CIT 424
Computer System Security

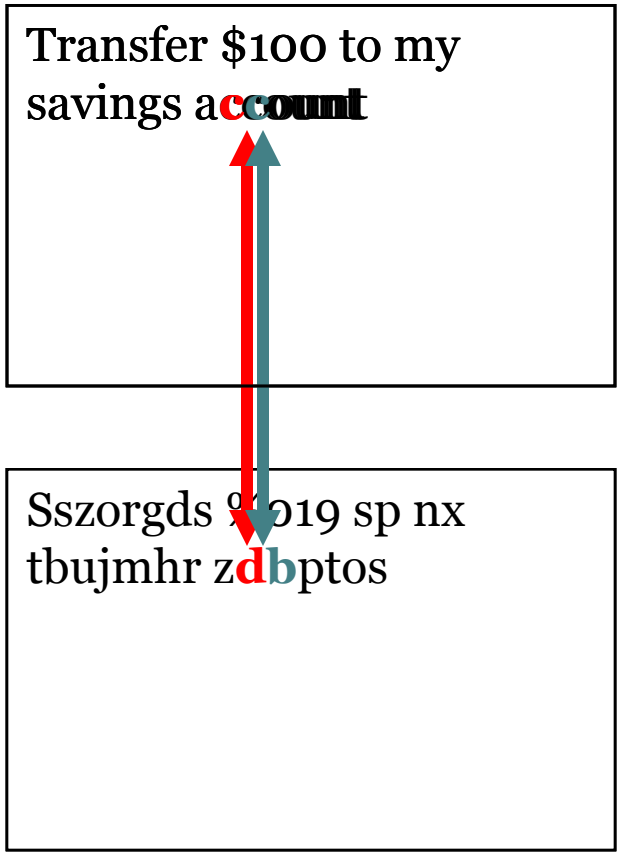
01/26/15
Spring 2015

Polyalphabetic Ciphers

- Ciphers that don't always translate a given plaintext character into the same ciphertext character
- For example, use different substitutions for odd and even positions

Example of Simple Polyalphabetic Cipher

- Move one character "up" in even positions, one character "down" in odd positions
- Note that same character translates to different characters in some cases



Are Polyalphabetic Ciphers Better?

- Depends on how easy it is to determine the pattern of substitutions
- If it's easy, then you've gained little

Cryptanalysis of Our Example

- Consider all even characters as one set
- And all odd characters as another set
- Apply basic cryptanalysis to each set
- The transformations fall out easily
- How did you know to do that?
  - You guessed
  - Might require several guesses to find the right pattern

How About For More Complex Patterns?

- Good if the attacker doesn't know the choices of which characters get transformed which way
- Attempt to hide patterns well
- But known methods still exist for breaking them

Methods of Attacking Polyalphabetic Ciphers

- Kasiski method tries to find repetitions of the encryption pattern
- Index of coincidence predicts the number of alphabets used to perform the encryption
- Both require lots of ciphertext

How Does the Cryptanalyst "Know" When He's Succeeded?

- Every key translates a message into something
- If a cryptanalyst thinks he's got the right key, how can he be sure?
- Usually because he doesn't get garbage when he tries it
- He almost certainly will get garbage from any other key
- Why?

Consider A Caesar Cipher

- There are 25 useful keys (in English)
- The right one will clearly yield meaningful text
- What's the chances that any of the other 24 will?
  - Pretty poor
- So if the decrypted text makes sense, you've got the key

The Unbreakable Cipher

- There is a “perfect” substitution cipher
- One that is theoretically (and practically) unbreakable without the key
- And you can’t guess the key
  - If the key was chosen in the right way . . .

One-Time Pads

- Essentially, use a new substitution alphabet for every character
- Substitution alphabets chosen purely at random
  - These constitute the key
- Provably unbreakable without knowing this key

Example of One Time Pads

- Usually explained with bits, not characters
- We shall use a highly complex cryptographic transformation:
  - XOR
- And a three bit message
  - 010

One Time Pads at Work

0	1	0

Flip some coins to get random numbers

0	0	1

0	1	1

Apply our sophisticated cryptographic algorithm

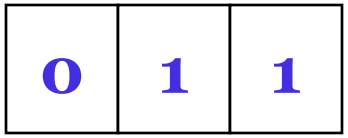
We now have an unbreakable cryptographic message

What's So Secure About That?

