why do 1,3,7,9 work?

- {1,3,7,9} work because they're *relatively prime* with 10.
- Recall: *a* and *b* are relatively prime if gcd(*a*,*b*)=1
- When working mod n, all #s relatively prime with n will have multiplicative inverses
  - Any number that isn't relatively prime will not have a multiplicative inverse.



# RSA Key Generation

- choose large primes p, q; $p \neq q$
- calculate n = pq
- calculate $\varphi(n) = (p-1)(q-1)$
- choose e where $\gcd(e, \varphi(n))=1$; $1 < e < \varphi(n)$
- choose d where $d = e^{-1} \bmod \varphi(n)$;

- public key = {e, n}
- private key = {d, n}



# Multiplication mod 10

Is it reversible?

Are there certain values that are?

| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **2** | 0 | 2 | 3 | 6 | 7 | 0 | 2 | 4 | 6 | 8 |
| **3** | 0 | 3 | 6 | 9 | 2 | 5 | 8 | 1 | 4 | 7 |
| **4** | 0 | 4 | 8 | 2 | 6 | 0 | 4 | 8 | 2 | 6 |
| **5** | 0 | 5 | 0 | 5 | 0 | 5 | 0 | 5 | 0 | 5 |
| **6** | 0 | 6 | 2 | 8 | 4 | 0 | 6 | 2 | 8 | 4 |
| **7** | 0 | 7 | 4 | 1 | 8 | 5 | 2 | 9 | 6 | 4 |
| **8** | 0 | 8 | 6 | 4 | 2 | 0 | 8 | 6 | 4 | 2 |
| **9** | 0 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |



# Multiplication mod 10

| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **2** | 0 | 2 | 3 | 6 | 7 | 0 | 2 | 4 | 6 | 8 |
| **3** | 0 | 3 | 6 | 9 | 2 | 5 | 8 | 1 | 4 | 7 |
| **4** | 0 | 4 | 8 | 2 | 6 | 0 | 4 | 8 | 2 | 6 |
| **5** | 0 | 5 | 0 | 5 | 0 | 5 | 0 | 5 | 0 | 5 |
| **6** | 0 | 6 | 2 | 8 | 4 | 0 | 6 | 2 | 8 | 4 |
| **7** | 0 | 7 | 4 | 1 | 8 | 5 | 2 | 9 | 6 | 3 |
| **8** | 0 | 8 | 6 | 4 | 2 | 0 | 8 | 6 | 4 | 2 |
| **9** | 0 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

# Aside: Finding gcd(*a*, *b*)

- Factor *a* and *b* into primes
- For each prime factor that appears in both *a* and *b*'s list, take the smallest exponent, and combine all.
- Example gcd(250, 100)

$250=2*5^3$ $100=2^2*5^2$ $\gcd(250, 100)=2*5^2=50$



# Finding gcd(a,b)

- Prime factorization is slow
- Faster way. Use the fact:
  - gcd(a,b)=gcd(b, a mod b)



# Aside: The φ function

- Euler's φ (Phi) function
- φ(n) = the number of integers < n which are relatively prime to n
- If n is large, it's hard to calculate φ(n)
- How hard?
  - no easier than factoring n



# Aside: GCD

- Review: gcd(*a*, *b*) is the largest positive integer which divides *a* and *b*
- *Examples:* gcd(12, 8)=4; gcd(7, 3)=1
- If gcd(*x*, *y*) = 1, *x* and *y* are *relatively prime*
- Finding gcd(*a*, *b*). Two ways:
  - 1) factor into primes, for each prime that appears in both *a*'s and *b*'s list, look at the smallest exponent that appears in each. Example follows.
  - 2) use the Euclidean Algorithm

