Jan 31, 2005 -- Lecture 7

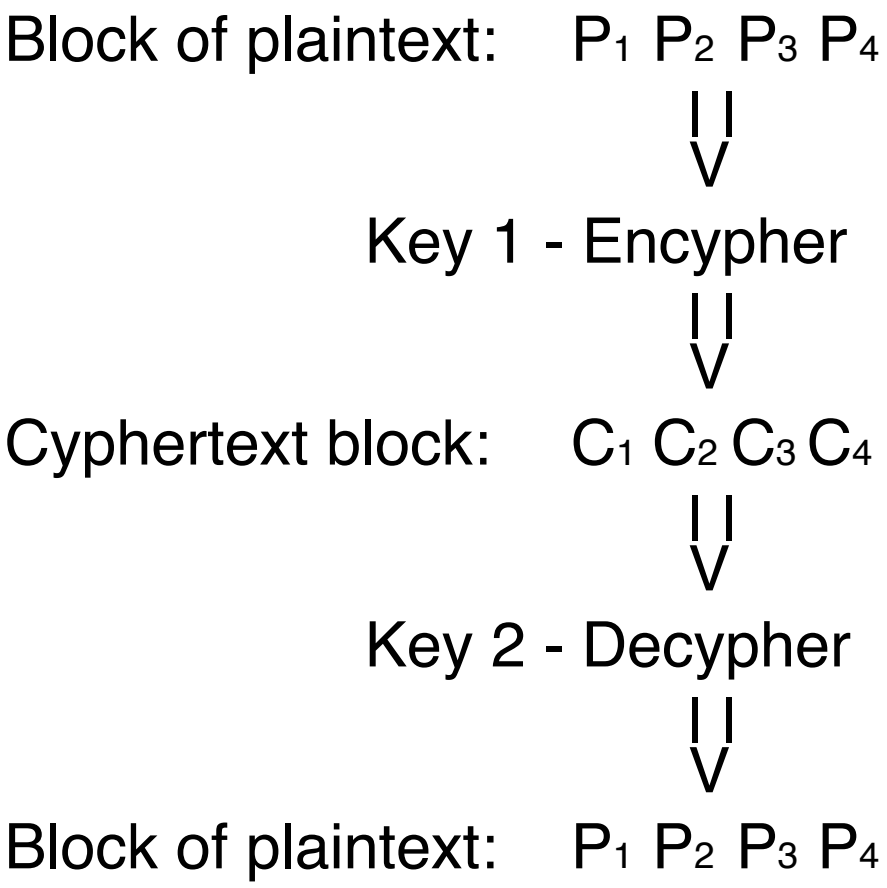
22C:169
Computer Security

Douglas W. Jones

Department of Computer Science

Public Key Cyphers

Public Key Cryptography:



Characteristics of Public Key Cyphers

For each key, encypher and decypher are
    One to One functions

Key_1 and Key_2 are independent
    One cannot be derived from the other

How do you find functions that satisfy this?
    Hard work!

Fundamental Insight

Trapdoor functions:

f_{TRAPDOOR} is P

f^{-1}_{TRAPDOOR} is NP complete

Example

$f(a,b) = a \times b$

$f^{-1}(x) = factor(x)$

WARNING: what if P = NP

RSA - The example public key cypher

Rivest, Shamir, Adleman, 1977

Basic outline:

p = 256-bit prime number

q = 258-bit prime number

(de - 1) divisible by (p - 1)(q - 1)

$Encrypt(P, <pq,e>) = P^{e} \text{ MOD } pq$

$Decrypt(C, <pq,d>) = C^{d} \text{ MOD } pq$

These are the same function!

RSA Continued:

Key generation:

Select {p,q,d,e} at random

Publish <*pq,e*> = *PK*

This is my public key

Hold as a secret <*pq,d*> = *SK*

This is my secret key

Discard {*p,q*}

To avoid possible security breach

Basic uses of any public key cypher

Users A and B publish PK_A, PK_B

A can send message only B can decode

A *sends* $C = PK_B(P)$

B *decodes* $P = SK_B(C)$

B can authenticate a message sent by A

A *sends* $C = SK_A(P)$

B *decodes* $P = PK_A(C)$

Another example use

Document Signatures

A *Publishes* < *D*, *H* >

D is a document, plaintext

$H = SK_A(\text{HASH}(D))$

Recipient can check that

$\text{HASH}(D) = PK_A(H)$

Obviously

HASH(*D*) *must be good*

Practical considerations

RSA is computationally expensive

Use it to exchange session keys

To communicate *M* from A to B

generate random session key k
send $SK_A(PK_B(\ k\ ))$
send $Encrypt_{AES}(\ M, k\ )$
discard session key k

Risk: We need a good randomness

How Secure is RSA?

Conjecture: depends on factoring speed

Known algorithms are exponential time

RSA Factoring Challenges

$20,000 to factor 640-bit number

$200,000 to factor 2048-bit number

140 bits Feb 2, 1999

155 bits Aug 22, 1999

576 bits Dec 3, 2003

The Public Key Infrastructure Problem

Suppose A wants to communicate with B

A *needs* PK_B

B *needs* PK_A

How do they know these are authentic?

Meet to personally exchange them?

could have exchanged private keys

Publish on the web?

how do we verify not spoofed?

Trusted third party?

who can we trust?