ENCRYPTION

David Kauchak
CS54 – Fall 2022

1

Admin

Assignment 5

Assignment 6

2

Course feedback

https://forms.gle/A9KDkC7jxaMsrHn26

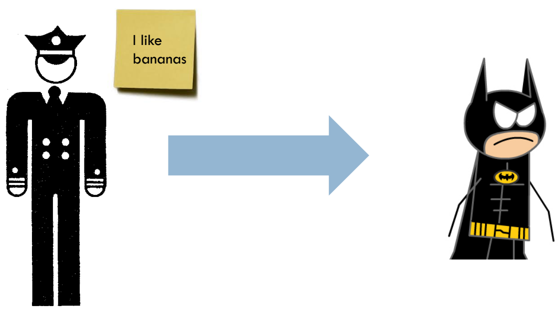
3

Encryption

What is it and why do we need it?

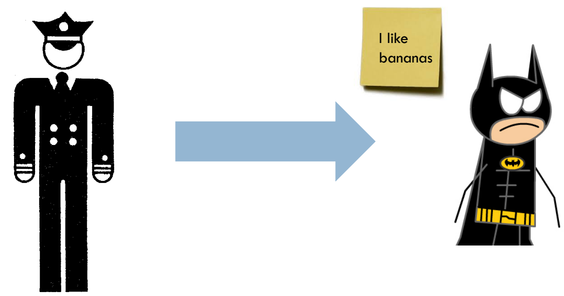
4

Encryption



5

Encryption

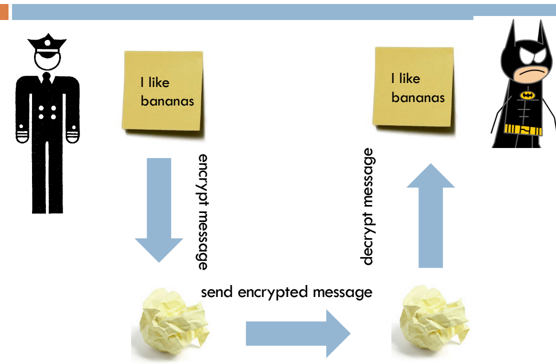


6

Encryption: a bad attempt

7

Encryption: the basic idea



8

Encryption: a better approach

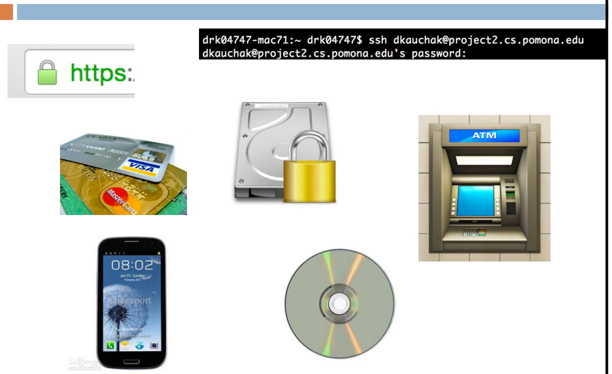
9

Encryption uses

Where have you seen encryption used?

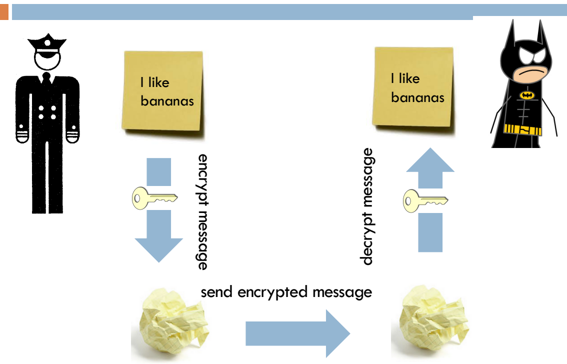
10

Encryption uses



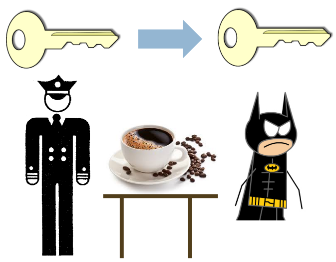
11

Private key encryption



12

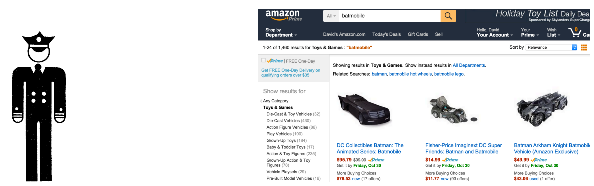
Private key encryption



Any problems with this?

