

- HW questions
- Visual Cryptography
- Number theory review
- RSA Example
- One-way Functions
  - Digital Signatures
- Other applications of Public Key Encryption:
  - Key Distribution and Management

*CSCI283 Fall 2003 Lecture 3*  
*GWU*

# Announcements

# Conferences

- Call for papers:

Computers, Freedom, Privacy, 2004 (Oakland, CA).

Student paper competition

<http://cfp2004.org/StudentCompetition.html>

- ACM Computer and Communications Security

ACM CCS

Windham City Centre Hotel, DC

October 27-31

[www.acm.org/sigs/sigsac/ccs/CCS2003](http://www.acm.org/sigs/sigsac/ccs/CCS2003)

Full time students interested in registering contact me

# GWU Writing Centre

For those writing lots of term papers: engineers/CS students have often not had much help writing papers, GWU writing centre provides free help

[www.gwu.edu/~gwriter/](http://www.gwu.edu/~gwriter/)

No office hours tomorrow  
courtesy: Isabel

Decrypting scrambling

how many miles must a man walk on before  
he can call himself a man

hmøeuanlnføccøsa  
oamssøøkøohaahean  
wniøtmwøbrenliløø  
øylmøaaøeeøølmfmø

hmøeuanlnføccøsaøoamssøøkøohaaheanwniøtmw  
øbrenliløøøylmøaaøeeøølmfmø

68 characters

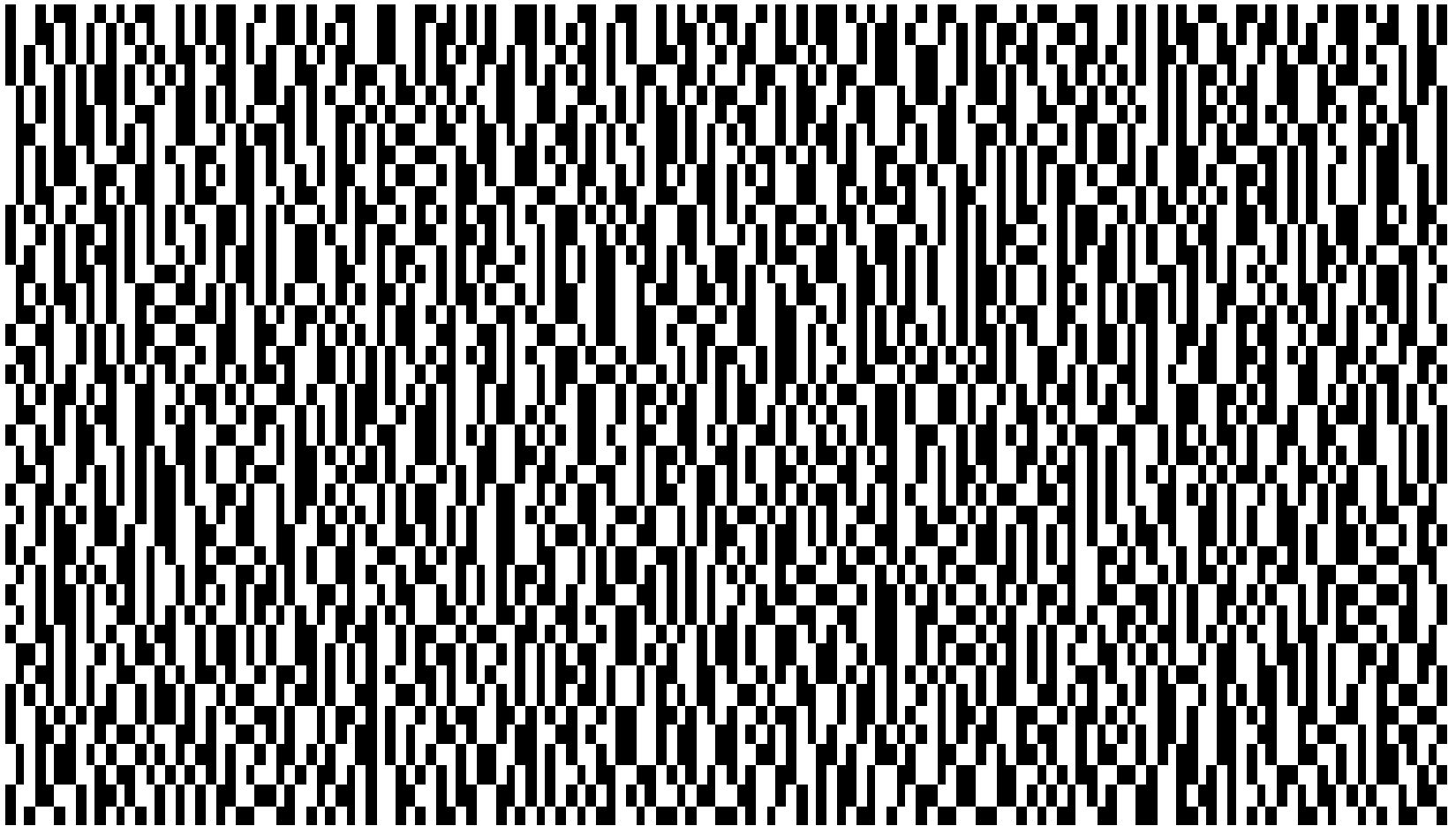
$68/4 = 17$

If cut up the above single string into 4 strings of  
size 17 each, and read column by column, get answer

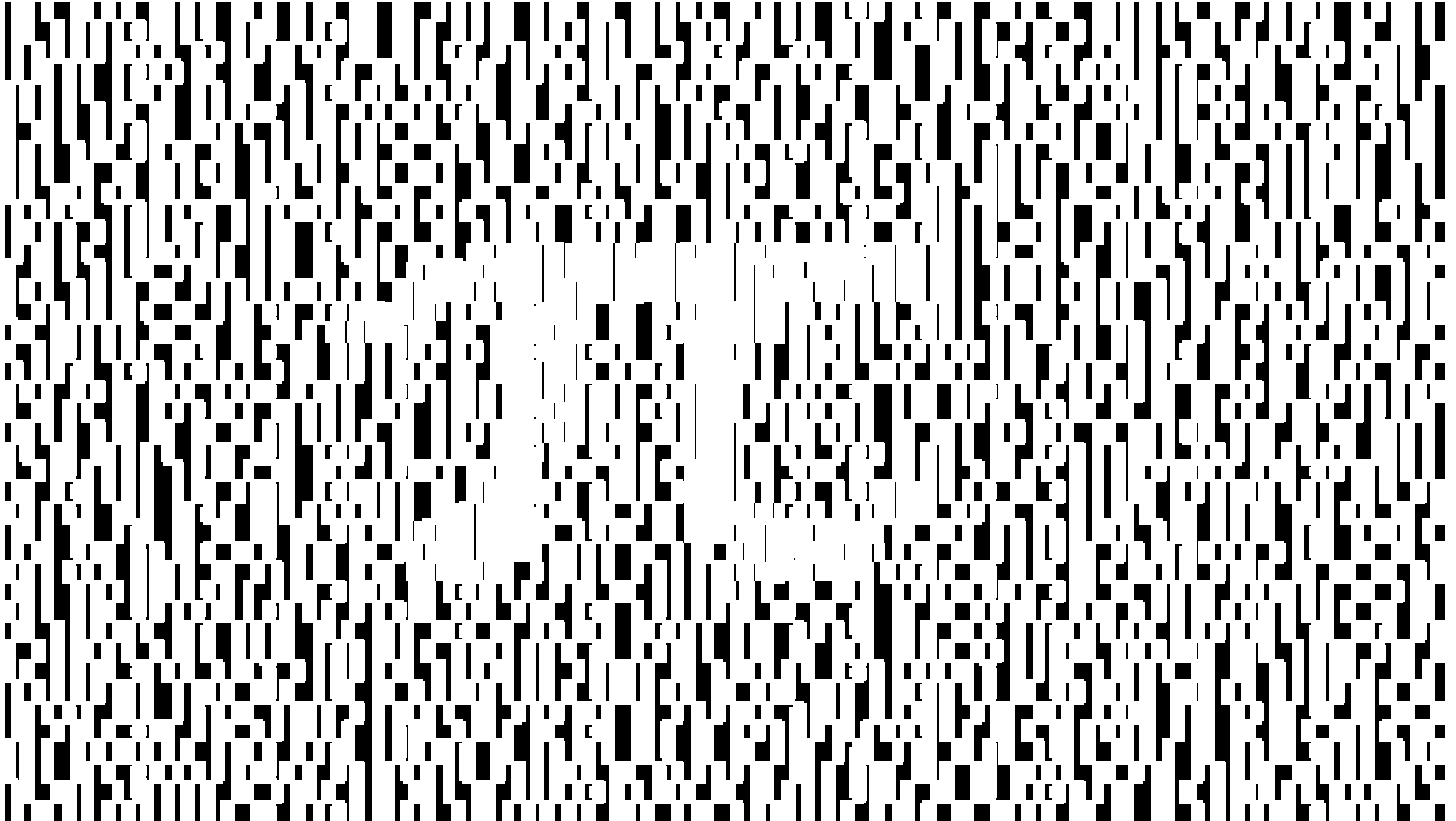
Visual Crypto  
from Doug Stinson website