- Any key was equally likely
- Any plaintext could have produced this message with one of those keys
- Let's look at our example more closely

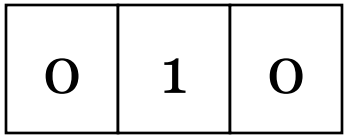
Why Is the Message Secure?

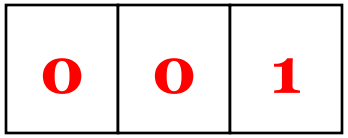
Let's say there are only two possible meaningful messages



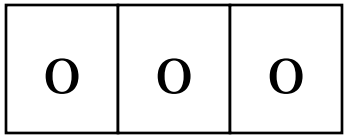
There's a key that works for each

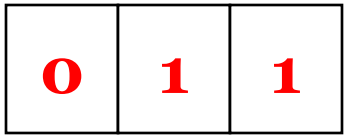
And they're equally likely





Could the message decrypt to either or both of these?





Security of One-Time Pads

- If the key is truly random, provable that it can't be broken without the key
- But there are problems
- Need one bit of key per bit of message
- Key distribution is painful
- Synchronization of keys is vital
- A good random number generator is hard to find

One-Time Pads and Cryptographic Snake Oil

- Companies regularly claim they have "unbreakable" cryptography
- Usually based on one-time pads
- But typically misused
  - Pads distributed with some other crypto mechanism
  - Pads generated with non-random process
  - Pads reused

Permutation Ciphers

- Instead of substituting different characters, scramble up the existing characters
- Use algorithm based on the key to control how they're scrambled
- Decryption uses key to unscramble

Characteristics of Permutation Ciphers

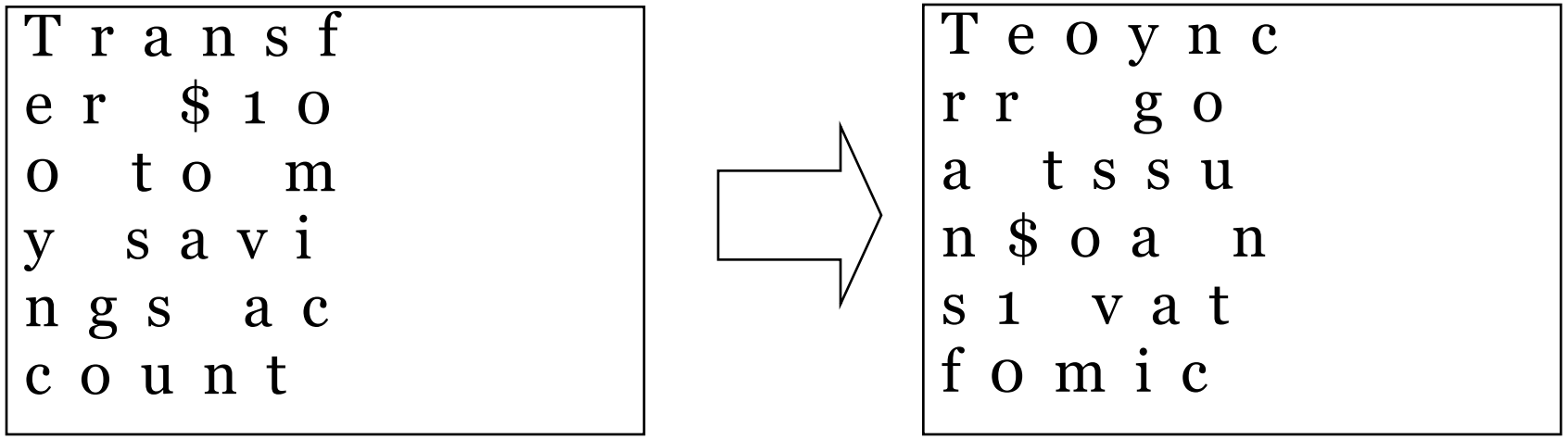
- Doesn't change the characters in the message
  - Just where they occur
- Thus, character frequency analysis doesn't help cryptanalyst

Columnar Transpositions

- Write the message characters in a series of columns
- Copy from top to bottom of first column, then second, etc.

Example of Columnar Substitution

How did this transformation happen?