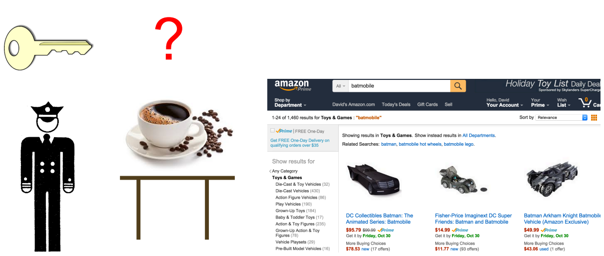
13

Private key encryption



14

Private key encryption



15

Public key encryption

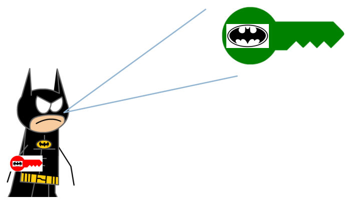
private key

public key

Two keys, one you make publicly available and one you keep to yourself

16

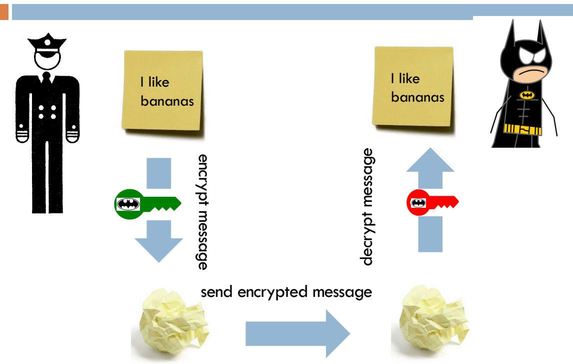
Public key encryption



Share your public key with everyone

17

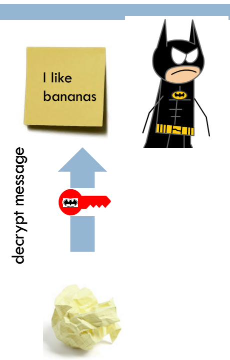
Public key encryption



18

Public key encryption

Only the person with the private key can decrypt!



19

Modular arithmetic

Normal arithmetic:

a = b

a is equal to b *or* *a-b = 0*

Modular arithmetic:

a ≡ b *(mod n)*

a-b = n*k for some integer k or

a = b + n*k for some integer k or

a % n = b % n (where % is the mod operator)

20

Modular arithmetic

Which of these statements are true?

$12 \equiv 5 \ (mod\ 7)$

$52 \equiv 92 \ (mod\ 10)$

$17 \equiv 12 \ (mod\ 6)$

a-b = n*k for some integer k or
a = b + n*k for some integer k or
a % n = b % n (where % is the mod operator)

$65 \equiv 33 \ (mod\ 32)$

21

Modular arithmetic

Which of these statements are true?

$12 \equiv 5 \ (mod\ 7)$
12-5 = 7 = 1*7
12 % 7 = 5 = 5 % 7

$52 \equiv 92 \ (mod\ 10)$
92-52 = 40 = 4*10
92 % 10 = 2 = 52 % 20

$17 \equiv 12 \ (mod\ 6)$
17-12 = 5
17 % 6 = 5
12 % 6 = 0

$65 \equiv 33 \ (mod\ 32)$
65-33 = 32 = 1*32
65 % 32 = 1 = 33 % 32

22

Modular arithmetic properties

If:

$a \equiv b \ (mod\ n)$

then:

a mod n = b mod n

"mod"/remainder operator

congruence (mod n)

23

Modular arithmetic properties

If:

$a \equiv b \ (mod\ n)$

then:

a mod n = b mod n

More importantly:

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

and

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

What do these say?

24

Modular arithmetic examples

(1712 +1637) mod 10 =

25

Modular arithmetic examples

(1712 +1637) mod 10 =

The hard way:

1712 +1637 = 3349

3349 mod 10 = 9

26

Modular arithmetic examples

(1712 +1637) mod 10 =

The easy way:

1712 mod 10 + 1637 mod 10 =

(2 + 7) mod 10 = 9

27

Modular arithmetic examples

(1712*1637) mod 10 =

The easy way:

1712 mod 10 * 1637 mod 10 =

(2 * 7) mod 10 = 4

1712*1637 = 2802544 mod 10 = 4

28

Modular arithmetic

Why talk about modular arithmetic and congruence? How is it useful? Why might it be better than normal arithmetic?