## Aside: Finding Inverses mod n

- Use the Extended Euclidean Algorithm

$$\begin{aligned} 192 &= (17)11+5 \\ 11 &= 2(5)+1 \\ 5 &= 5(1)+0 \end{aligned}$$



## Aside: Finding Inverses mod n

- Use the Extended Euclidean Algorithm

$$\begin{aligned} 192 &= (17)11+5 \\ 11 &= 2(5)+1 \\ 5 &= 5(1)+0 \end{aligned}$$

*gcd(192,11) = 1*



## Aside: gcd(a,b) with Euclidean Algo.

- Do the following steps:

$$\begin{aligned} a &= q_1 b + r_1 \\ b &= q_2 r_1 + r_2 \\ r_1 &= q_3 r_2 + r_3 \\ &\ddots \\ r_{k-2} &= q_k r_{k-1} + r_k \\ r_{k-1} &= q_{k+1} r_k \end{aligned}$$

- but it makes more sense with an example



## Euclidean Algorithm Example

- Find the gcd of 193 and 7
- Do the Euclidean algorithm

193=27*7+4

7=1*4+3

4=1*3+1

3=3*1

- So gcd(193,7) = 1

## RSA Key Generation

- choose large primes p, q;   $p \neq q$
  - $p = 17$, $q=13$
- calculate $n = pq$
- calculate $\varphi(n) = (p-1)(q-1)$
- choose e where $\gcd(e, \varphi(n))=1$;    $1 < e < \varphi(n)$
- choose d where $d = e^{-1} \bmod \varphi(n)$;



## RSA Key Generation

- choose large primes p, q;   $p \neq q$
  - $p = 17$, $q=13$
- calculate $n = pq$
- calculate $\varphi(n) = (p-1)(q-1)$
- choose e where $\gcd(e, \varphi(n))=1$;    $1 < e < \varphi(n)$
- choose d where $d = e^{-1} \bmod \varphi(n)$;



## Aside:  Finding Inverses mod n

- Use the Extended Euclidean Algorithm

$$\begin{aligned} 192 &= (17)11+5 \\ 11 &= 2(5)+1 \\ 5 &= 5(1)+0 \end{aligned}$$

- Now work backwards:

$$\begin{aligned} 1 &= 11-2(5) \\ &= 11-2(192-17(11)) \\ &= 11-2(192)+34(11) \\ &= 35(11)-2(192) \end{aligned}$$



## Aside:  Finding Inverses mod n

- Use the Extended Euclidean Algorithm

$$\begin{aligned} 192 &= (17)11+5 \\ 11 &= 2(5)+1 \\ 5 &= 5(1)+0 \end{aligned}$$

- Now work backwards:

*inverse of 11 mod 192*

$$\begin{aligned} 1 &= 11-2(5) \\ &= 11-2(192-17(11)) \\ &= 11-2(192)+34(11) \\ &= 35(11)-2(192) \end{aligned}$$

## RSA Key Generation

- choose large primes p, q; $p \neq q$
  - $p = 17$, $q=13$
- calculate $n = pq$
  - $17*13 = 221$
- calculate $\varphi(n) = (p-1)(q-1)$
  - $16*12 = 192$
- choose e where $\gcd(e, \varphi(n))=1$; $1 < e < \varphi(n)$
- choose d where $d = e^{-1} \bmod \varphi(n)$;



## RSA Key Generation

- choose large primes p, q; $p \neq q$
  - $p = 17$, $q=13$
- calculate $n = pq$
  - $17*13 = 221$
- calculate $\varphi(n) = (p-1)(q-1)$
  - $16*12 = 192$
- choose e where $\gcd(e, \varphi(n))=1$; $1 < e < \varphi(n)$
- choose d where $d = e^{-1} \bmod \varphi(n)$;

*then throw away p and q.*
*we don't need them anymore.*

*(and if someone found them,*
*they'd be able to figure out d)*



## RSA Key Generation

- choose large primes p, q; $p \neq q$
  - $p = 17$, $q=13$
- calculate $n = pq$
  - $17*13 = 221$
- calculate $\varphi(n) = (p-1)(q-1)$
- choose e where $\gcd(e, \varphi(n))=1$; $1 < e < \varphi(n)$
- choose d where $d = e^{-1} \bmod \varphi(n)$;



