

More on Cryptography  
CS 136  
Computer Security  
Peter Reiher  
April 7, 2016

# Outline

- Desirable characteristics of ciphers
- Stream and block ciphers
- Cryptographic modes
- Uses of cryptography
- Symmetric and asymmetric cryptography
- Digital signatures

## Desirable Characteristics of Ciphers

- Well matched to requirements of application
  - Amount of secrecy required should match labor to achieve it
- Freedom from complexity
  - The more complex algorithms or key choices are, the worse

# More Characteristics

- Simplicity of implementation
  - Seemingly more important for hand ciphering
  - But relates to probability of errors in computer implementations
- Errors should not propagate

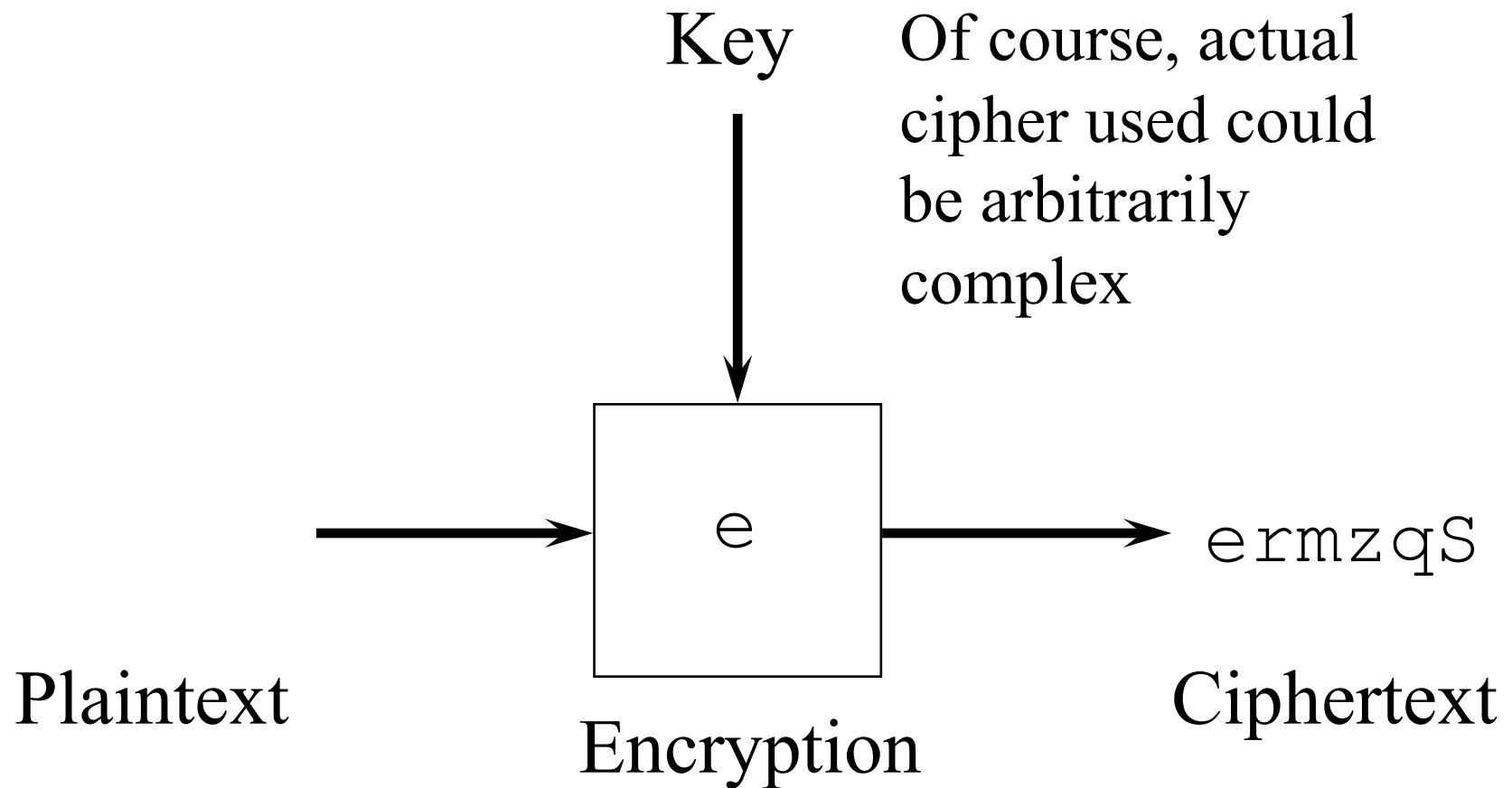
# Yet More Characteristics

- Ciphertext size should be same as plaintext size
- Encryption should maximize *confusion*
  - Relation between plaintext and ciphertext should be complex
- Encryption should maximize *diffusion*
  - Plaintext information should be distributed throughout ciphertext