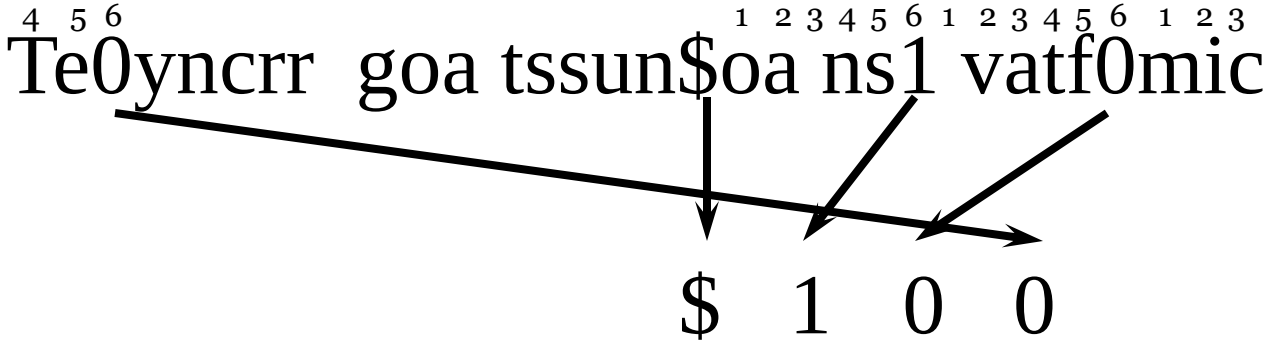
Looks a lot more cryptic written this way:

Te0yncrr goa tssun$oa ns1 vatf0mic

Attacking Columnar Transformations

- The trick is figuring out how many columns were used
- Use information about digrams, trigrams, and other patterns
- Digrams are pairs of letters that frequently occur together (“re”, “th”, “en”, e.g.)
- For each possibility, check digram frequency

For Example,



In our case, the presence of dollar signs and numerals in the text is suspicious

Maybe they belong together?

Umm, maybe there's 6 columns?

Double Transpositions

- Do it twice
- Using different numbers of columns
- How do you break it?
  - Find pairs of letters that probably appeared together in the plaintext
  - Figure out what transformations would put them in their positions in the ciphertext
- Can transform more than twice, if you want

Generalized Transpositions

- Any algorithm can be used to scramble the text
- Usually somehow controlled by a key
- Generality of possible transpositions makes cryptanalysis harder

Which Is Better, Transposition or Substitution?

- Well, neither, really
- Strong modern ciphers tend to use both
- Transposition scrambles text patterns
- Substitution hides underlying text characters/bits
- Combining them can achieve both effects
  - If you do it right . . .