## RSA Key Generation

- choose large primes p, q; $p \neq q$
  - $p = 17$, $q=13$
- calculate $n = pq$
  - $17*13 = 221$
- calculate $\varphi(n) = (p-1)(q-1)$
- choose e where $\gcd(e, \varphi(n))=1$; $1 < e < \varphi(n)$
- choose d where $d = e^{-1} \bmod \varphi(n)$;

# RSA Key Generation

- choose large primes p, q; p≠q
  - p = 17, q=13
- calculate n = pq
  - 17*13 = 221
- calculate φ(n) = (p-1)(q-1)
  - 16*12 = 192
- choose e where gcd(e, φ(n))=1; 1 < e < φ(n)
  - choose 11
- choose d where d = $e^{-1}$ mod φ(n);
  - pick 35



# RSA Key Generation

- choose large primes p, q; p≠q
  - p = 17, q=13
- calculate n = pq
  - 17*13 = 221
- calculate φ(n) = (p-1)(q-1)
  - 16*12 = 192
- choose e where gcd(e, φ(n))=1
  - choose 11
- choose d where d = $e^{-1}$ mod φ(n);
  - pick 35

*encryption key* = {e,n}
= {11,221}

*decryption key* = {d,n}
= {35,221}



# RSA Key Generation

- choose large primes p, q; p≠q
  - p = 17, q=13
- calculate n = pq
  - 17*13 = 221
- calculate φ(n) = (p-1)(q-1)
  - 16*12 = 192
- choose e where gcd(e, φ(n))=1; 1 < e < φ(n)
  - choose 11
- choose d where d = $e^{-1}$ mod φ(n);



# RSA Key Generation

- choose large primes p, q; p≠q
  - p = 17, q=13
- calculate n = pq
  - 17*13 = 221
- calculate φ(n) = (p-1)(q-1)
  - 16*12 = 192
- choose e where gcd(e, φ(n))=1; 1 < e < φ(n)
  - choose 11
- choose d where d = $e^{-1}$ mod φ(n);

# RSA Example

| | | F | A | M | I | L | Y | G | U | Y |
|---|---|---|---|---|---|---|---|---|---|---|
| | *m* | F | A | M | I | L | Y | G | U | Y |
| | *m numeric* | 5 | 0 | 12 | 8 | 11 | 24 | 6 | 20 | 24 |
| $m^e$ mod n | $m^{11}$ mod 221 | 164 | 0 | 142 | 70 | 97 | 201 | 141 | 41 | 201 |



# RSA Example

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *m* | F | A | M | I | L | Y | G | U | Y |
| | *m numeric* | 5 | 0 | 12 | 8 | 11 | 24 | 6 | 20 | 24 |
| $m^e$ mod n | $m^{11}$ mod 221 | 164 | 0 | 142 | 70 | 97 | 201 | 141 | 41 | 201 |

*ciphertext*



# RSA Example

- Encrypt "FAMILY GUY"
- Use the keys that we generated
  - encryptiong key: {e=11, n=221}
  - decryption key: {d=35, n=221}



# RSA Example

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *m* | F | A | M | I | L | Y | G | U | Y |
| *m numeric* | 5 | 0 | 12 | 8 | 11 | 24 | 6 | 20 | 24 |

# RSA Security

- RSA is thought to be secure because:
  - to find *d* (inverse of *e* mod $\varphi(n)$)
    - need to know $\varphi(n)$
  - given *n* it's very difficult to find $\varphi(n)$
    - thought to be no easier than factoring *n*
- *Note*: when *p* and *q* are 100 decimal digits
  - *n* is about 200 decimal digits
  - millions of years of computer time needed to factor



# Why This Works

$$c^d = (m^e)^d = m^{k\varphi(n)+1} = (m^{\varphi(n)})^k m = (1)^k m = m$$

- all of these are (mod n)