# Stream and Block Ciphers

- Stream ciphers convert one symbol of plaintext immediately into one symbol of ciphertext
- Block ciphers work on a given sized chunk of data at a time

# Stream Ciphers



# Advantages of Stream Ciphers

- + Speed of encryption and decryption
  - Each symbol encrypted as soon as it's available
- + Low error propagation
  - Errors affect only the symbol where the error occurred
    - Depending on *cryptographic mode*



# Disadvantages of Stream Ciphers

- Low diffusion
  - Each symbol separately encrypted
  - Each ciphertext symbol only contains information about one plaintext symbol
- Susceptible to insertions and modifications
- Not good match for many common uses of cryptography
- Some disadvantages can be mitigated by use of proper cryptographic mode

# Sample Stream Cipher: RC4

- Creates a changing key stream
  - Supposedly unpredictable
- XOR the next byte of the key stream with the next byte of text to encrypt
- XOR ciphertext byte with same key stream byte to decrypt
- Alter your key stream as you go along

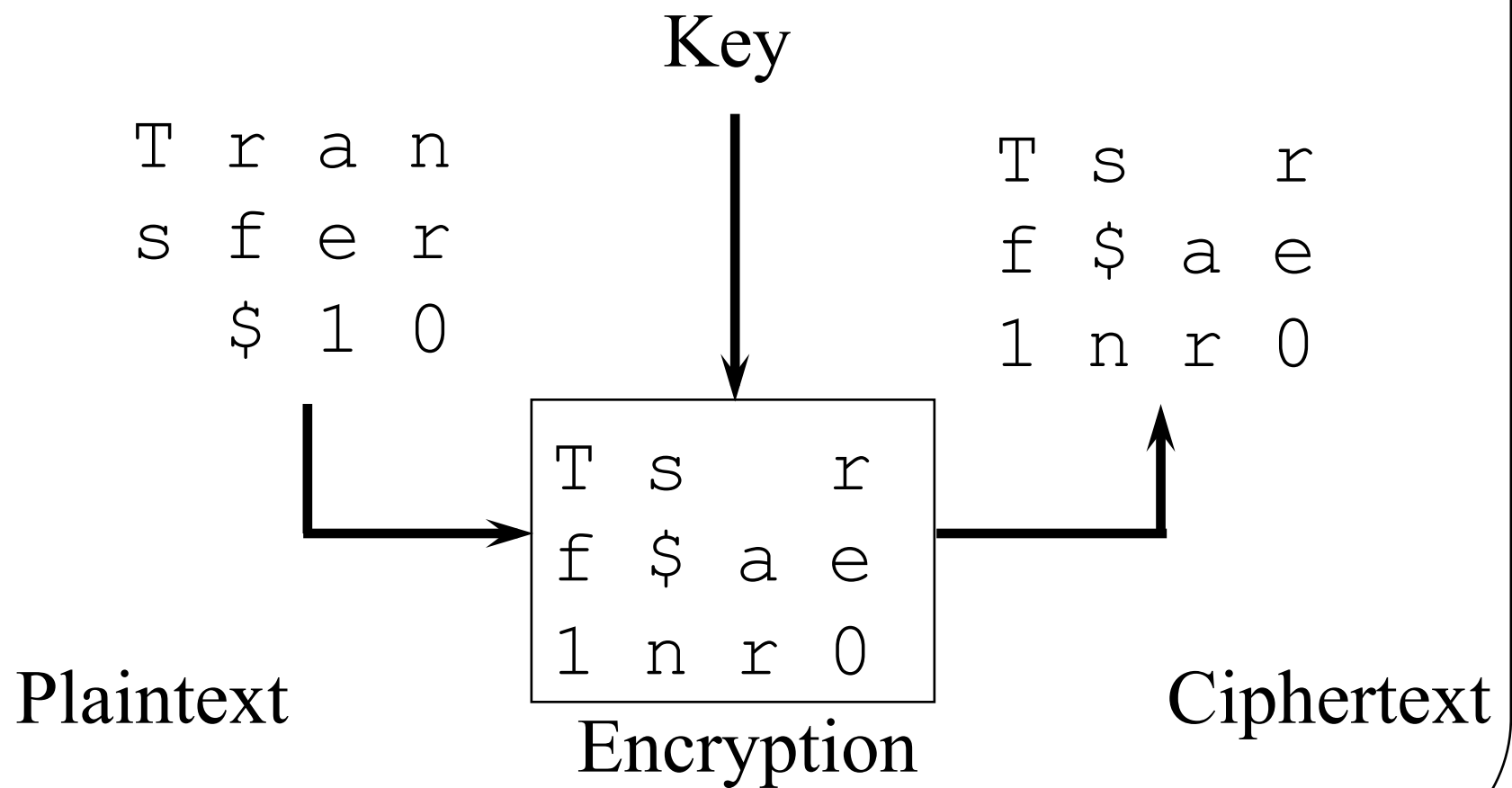
# Creating an RC4 Key

- Fill an 256 byte array with 0-255
- Choose a key of 1-255 bytes
- Fill a second array with the key
  - Size of array depends on the key
- Use a simple operation based on the key to swap around bytes in the first array
- That produces the key stream you'll use
- Swap two array bytes each time you encrypt