We can limit the size of the numbers we're dealing with to be at most n (if it gets larger than n at any point, we can always just take the result mod n)

The mod operator can be thought of as mapping a number in the range 0 … n-1

29

Modular arithmetic examples

(1712^{237}) mod 10 =

30

Modular arithmetic examples

(1712^{237}) mod 10 =

The hard way:

2189733188915527033845242014775024662365379214649108861079776729377311646
4178863410200431314724639065631582340030916000535491050743393313989255160
6348256002908856782720027938471702516151831261883438208185382676856143035
8555422262025688935645992713224910081777580598384256361226430744486783684
8972183344917544635567789574283214685603416614354211724441199147585377319
0741448045780204468613804157642484533042787410205542844720697624880058469
5095208486453956254338665468200146017457909832081507762047670719732913228
0181637111587444836647339009142865957557814364142769623374883706605049058
2750630252268175042103727092839763924653863693246547423290880175403121554
3907099468990249536971584503074405804732055649986685982347798454659692375
307405181035086429052892112548475692 mod 10

2

31

Modular arithmetic examples

(1712^{237}) mod 10 =

The easy way:

$((1712 \bmod 10)^{237}) \bmod 10 =$

$(2^{237}) \bmod 10 =$

220855883097298041197912187592864814478435487109452369765200775161577472

mod 10 = 2

32

Modular arithmetic examples

$(2^{237}) \bmod 10 =$

$(2^{10} \bmod 10 * 2^{227} \bmod 10) \bmod 10 =$

$(4 * 2^{227} \bmod 10) \bmod 10 =$

$(4 * 4 * 2^{217} \bmod 10) \bmod 10 =$

$(6 * 2^{217} \bmod 10) \bmod 10 =$

$(4 * 2^{207} \bmod 10) \bmod 10 =$

33

GCD

What does GCD stand for?

34

Greatest Common Divisor

gcd(a, b) is the largest positive integer that divides both numbers without a remainder

gcd(25, 15) = ?

35

Greatest Common Divisor

gcd(a, b) is the largest positive integer that divides both numbers without a remainder

gcd(25, 15) = 5

	25	15
Divisors:	25	15
	5	5
	1	3
		1

36

Greatest Common Divisor

gcd(a, b) is the largest positive integer that divides both numbers without a remainder

gcd(100, 52) = ?

37

Greatest Common Divisor

gcd(a, b) is the largest positive integer that divides both numbers without a remainder

gcd(100, 52) = 4

	100	52
Divisors:	100 50 25 20 10 5 4 2 1	52 13 4 2 1

38

Greatest Common Divisor

gcd(a, b) is the largest positive integer that divides both numbers without a remainder

gcd(14, 63) = ?

gcd(7, 56) = ?

gcd(23, 5) = ?

gcd(100, 9) = ?

gcd(111, 17) = ?

39

Greatest Common Divisor

gcd(a, b) is the largest positive integer that divides both numbers without a remainder

gcd(14, 63) = 7

gcd(7, 56) = 7

gcd(23, 5) = 1

gcd(100, 9) = 1

gcd(111, 17) = 1

Any observations?

40

Greatest Common Divisor

When the gcd = 1, the two numbers share no factors/divisors in common

If gcd(a,b) = 1 then a and b are *relatively prime*

This a weaker condition than primality, since any two prime numbers are also relatively prime, but not vice versa

41

Greatest Common Divisor

A useful property:

If two numbers, a and b, are relatively prime (i.e. gcd(a,b) = 1), then there exists a c such that

a*c mod b = 1

42

RSA public key encryption

Have you heard of it?

What does it stand for?

43

RSA public key encryption

RSA is one of the most popular public key encryption algorithms in use

RSA = Ron Rivest, Adi Shamir and Leonard Adleman

44

RSA public key encryption

1. Choose a bit-length k
   Security increases with the value of k, though so does computation
2. Choose two primes p and q which can be represented with at most k bits
3. Let $n = pq$ and $\phi(n) = (p-1)(q-1)$
   φ() is called *Euler's totient function*
4. Find d such that $0 < d < n$ and $\gcd(d,\phi(n)) = 1$
5. Find e such that $de \bmod \phi(n) = 1$
   Remember, we know one exists!