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

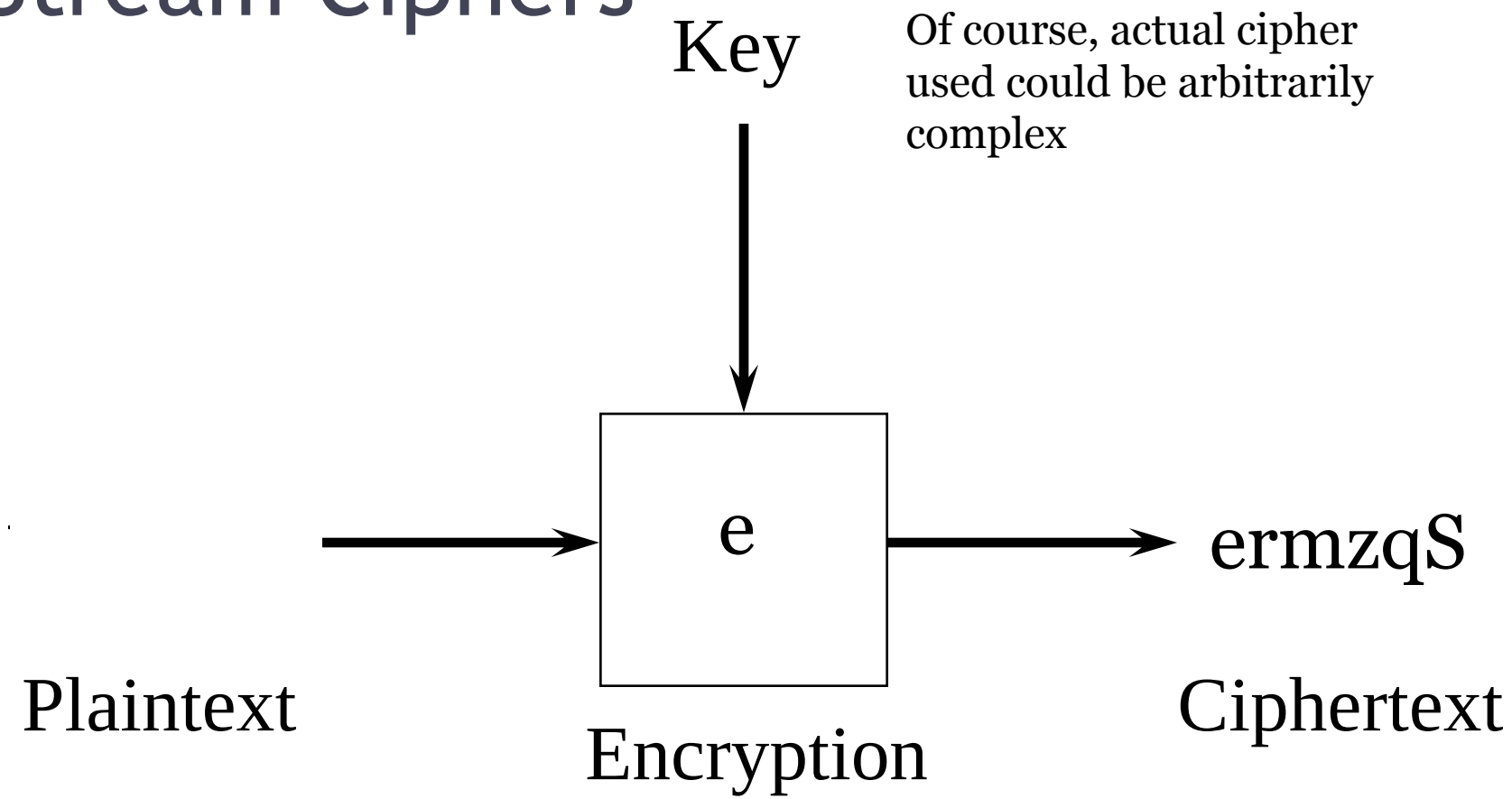
Yet More Characteristics

- 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



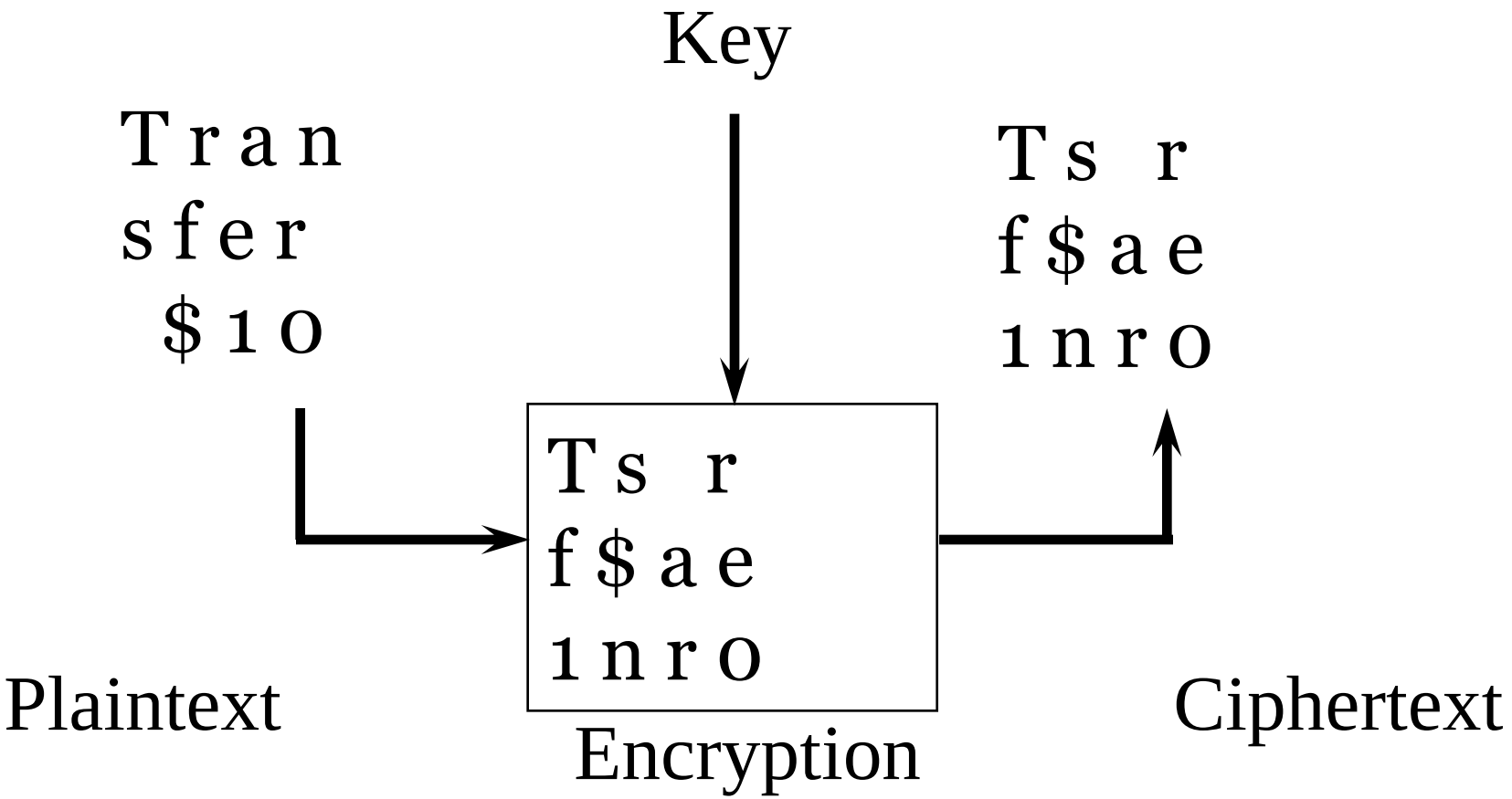
Advantages of Stream Ciphers

- 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

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

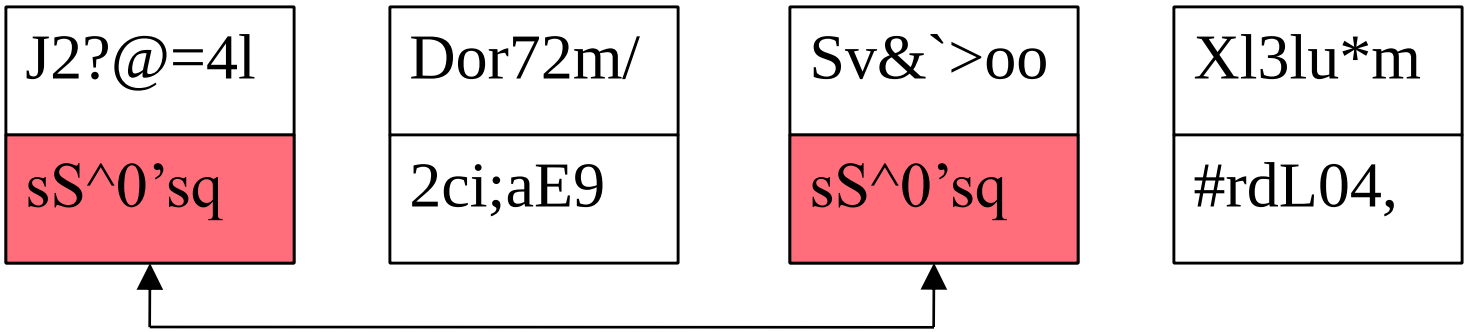
Disadvantages of Block Ciphers

- 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



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,

Insertion Attack!

Dor72m/
2ci;aE9

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 1st block again

How To Decrypt With Initialization Vectors?

- First block received decrypts to
  $P = plaintext \oplus IV$
- $plaintext = P \oplus IV$
- 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

Encrypted data

And XOR with the plaintext IV

Initialization vector

IP header

1	1	0	1	0	0	0	1

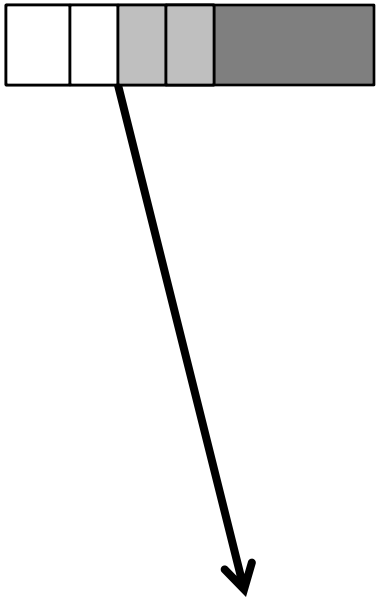
0	1	0	0	1	1	0	0

1	0	0	1	1	1	0	1

The message probably contains multiple encrypted blocks

Now decrypt the message

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)
- Counter mode (CTR)
- 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