# Characteristics of RC4

- Around 10x faster than DES
- Significant cryptographic weakness in its initial key stream
  - Fixable by dropping the first few hundred of the keys
- Easy to use it wrong
  - Key reuse is a serious problem

# Block Ciphers



# Advantages of Block Ciphers

## + Good diffusion

- Easier to make a set of encrypted characters depend on each other

## + Immunity to insertions

- Encrypted text arrives in known lengths

Most common Internet crypto done with block ciphers

# Disadvantages of Block Ciphers

- Slower
  - Need to wait for block of data before encryption/decryption starts
- Worse error propagation
  - Errors affect entire blocks

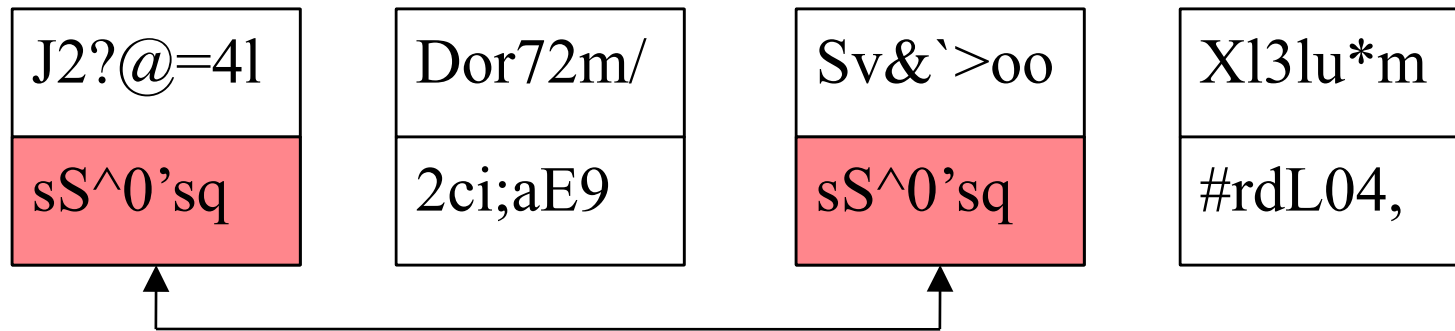
# Cryptographic Modes

- Let's say you have a bunch of data to encrypt
  - Using the same cipher and key
- How do you encrypt the entire set of data?
  - Given block ciphers have limited block size
  - And stream ciphers just keep going



# The Basic Situation

J2?@=4l	Dor72m/	Sv&`>oo	Xl3lu*m
sS^0'sq	2ci;aE9	sS^0'sq	#rdL04,



Let's say our block cipher has a block size of 7 characters and we use the same key for all

Now let's encrypt

There's something odd here . . .

*Is this good?*

*Why did it happen?*

# Another Problem With This Approach

What if these are transmissions representing deposits into bank accounts?



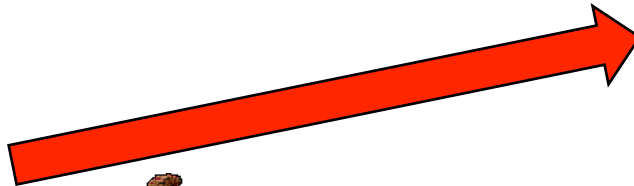
Xl3lu\*m

#rdL04,

Dor72m/

2ci;aE9

## Insertion Attack!



1840326	450
2201568	5000
3370259	8900
5610993	1579
6840924	2725
8436018	10

What if account 5610993  
belongs to him?

So far, so good . . .

## What Caused the Problems?

- Each block of data was independently encrypted
  - With the same key
- So two blocks with identical plaintext encrypt to the same ciphertext
- Not usually a good thing
- We used the wrong *cryptographic mode*
  - Electronic Codebook (ECB) Mode

# Cryptographic Modes

- A cryptographic mode is a way of applying a particular cipher
  - Block or stream
- The same cipher can be used in different modes
  - But other things are altered a bit
- A cryptographic mode is a combination of cipher, key, and feedback
  - Plus some simple operations

## So What Mode Should We Have Used?

- Cipher Block Chaining (CBC) mode might be better
- Ties together a group of related encrypted blocks
- Hides that two blocks are identical
- Foils insertion attacks