<http://www.cacr.math.uwaterloo.ca/~dstinson/visual.html>

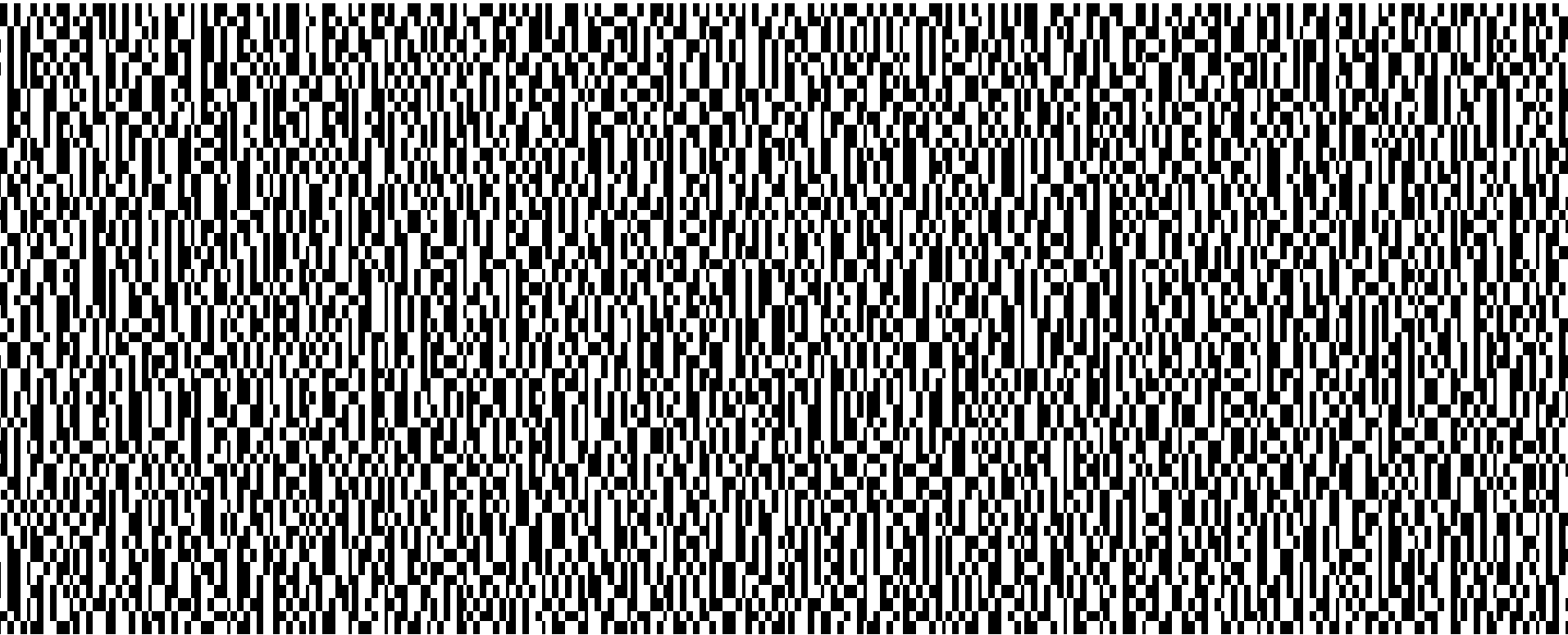
# One time pad, visually, with a twist



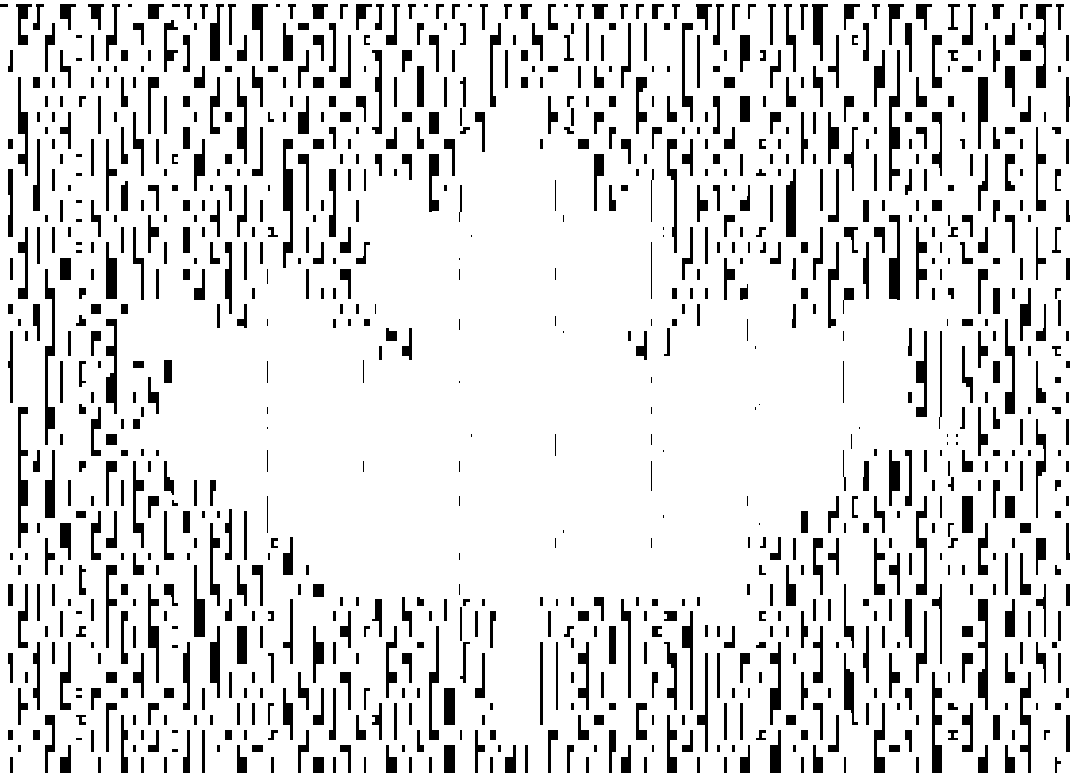
# Add other random component



# First



# Second



# Applications

- Newest (early 2003): Voting receipts (David Chaum)
- Electronic voting requires some physical receipt for voter
- But he cannot take any proof outside booth of how he voted
- Suppose he gets printed out two visual crypto slides that when superimposed show his votes
- He leaves with one of those, which also contain other information in encrypted form

# Number theory review

(from Memon's notes, Brooklyn Poly)

Divisors, Prime numbers, Relative primes, Modular arithmetic

# Divisors

- $x$  divides  $y$  (written  $x \mid y$ ) if the remainder is 0 when  $y$  is divided by  $x$ 
  - $1 \mid 8, 2 \mid 8, 4 \mid 8, 8 \mid 8$
- The divisors of  $y$  are the numbers that divide  $y$ 
  - divisors of 8:  $\{1, 2, 4, 8\}$
- For every number  $y$ 
  - $1 \mid y$
  - $y \mid y$

# Common divisors

- The common divisors of two numbers  $x, y$  are the numbers  $z$  such that  $z|x$  and  $z|y$ 
  - common divisors of 8 and 12:
    - intersection of  $\{1,2,4,8\}$  and  $\{1,2,3,4,6,12\}$
    - $= \{1,2,4\}$
- greatest common divisor:  $\gcd(x,y)$  is the number  $z$  such that
  - $z$  is a common divisor of  $x$  and  $y$
  - no common divisor of  $x$  and  $y$  is larger than  $z$ 
    - $\gcd(8,12) = 4$

# Prime numbers and Relative Primes

- A number is prime if its only divisors are 1 and itself:
  - 2,3,5,7,11,13,17,19, ...
- $x$  and  $y$  are relatively prime if they have no common divisors, other than 1
- Equivalently,  $x$  and  $y$  are relatively prime if  $\gcd(x,y) = 1$ 
  - 9 and 14 are relatively prime
  - 9 and 15 are not relatively prime

# Modular Arithmetic

- Definition:  $x \equiv y \pmod{m}$ , if  $x$  and  $y$  have the same remainder when divided by  $m$ .