45

RSA public key encryption

p: prime number
q: prime number
n = pq

$\phi(n) = (p-1)(q-1)$
d: $0 < d < n$ and $\gcd(d,\phi(n)) = 1$
e: $de \bmod \phi(n) = 1$

Given this setup, you can prove that given a number m:

$$(m^e)^d = m^{ed} = m \pmod{n}$$

What does this do for us, though?

46

RSA public key encryption

p: prime number
q: prime number
n = pq

$\phi(n) = (p-1)(q-1)$
d: $0 < d < n$ and $\gcd(d,\phi(n)) = 1$
e: $de \bmod \phi(n) = 1$

Given this setup, you can prove that given a number m:

m message

What does this do for us, though?

47

RSA public key encryption

p: prime number
q: prime number
n = pq

$\phi(n) = (p-1)(q-1)$
d: $0 < d < n$ and $\gcd(d,\phi(n)) = 1$
e: $de \bmod \phi(n) = 1$

Given this setup, you can prove that given a number m:

(m^e) encrypted message

What does this do for us, though?

48

RSA public key encryption

p: prime number
q: prime number
n = pq

$\phi(n) = (p-1)(q-1)$
d: $0 < d < n$ and $gcd(d,\phi(n)) = 1$
e: $de \bmod \phi(n) = 1$

Given this setup, you can prove that given a number *m*:

$(m^e)^d = m^{ed} = m \pmod{n}$ decrypted message

What does this do for us, though?

49

RSA public key encryption

p: prime number
q: prime number
n = pq

$\phi(n) = (p-1)(q-1)$
d: $0 < d < n$ and $gcd(d,\phi(n)) = 1$
e: $de \bmod \phi(n) = 1$

private key | public key

(d, n) | (e, n)

50

RSA encryption/decryption

private key | public key

(d, n) | (e, n)

You have a number *m* that you want to send encrypted

$encrypt(m) = m^e \bmod n$ (uses the public key)

How does this encrypt the message?

51

RSA encryption/decryption

private key | public key

(d, n) | (e, n)

You have a number *m* that you want to send encrypted

$encrypt(m) = m^e \bmod n$ (uses the public key)

- Maps *m* onto some number in the range 0 to *n*-1
- If you vary e, it will map to a different number
- Therefore, unless you know d, it's hard to know what the original *m* was after the transformation

52

RSA encryption/decryption

private key: (d, n) public key: (e, n)

You have a number *m* that you want to send encrypted

encrypt(m) = m^e mod n (uses the public key)

decrypt(z) = z^d mod n (uses the private key)

Does this work?

53

RSA encryption/decryption

encrypt(m) = m^e mod n

decrypt(z) = z^d mod n

decrypt(z) = decrypt(m^e mod n) z is some encrypted message

= $(m^e \text{ mod } n)^d$ mod n definition of decrypt

= $(m^e)^d$ mod n modular arithmetic

= m mod n $(m^e)^d = m^{ed} = m \pmod n$

Did we get the original message?

54

RSA encryption/decryption

encrypt(m) = m^e mod n

decrypt(z) = z^d mod n

decrypt(z) = decrypt(m^e mod n) z is some encrypted message

= $(m^e \text{ mod } n)^d$ mod n definition of decrypt

= $(m^e)^d$ mod n modular arithmetic

= m mod n $(m^e)^d = m^{ed} = m \pmod n$

If $0 \leq m < n$, yes!

55

RSA encryption: an example

p: prime number
q: prime number
n = pq

$\phi(n) = (p-1)(q-1)$
d: $0 < d < n$ and $\gcd(d, \phi(n)) = 1$
e: $de \text{ mod } \phi(n) = 1$

p = 3
q = 13
n = ?
$\phi(n)$ = ?
d = ?
e = ?

56

RSA encryption: an example

p: prime number
q: prime number
n = pq

$\phi(n)$ = (p-1)(q-1)
d: 0 < d < n and gcd(d,$\phi(n)$) = 1
e: de mod $\phi(n)$ = 1

p = 3
q = 13
n = ?