# Cipher Block Chaining Mode

- Adds feedback into encryption process
- The encrypted version of the previous block is used to encrypt this block
- For block  $X+1$ , XOR the plaintext with the ciphertext of block  $X$ 
  - Then encrypt the result
- Each block's encryption depends on all previous blocks' contents
- Decryption is similar

# What About the First Block?

- If we send the same first block in two messages with the same key,
  - Won't it be encrypted the same way?
- Might easily happen with message headers or standardized file formats
- CBC as described would encrypt the first block of the same message sent twice the same way both times

# Initialization Vectors

- A technique used with CBC
  - And other crypto modes
  - Abbreviated IV
- Ensures that encryption results are always unique
  - Even for duplicate message using the same key
- XOR a random string with the first block
  - $plaintext \oplus IV$
  - Then do CBC for subsequent blocks



# Encrypting With An IV

First block of message

1	1	0	1	0	0	0	1
---	---	---	---	---	---	---	---

Initialization vector

0	1	0	0	1	1	0	0
---	---	---	---	---	---	---	---

XOR IV and message

0	0	1	1	0	1	1	1
---	---	---	---	---	---	---	---

Encrypt msg and send  
IV plus message

Second block of message

0	0	0	1	1	0	0	0
---	---	---	---	---	---	---	---

Use previous msg for CBC

Apply CBC

1	0	0	1	1	1	1	0
---	---	---	---	---	---	---	---

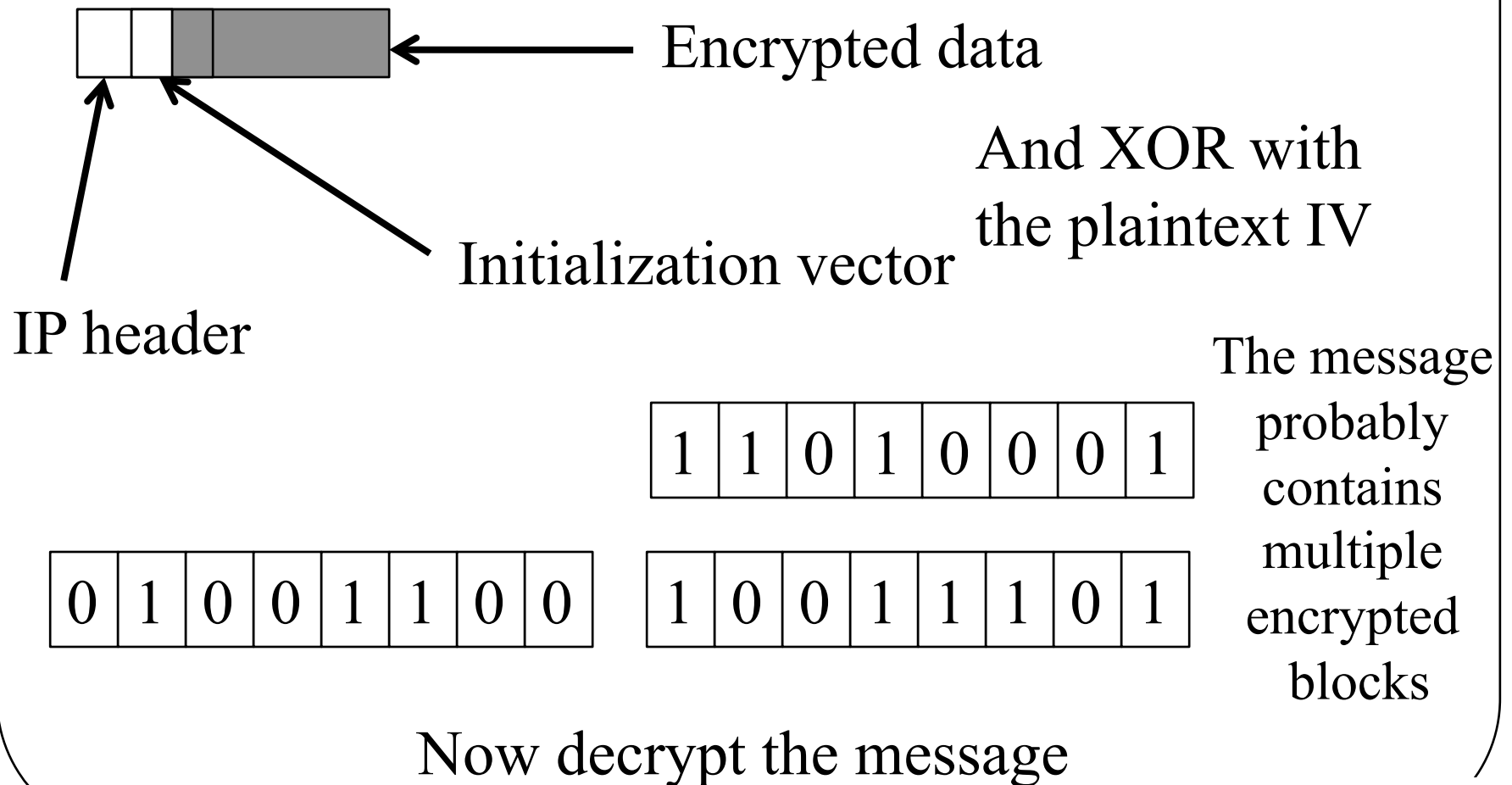
Encrypt and send second  
block of msg

No need to also send 1<sup>st</sup> block again

# How To Decrypt With Initialization Vectors?

- First block received decrypts to
$$P = \textit{plaintext} \oplus IV$$
- $\textit{plaintext} = P \oplus IV$
- No problem if receiver knows  $IV$ 
  - Typically,  $IV$  is sent in the message
- Subsequent blocks use standard CBC
  - So can be decrypted that way

# An Example of IV Decryption



# For Subsequent Blocks



Use previous ciphertext  
block instead of IV

And XOR with the  