*Example:*  $14 \equiv 25 \pmod{11}$

- We work in  $Z_m = \{0, 1, 2, \dots, m-1\}$ , the ring of integers modulo  $m$  with binary operators  $+$  and  $*$  defined modulo  $m$ .

*Example:*  $Z_9 = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$

- We abuse notation and often write  $=$  instead of

# Mathematical Operations

Calculate the following:

- $(5 + 4) \bmod 6$

$3 \bmod 6$

- $(5 - 6) \bmod 7$

$-1 \bmod 7 = 6 \bmod 7$

- $(3 \times 9) \bmod 5$

$27 \bmod 5 = 2 \bmod 5$

- $(15 \div 3) \bmod 16$

$5 \bmod 16$

- $3^3 \bmod 7$

$27 \bmod 7 = 6 \bmod 7$

- $\sqrt{2} \bmod 7$

$2 \times 2 = 4 \neq 2$ ;  $3 \times 3 = 9 = 2 \bmod 7$ .  $\sqrt{2} \bmod 7 = 3 \bmod 7$

Also:  $4 \times 4 = 16 = 2 \bmod 7$ , and  $\sqrt{2} \bmod 7 = 4$  as well

Observe that  $4 = -3 \bmod 7$

# Efficient exponentiation (from Memon notes)

Usual approach to computing  $x^c \bmod n$  is inefficient when  $c$  is large.  
Example:  $5^{39}$  involves 39 multiplications mod  $n$

Instead, represent  $c$  as bit string  $b_{k-1} \dots b_0$  and use the following algorithm:

$$z = 1$$

*For  $i = k-1$  downto  $0$  do*

$$z = z^2 \bmod n$$

$$\text{if } b_i = 1 \text{ then } z = z \times x \bmod n$$

How many multiplications?  $k = 2\lceil \log_2 c \rceil$

# Example

Calculate  $5^{39} \bmod 7$  efficiently

$$39 = 100111 = 2^5 + 2^2 + 2^1 + 2^0$$

$$5^{39} = (((((5^2)^2)^2)^2)^2 \times (5^2)^2 \times 5^2 \times 5^1$$

How many multiplications did you need?

$$5^{39} \bmod 7$$

$i$	$b_i$	$z$	
5	1	1	5
4	0	$25 \bmod 7 = 4$	4
3	0	$16 \bmod 7 = 2$	2
2	1	4	$20 \bmod 7 = 6$
1	1	$36 \bmod 7 = 1$	5
0	1	$25 \bmod 7 = 4$	$20 \bmod 7 = 6$

# RSA Again

# RSA Example

- Large primes  $p$  and  $q$

5, 7

- Their product  $n$

35

- A number  $e$  relatively prime to  $(p-1)(q-1)$

11

- A number  $d$  such that  $x^{ed} = 1 \pmod n \quad \forall x$

(In fact,  $d$  such that  $ed = 1 \pmod{(p-1)(q-1)}$ )

$d$  such that  $11d = 1 \pmod{24}$

$d = 11$

## RSA Example contd.

- Encryption with  $e$

$$c = m^e$$

$$m = 3 \text{ mod } 35$$

$$c = 3^{11} \text{ mod } 35 = 12 \text{ mod } 35$$

*In class, using previous algorithm, we had an apparent contradiction – but we were calculating  $11^{11}$  instead of  $3^{11}$*

- Decryption

$$m = c^d;$$

$$m = 12^{11} \text{ mod } 35 = 3 \text{ mod } 35$$

# Applications of public key encryption

- Digital signatures
- Certificates and user authentication
- Key management and exchange

# Applications of Public Key Encryption: Digital Signatures

# Digital Signatures

- A digital signature is:
  - a string  $y$  sent along with
  - a message  $x$such that  
the receiver can use  $y$  to determine that
  - the sender did indeed send  $x$  and
  - it was not changed along the way
- Provides:
  - Sender Authentication (protects against impersonation attack)
  - Message integrity (protects against substitution attack)

# RSA can be used for digital signatures

$y$  is sent along with  $x$ ,

- the signature of  $x$ ,
- encrypted using a private key by message sender/creator

$$y = e_K(x) = x^b \bmod n;$$

Signature verification is done by

- decrypting  $y$  with the public key and
- verifying that the decryption is indeed  $x$ .