# RSA Example

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *m* | F | A | M | I | L | Y | G | U | Y |
| | *m numeric* | 5 | 0 | 12 | 8 | 11 | 24 | 6 | 20 | 24 |
| $m^e$ mod n | $m^{11}$ mod 221 | 164 | 0 | 142 | 70 | 97 | 201 | 141 | 41 | 201 |
| $c^d$ mod n | $c^{35}$ mod 221 | 5 | 0 | 12 | 8 | 11 | 24 | 6 | 20 | 24 |



# RSA Example

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *m* | F | A | M | I | L | Y | G | U | Y |
| | *m numeric* | 5 | 0 | 12 | 8 | 11 | 24 | 6 | 20 | 24 |
| $m^e$ mod n | $m^{11}$ mod 221 | 164 | 0 | 142 | 70 | 97 | 201 | 141 | 41 | 201 |
| $c^d$ mod n | $c^{35}$ mod 221 | 5 | 0 | 12 | 8 | 11 | 24 | 6 | 20 | 24 |

*Original plaintext is recovered*

# public key crypto difficulties

- key distribution still a problem
  - proving to whom a key belongs
- slow
  - keys must be much longer than symmetric keys to provide the same degree of security
  - hybrid scheme (public + session key) often used
- RSA – size of message to be encrypted is limited by *n*.



# hybrid scheme

- public key crypto is slow
- symmetric key is fast
  - but key distribution problem
- solution:
  - create a symmetric key called *session key*
  - encrypt the data with the session key
  - encrypt the session key with the receiver's public key



# Why This Works

$$c^d = (m^e)^d = m^{k\varphi(n)+1} = (m^{\varphi(n)})^k m = (1)^k m = m$$

- all of these are (mod n)
- because *e* and *d* are inverses mod n
  - $ed \equiv 1 + k\varphi(n)$



# Why This Works

$$c^d = (m^e)^d = m^{k\varphi(n)+1} = (m^{\varphi(n)})^k m = (1)^k m = m$$

- all of these are (mod n)
- because *e* and *d* are inverses mod n
  - $ed \equiv 1 + k\varphi(n)$
- by rules of modular arithmetic (*Fermat* ...)
  - $a \equiv a^p \pmod p \rightarrow 1 \equiv a^{p-1} \pmod p$
  - if p is prime, and a < p

# driver's licenses and digital certificates

- why do you trust a driver's license but not the ID card that I created saying I'm Barack Obama?



# driver's licenses and digital certificates

- why do you trust a driver's license but not the ID card that I created saying I'm Barack Obama?
  - State of PA vouches the picture matches:
    - the name, address, *etc.* of the info on the card
- if you trust PA, you believe info on license
- digital certificate does same for public key



# bogus keys?

- B gets a message claiming to be from A
- message is signed with key claiming to be A's
- signature matches the message
- is B sure that it came from A?
- related question. can I:
  - take a small picture of myself
  - attach it to a card saying that I'm Barack Obama
  - get a free ride on Air Force One?



# Certificates

- A *Certification Authority* verifies that your public key belongs to you
  - e.g. Verisign
- X.509 standard
- Think *Donnie Brasco*

## verifying certificates

- so how can know and trust Verisign's certificate?
  - some certificates are built into browser, OS
  - others can be later added manually
  - be careful which certificates you add



## what happens when things go wrong?

- when a bad guy gets a valid certificate?
  - microsoft/verisign debacle
  - mountain america credit union

- certificate revocation lists



## certificate verifies owner of public key

- go to an HTTPS, *e.g.* tuportal.temple.edu
- click on the lock on the browser
- shows that certificate contains:
  - a public key
  - owner of the key
  - expiration date
- certificate signed by *Certification Authority* (CA)
  - *e.g.* Verisign
- you believe the public key is really Temple's if
  - the certificate is valid
  - you trust Verisign



## verifying certificates

- so how can know and trust Verisign's certificate?