previous ciphertext block

0	0	0	1	1	0	0	0
---	---	---	---	---	---	---	---

0	0	1	1	0	1	1	1
---	---	---	---	---	---	---	---

1	1	0	0	0	0	1	1
---	---	---	---	---	---	---	---

Now decrypt the message

# Some Important Crypto Modes

- Electronic codebook mode (ECB)
- Cipher block chaining mode (CBC)
- Cipher-feedback mode (CFB) and Output-feedback mode (OFB)

Both convert block to stream cipher

# Uses of Cryptography

- What can we use cryptography for?
- Lots of things
  - Secrecy
  - Authentication
  - Prevention of alteration

# Cryptography and Secrecy

- Pretty obvious
- Only those knowing the proper keys can decrypt the message
  - Thus preserving secrecy
- Used cleverly, it can provide other forms of secrecy

# Cryptography and Authentication

- How can I prove to you that I created a piece of data?
- What if I give you the data in encrypted form?
  - Using a key only you and I know
- Then only you or I could have created it
  - Unless one of us told someone else the key . . .



# Using Cryptography for Authentication

- If both parties cooperative, standard cryptography can authenticate
  - Problems with non-repudiation, though
- What if three parties want to share a key?
  - No longer certain who created anything
  - Public key cryptography can solve this problem
- What if I want to prove authenticity without secrecy?

# Cryptography and Non-Alterability

- Changing one bit of an encrypted message completely garbles it
  - For many forms of cryptography
- If a checksum is part of encrypted data, that's detectable
- If you don't need secrecy, can get the same effect
  - By encrypting only the checksum

# Symmetric and Asymmetric Cryptosystems

- Symmetric - the encrypter and decrypter share a secret key
  - Used for both encrypting and decrypting
- Asymmetric – encrypter has different key than decrypter

# Description of Symmetric Systems

- $C = E(K, P)$
- $P = D(K, C)$
- $E()$  and  $D()$  are not necessarily the same operations