# Would prefer if

- The signature were smaller instead of being as long as message
- Need functions:
  - easy to compute, but
  - difficult to undo/invert if some information is unknown (just as in locking a door - easy to lock, difficult to unlock without the key)
- Motivates the study of *one-way functions*

One-way functions:  
easy to compute, difficult to invert

# One-Way Functions

$f: D \rightarrow R$  is *one-way* if

1.  $f(x)$  is *easy to compute* for every  $x$  in  $D$

or, more precisely:

1.  $f \in P$

Note:  $f$  one-to-one  $\Rightarrow$  encryption/decryption;

$f$  many-to-one  $\Rightarrow$  message authentication/message digest/secure hash

## One-Way Functions - contd.

2. Given  $f(x) = y$ , determining  $x$  *is not feasible for a large proportion* of  $y$  in  $R$

Instead, can we say:

2. Given  $f(x) = y$ , no polynomial-time algorithm is known for determining  $x$

stronger still:

2. Given  $f(x) = y$ , determining  $x$  *is NP-complete*

# One-way function, example 1

$$f(m,n) = mn$$

Inverting  $f$  is not known to be in  $P$ , i.e. **no known polynomial-time algorithm for factoring in the worst-case** (size of data set is number of bits)

BUT even numbers and numbers divisible by 3 are readily detectable

Though **one-way in a sense, not really secure**

## One-way function, example 2

$$f(p, q) = pq$$

$p, q$ , large primes

Inverting  $f$  is not known to be in  $P$ , i.e. **no known polynomial-time algorithm for factoring** (size of data set is number of bits)

Further, the inversion is known to be 'usually hard'

**one-way and more secure**

# Attacks

- One-way functions can be used for message signatures/authenticators. *Note: The one-way function will be many-to-one*
- Matching a specific signature with a randomly generated message requires at worst  $2^b$  attempts where  $b$  is the number of bits in a signature
- Example: choose one person of a group of 23, the probability that another person from the group will have the same birthday as this person is  $1 - (364/365)^{22} \approx 0.06$  (Low)

# Birthday Attack

- Problem '*birthday attack*' on signature: if it is easy to find two random messages that map to the same signature then a *birthday attack* is easy
- Example: the probability of 2 people having the same birthday in a group of 23 people is more than 0.5
- Difference from previous: did not pick a specific person's birthday to match

## Birthday Attack - description

- Suppose there is a high enough probability that **of  $k$  randomly chosen messages at least two will map to the same authenticator**
  - (i.e. finding two messages that map to the same authenticator is easy).
- The attacker selects two messages:
  - one he wants to get sent,
  - one the sender is likely to sign.

# Birthday Attack - description

- The attacker then
  - generates  $k$  innocent-looking variations of each of the two messages
  - till he finds one from each set that map to the same authenticator.
- Of these two, he gets the sender to sign the one she is more likely to sign.

# Birthday Attack - Implications for size of message digest

The number of random attempts for a birthday attack is of the order of  $\sqrt{n}$  where  $n$  is the number of total messages;

$n = 2^b$  where  $b$  is the number of bits in an authenticator or digest.

Hence, signatures should be of length at least 128

# One-way functions - *collision resistance*

- For one-way functions need:
  - *collision resistance*: it is 'hard' to find values  $x_1$  and  $x_2$  such that  $f(x_1) = f(x_2)$
- Hash functions?
- Collision resistance required for one-way (secure) hash functions, where
  - $y$  is used as an index for  $x$
  - $x$  is stored in a hash table,
  - the length of the hash table is much smaller than the size of the set  $x$  comes from.

# Cryptographic Hash Functions

- SHA – Secure Hash Algorithm
- RIPEM
- MD4
- MD5
- MD6
- Etc.

Applications of  
Public Key Encryption and One-way Functions:  
Digital Signatures

## Recall:

Send  $y = e_K(x)$  with  $x$

Signature verification is done by decrypting  $y$  with the public key and verifying that the decryption is indeed  $x$ .