57

RSA encryption: an example

p: prime number
q: prime number
n = pq

$\phi(n)$ = (p-1)(q-1)
d: 0 < d < n and gcd(d,$\phi(n)$) = 1
e: de mod $\phi(n)$ = 1

p = 3
q = 13
n = 3*13 = 39

58

RSA encryption: an example

p: prime number
q: prime number
n = pq

$\phi(n)$ = (p-1)(q-1)
d: 0 < d < n and gcd(d,$\phi(n)$) = 1
e: de mod $\phi(n)$ = 1

p = 3
q = 13
n = 39
$\phi(n)$ = ?

59

RSA encryption: an example

p: prime number
q: prime number
n = pq

$\phi(n)$ = (p-1)(q-1)
d: 0 < d < n and gcd(d,$\phi(n)$) = 1
e: de mod $\phi(n)$ = 1

p = 3
q = 13
n = 39
$\phi(n)$ = 2*12 = 24

60

RSA encryption: an example

p: prime number
q: prime number
n = pq

$\phi(n)$ = (p-1)(q-1)
d: 0 < d < n and gcd($d,\phi(n)$) = 1
e: $de \bmod \phi(n) = 1$

p = 3
q = 13
n = 39
$\phi(n)$ = 24
d = ?
e = ?

61

RSA encryption: an example

p: prime number
q: prime number
n = pq

$\phi(n)$ = (p-1)(q-1)
d: 0 < d < n and gcd($d,\phi(n)$) = 1
e: $de \bmod \phi(n) = 1$

p = 3
q = 13
n = 39
$\phi(n)$ = 24
d = 5
e = 5

62

RSA encryption: an example

p: prime number
q: prime number
n = pq

$\phi(n)$ = (p-1)(q-1)
d: 0 < d < n and gcd($d,\phi(n)$) = 1
e: $de \bmod \phi(n) = 1$

p = 3
q = 13
n = 39
$\phi(n)$ = 24
d = 5
e = 29

63

RSA encryption: an example

n = 39
d = 5
e = 29

encrypt(m) = m^e mod n

decrypt(z) = z^d mod n

encrypt(10) = ?

64

RSA encryption: an example

n = 39
d = 5
e = 29

encrypt(m) = m^e mod n

decrypt(z) = z^d mod n

encrypt(10) = 10^{29} mod 39 = 4

65

RSA encryption: an example

n = 39
d = 5
e = 29

encrypt(m) = m^e mod n

decrypt(z) = z^d mod n

encrypt(10) = 10^{29} mod 39 = 4

decrypt(4) = ?

66

RSA encryption: an example

n = 39
d = 5
e = 29

encrypt(m) = m^e mod n

decrypt(z) = z^d mod n

encrypt(10) = 10^{29} mod 39 = 4

decrypt(4) = 4^5 mod 39 = 10

67

RSA encryption: an example

n = 39
d = 5
e = 5

encrypt(m) = m^e mod n

decrypt(z) = z^d mod n

encrypt(2) = ?

68

RSA encryption: an example

n = 39
d = 5
e = 5

encrypt(m) = m^e mod n

decrypt(z) = z^d mod n

encrypt(2) = 2^5 mod 39 = 32 mod 39 = 32

decrypt(32) = ?

69

RSA encryption: an example

n = 39
d = 5
e = 5

encrypt(m) = m^e mod n

decrypt(z) = z^d mod n

encrypt(2) = 2^5 mod 39 = 32 mod 39 = 32

decrypt(32) = 32^5 mod 39 = 2

70

RSA encryption in practice

For RSA to work: $0 \leq m < n$

What if our message isn't a number?

What if our message is a number that's larger than n?

71

RSA encryption in practice

For RSA to work: $0 \leq m < n$

What if our message isn't a number?
- We can always convert the message into a number (remember everything is stored in binary already somewhere!)

What if our message is a number that's larger than n?
- Break it into n sized chunks and encrypt/decrypt those chunks

72

RSA encryption in practice

encrypt("I like bananas") =

0101100101011100 …	encode as a binary string (i.e. number)
4, 15, 6, 2, 22, …	break into multiple < n size numbers
17, 1, 43, 15, 12, …	encrypt each number

73

RSA encryption in practice

decrypt((17, 1, 43, 15, 12, …)) =

4, 15, 6, 2, 22, …	decrypt each number
0101100101011100 …	put back together
"I like bananas"	turn back into a string (or whatever the original message was)

Often encrypt and decrypt just assume sequences of bits and the interpretation is done outside

74