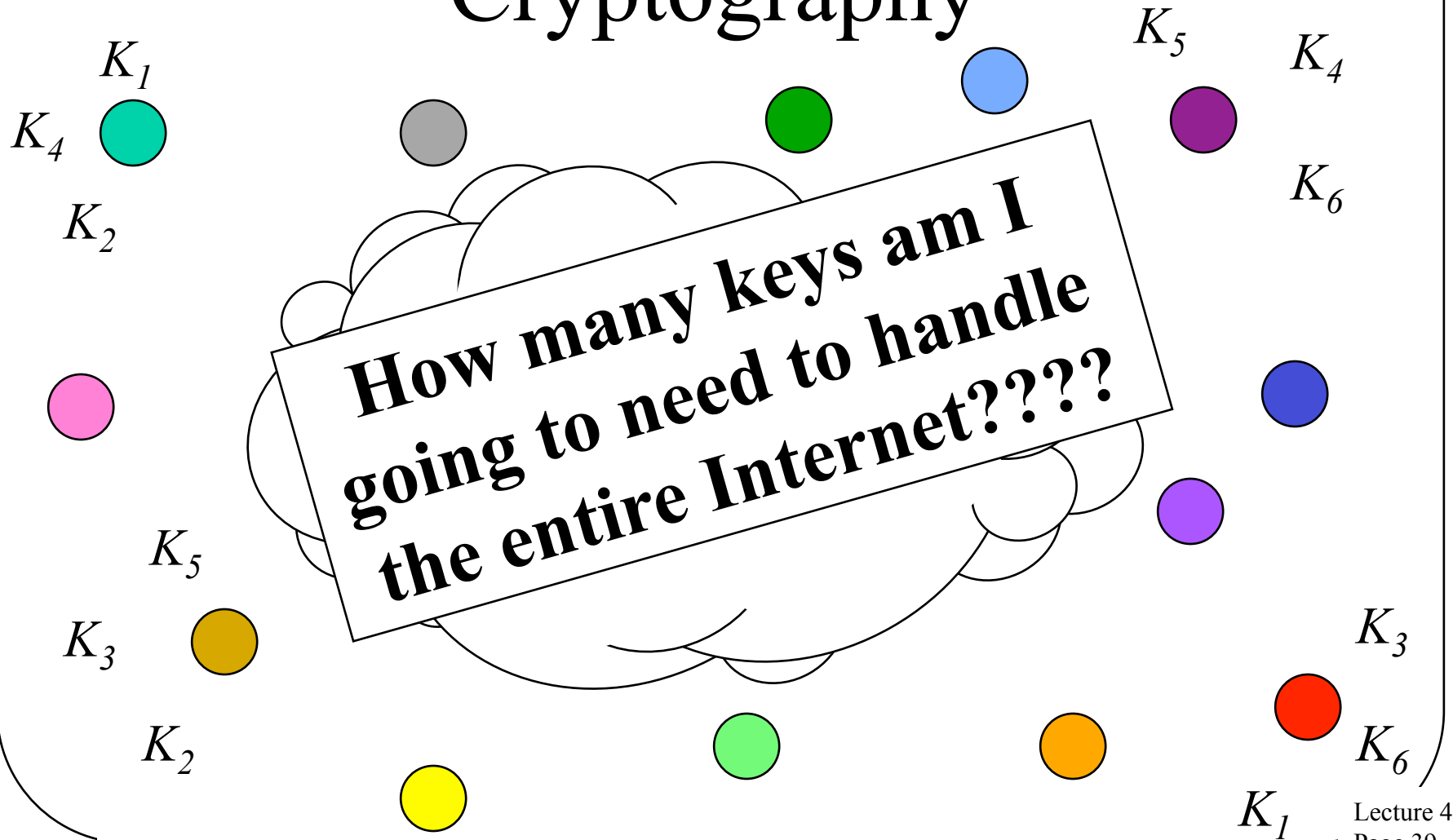
# Advantages of Symmetric Key Systems

- + Encryption and authentication performed in a single operation
- + Well-known (and trusted) ones perform faster than asymmetric key systems
- + Doesn't require any centralized authority
  - Though key servers help a lot

# Disadvantage of Symmetric Key Systems

- Encryption and authentication performed in a single operation
  - Makes signature more difficult
- Non-repudiation hard without servers
- Key distribution can be a problem
- Scaling

# Scaling Problems of Symmetric Cryptography



# Sample Symmetric Key Ciphers

- The Data Encryption Standard
- The Advanced Encryption Standard
- There are many others



# The Data Encryption Standard

- Well known symmetric cipher
- Developed in 1977, still much used
  - Shouldn't be, for anything serious
- Block encryption, using substitutions, permutations, table lookups
  - With multiple *rounds*
  - Each round is repeated application of operations
- Only serious problem based on short key

# The Advanced Encryption Standard

- A relatively new cryptographic algorithm
- Intended to be the replacement for DES
- Chosen by NIST
  - Through an open competition
- Chosen cipher was originally called Rijndael
  - Developed by Dutch researchers
  - Uses combination of permutation and substitution

# Increased Popularity of AES

- Gradually replacing DES
  - As was intended
- Various RFCs describe using AES in IPsec
- FreeS/WAN IPsec (for Linux) includes AES
- Most commercial VPNs use AES
- Used in modern Windows and Mac OS systems

# Is AES Secure?

- No complete breaks discovered so far
- But some disturbing problems
  - Attacks that work on versions of AES using fewer rounds
  - Attacks that get keys quicker than brute force
    - But not practical time (e.g. in  $2^{126}$  operations)
- But unusable crypto flaws often lead to usable ones
- Attacks on crypto only get better over time, never worse

# Public Key Encryption Systems

- The encrypter and decrypter have different keys

$$C = E(K_E, P)$$

$$P = D(K_D, C)$$

- Often, works the other way, too

$$C' = E(K_D, P)$$

$$P = D(K_E, C')$$

# History of Public Key Cryptography

- Invented by Diffie and Hellman in 1976
- Merkle and Hellman developed Knapsack algorithm in 1978
- Rivest-Shamir-Adelman developed RSA in 1978
  - Most popular public key algorithm
- Many public key cryptography advances secretly developed by British and US government cryptographers earlier

# Practical Use of Public Key Cryptography

- Keys are created in pairs
- One key is kept secret by the owner
- The other is made public to the world
- If you want to send an encrypted message to someone, encrypt with his public key
  - Only he has private key to decrypt