Would prefer if the signature were smaller instead of being as long as message

# Public key and digital signatures

- *Encrypt Digest(x) instead of x*

- Signature Creation by sender S:

$$x \rightarrow \text{Digest}(x) \rightarrow y = e_{\text{Private}}(\text{Digest}(x))$$

# Public key and digital signatures

- Signature Verification

Given  $(X, Y)$  sent by sender  $S$ , check that  $X$  was indeed sent by  $S$  and has not been changed along the way

$$d_{\text{Public}}(Y) ? \text{Digest}(X)$$

If not equal

- $\text{Digest}(X)$  is incorrect, i.e. message was not  $X$  OR
- $d_{\text{Public}}$  is incorrect, i.e. Sender is not  $S$

# Digital Signature Standard (DSS) (Memon's slides)

- Adopted as standard in 1994
- We do not study DSS in this course.

# Digital Signatures – signing and verification

## Digital Signatures – Signing.

- Alice signs  $m$  to get

$$S_{\text{private}(A)}(m) = E_{\text{private}(A)}(h(m))$$

- She then encrypts with Bob's public key to get

$$E_{\text{public}(B)}[m \parallel S_{\text{private}(A)}(m)].$$

# Signature Verification

- Bob decrypts with private key to get

$$D_{\text{private}(B)} E_{\text{public}(B)}[m \parallel a] = m \parallel a$$

- Bob then verifies Alice's signature with her public key to get

$$D_{\text{public}(A)}[a] ? h(m)$$

- It should match, as it would if  $a = S_{\text{private}(A)}(m)$

# Replay attack

- The message can be repeatedly sent and does not need to be resigned.
- “Give Oscar \$1000 on my behalf. I will pay you back”  
- Alice.
- Ways of avoiding.

# Applications of Public Key Encryption: Key Exchange and Management (from Memon notes)

# Key Exchange

- So far we have assumed either
  - Alice and Bob share a secret key which is unknown to attacker.
  - Alice has a “trusted” copy of Bob’s public key.
- But how does this happen in the first place?!!
- Alice and Bob meet and manually exchange key.
- We need *key distribution* and *key agreement* protocols.

# Protocol I

## Session Key Exchange With KDC.

- Protocol assumes that Alice and Bob share a session (private) key each  $K_A$  and  $K_B$  with a Key Distribution Center (KDC).
  - Alice calls Trent (Trusted KDC) and requests another session key to communicate with Bob.
  - Trent generates random session key  $K$  and sends  $E_{K_A}(K)$  to Alice and  $E_{K_B}(K)$  to Bob.
  - Alice and Bob decrypt with  $K_A$  and  $K_B$  respectively to get  $K$ .
- This is a key distribution protocol.
- Susceptible to replay attack!

# Proof of Knowledge (POK)

- If a user can prove she holds a number (usually a key) without revealing it, she has provided a *proof of knowledge* (of the number)
- Usually used to demonstrate one holds a private key

# Protocol II

## Session Key Exchange With KDC - 1

- $A \rightarrow KDC \quad ID_A \parallel ID_B \parallel N_1$   
(Hello, I am Alice, I want to talk to Bob, I need a session Key and here is a random nonce identifying this request)
- $KDC \rightarrow A \quad E_{K_A}(K \parallel ID_B \parallel N_1 \parallel E_{K_B}(K \parallel ID_A))$   
Encrypted(Here is a key, for you to talk to Bob as per your request  $N_1$  and also an envelope to Bob containing the same key)
- $A \rightarrow B \quad E_{K_B}(K \parallel ID_A)$  (Alice does not know  $E_{K_B}$ )  
(I would like to talk using key in envelope sent by KDC)

# Protocol II – contd.

## Session Key Exchange With KDC - 2

- B -> A       $E_K(N_2)$   
(OK Alice, But can you prove to me that you are indeed Alice and know the key?)
- A -> A       $E_K(f(N_2))$   
(Sure I can!)
- Last two steps - challenge-response. Commonly used to thwart replay attack.
- Why f? Why random  $N_2$ ?

# Protocol II: Protection against replay attacks

- Random  $N_2$  provides Bob with protection against somebody who knows the encrypted value of a single fixed  $N_2$
- $f$  provides Alice with protection from someone who is trying a known-plaintext attack, making her encrypt  $E_K(N_2)$

# References

- Bruce Schneier, *Applied Cryptography*
- Douglas Stinson, *Cryptography Theory and Practice*
- Dominic Welsh, *Cryptography and Codes*
- RSA FAQ <http://www.rsasecurity.com/rsalabs/faq/>