# Authentication With Shared Keys

- If only two people know the key, and I didn't create a properly encrypted message -
  - The other guy must have
- But what if he claims he didn't?
- Or what if there are more than two?
- Requires authentication servers

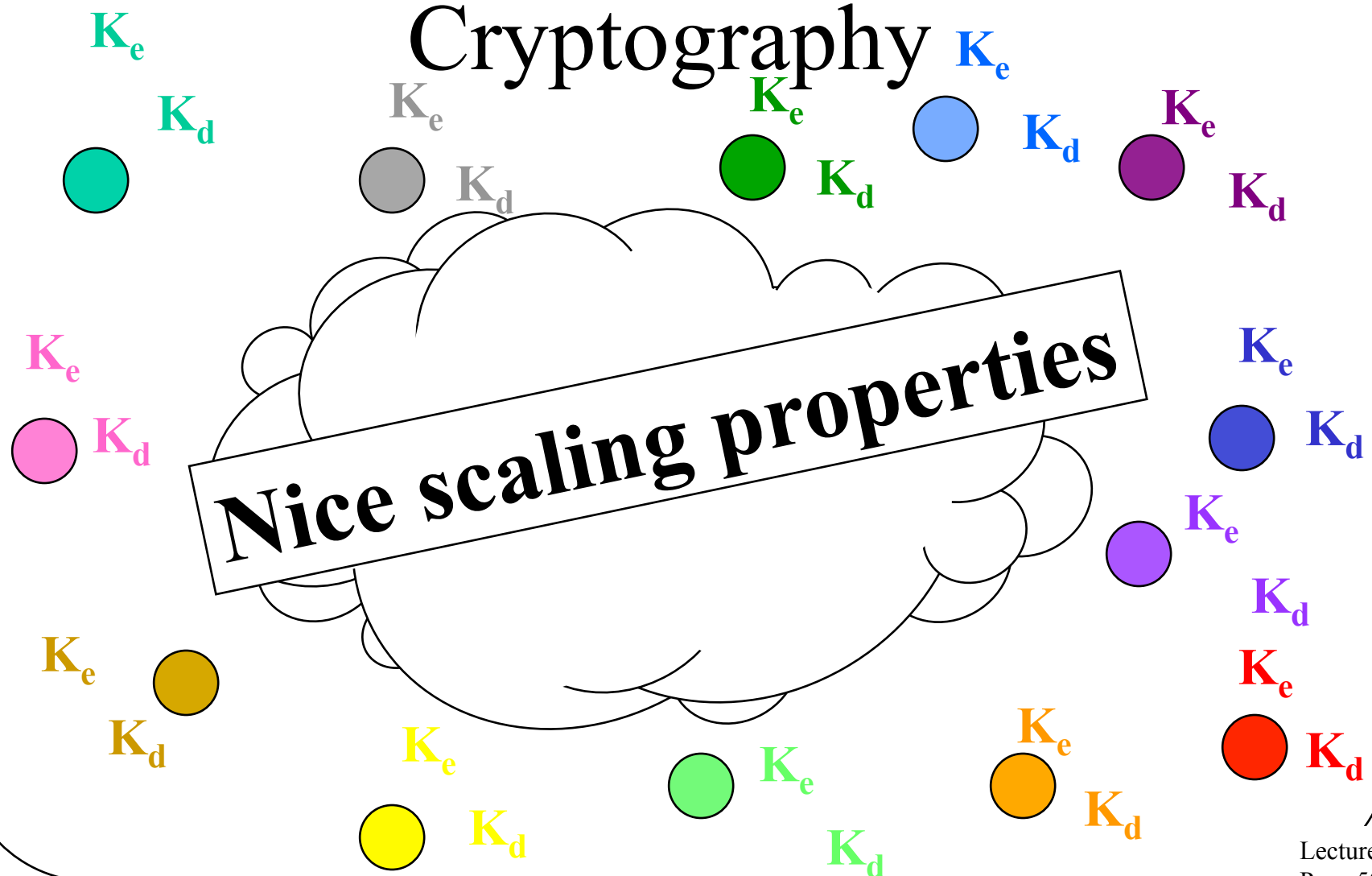


# Authentication With Public Keys

- If I want to “sign” a message, encrypt it with my private key
- Only I know private key, so no one else could create that message
- Everyone knows my public key, so everyone can check my claim directly

# Scaling of Public Key

## Cryptography



# Key Management Issues

- To communicate via shared key cryptography, key must be distributed
  - In trusted fashion
- To communicate via public key cryptography, need to find out each other's public key
  - “Simply publish public keys”

# Issues of Key Publication

- Security of public key cryptography depends on using the right public key
- If I am fooled into using the wrong one, that key's owner reads my message
- Need high assurance that a given key belongs to a particular person
- Which requires a *key distribution infrastructure*

# RSA Algorithm

- Most popular public key cryptographic algorithm
- In wide use
- Has withstood much cryptanalysis
- Based on hard problem of factoring large numbers

# RSA Keys

- Keys are functions of a pair of 100-200 digit prime numbers
- Relationship between public and private key is complex
- Recovering plaintext without private key (even knowing public key) is supposedly equivalent to factoring product of the prime numbers

# Comparison of AES and RSA

- AES is much more complex
- However, AES uses only simple arithmetic, logic, and table lookup
- RSA uses exponentiation to large powers
  - Computationally 1000 times more expensive in hardware, 100 times in software
- RSA key selection also much more expensive

# Is RSA Secure?

- Conjectured that security depends on factoring large numbers
  - But never proven
  - Some variants proven equivalent to factoring problem
- Probably the conjecture is correct
- Key size for RSA doesn't have same meaning as DES and AES



# Attacks on Factoring RSA Keys

- In 2005, a 663 bit RSA key was successfully factored
- A 768 bit key factored in 2009
- Research on integer factorization suggests keys up to 2048 bits may be insecure
- Insecure key length will only increase
- The longer the key, the more expensive the encryption and decryption

# Elliptical Cryptography

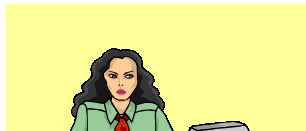
- RSA and similar algorithms related to factoring products of large primes
- Other math can be used for PK, instead
  - Properties of elliptical curves, e.g.
- Can give same security as other public key schemes, with much smaller keys
- Widely studied, regarded as safe
  - But the NSA is pushing it . . .
  - Often used for small devices

# Combined Use of Symmetric and Asymmetric Cryptography

- Common to use both in a single session
- Asymmetric cryptography essentially used to “bootstrap” symmetric crypto
- Use RSA (or another PK algorithm) to authenticate and establish a *session key*
- Use AES with that session key for the rest of the transmission

# Combining Symmetric and Asymmetric Crypto

Alice wants to share  
the key only with Bob



But there are problems we'll discuss later



Alice

$K_{EA}$   $K_{DA}$   
 $K_{EB}$

$K_S$

$$C = E(K_S, K_{EB})$$

$$M = E(C, K_{DA})$$

Only Bob

can decrypt it

Only Alice could  
have created it



Bob

$K_{EB}$   $K_{DB}$   
 $K_{EA}$

$$K_S = D(C, K_{DB})$$

$$M = D(K_S, K_{EA})$$

# Digital Signature Algorithms

- In some cases, secrecy isn't required
- But authentication is
- The data must be guaranteed to be that which was originally sent
- Especially important for data that is long-lived

# Desirable Properties of Digital Signatures

- Unforgeable
- Verifiable
- Non-repudiable
- Cheap to compute and verify
- Non-reusable
- No reliance on trusted authority
- Signed document is unchangeable

# Encryption and Digital Signatures

- Digital signature methods are based on encryption
- The basic act of having performed encryption can be used as a signature
  - If only I know  $K$ , then  $C=E(P,K)$  is a signature by me
  - But how to check it?

# Signatures With Shared Key Encryption

- Requires a trusted third party
- Signer encrypts document with secret key shared with third party
- Receiver checks validity of signature by consulting with trusted third party
- Third party required so receiver can't forge the signature



# For Example,

$K_s$



When in  
the Course  
of human  
events it  
becomes  
necessary  
for one

Elas7pa  
1o'gw0mega  
30'sswp.  
1f43'-s 4  
32.doas3  
Dsp5.a#1  
^o,a 02



$K_s$

When in  
the Course  
of human  
events it  
becomes  
necessary  
for one

# Signatures With Public Key Cryptography

- Signer encrypts document with his private key
- Receiver checks validity by decrypting with signer's public key
- Only signer has the private key
  - So no trusted third party required
- But receiver must be certain that he has the right public key

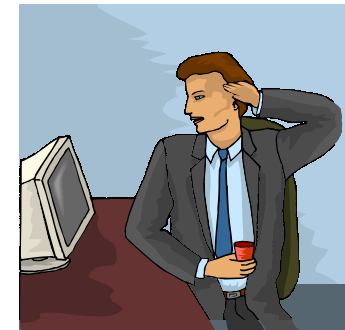
# For Example,

$K_d$



When in  
the Course  
of human  
events it  
becomes  
necessary  
for one

Elas7pa  
1o'gw0mega  
30'sswp.  
1f43'-s 4  
32.doas3  
Dsp5.a#1  
^o,a 02



$K_e$

Alice's  
public  
key

When in  
the Course  
of human  
events it  
becomes  
necessary  
for one

# Problems With Simple Encryption Approach

- Computationally expensive
  - Especially with public key approach
- Document is encrypted
  - Must be decrypted for use
  - If in regular use, must store encrypted and decrypted versions