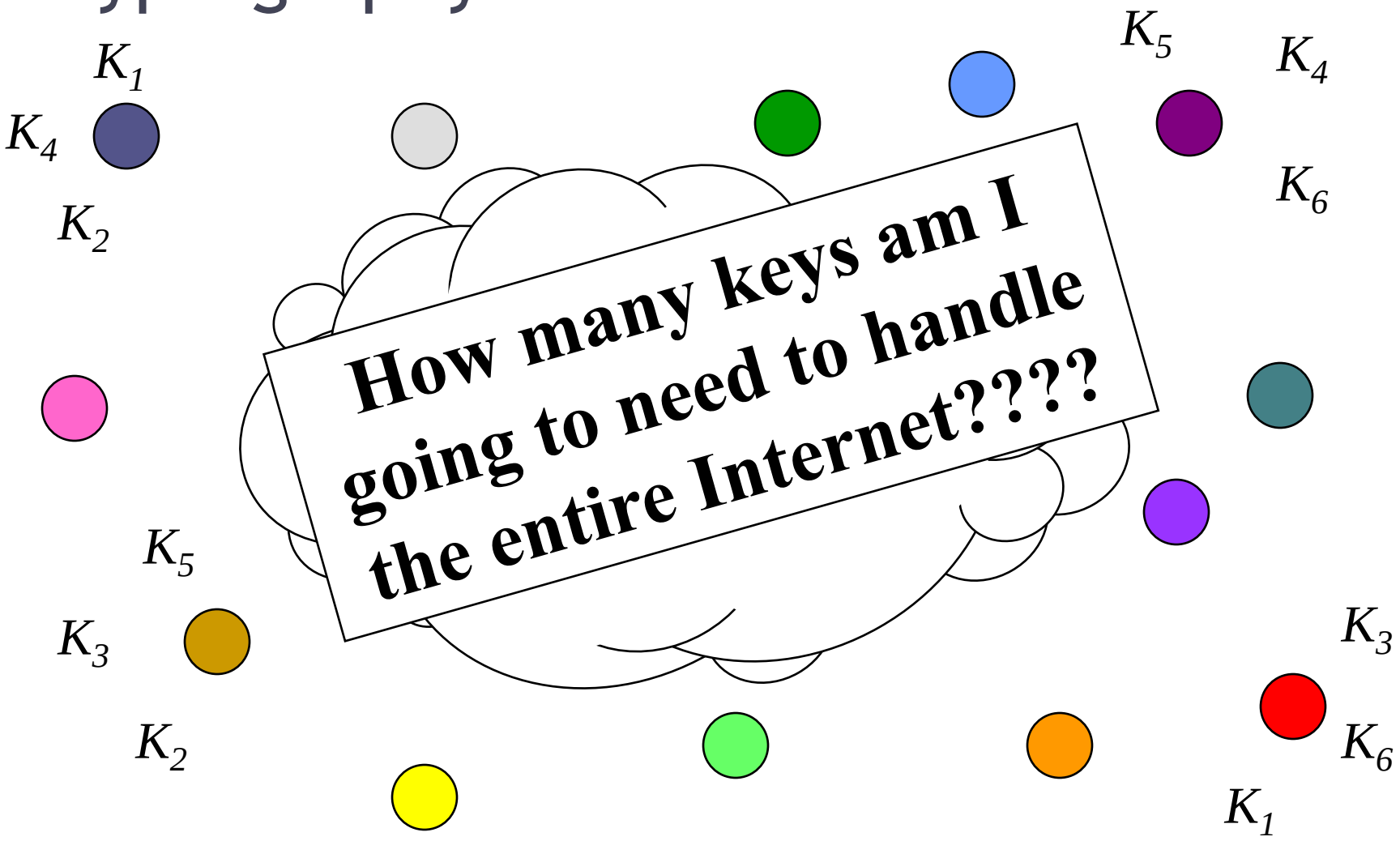
Advantages of Symmetric Key Systems

+ 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
- Some commercial VPNs use AES
- Used in modern Windows systems
  - Also recent versions of Mac OS

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

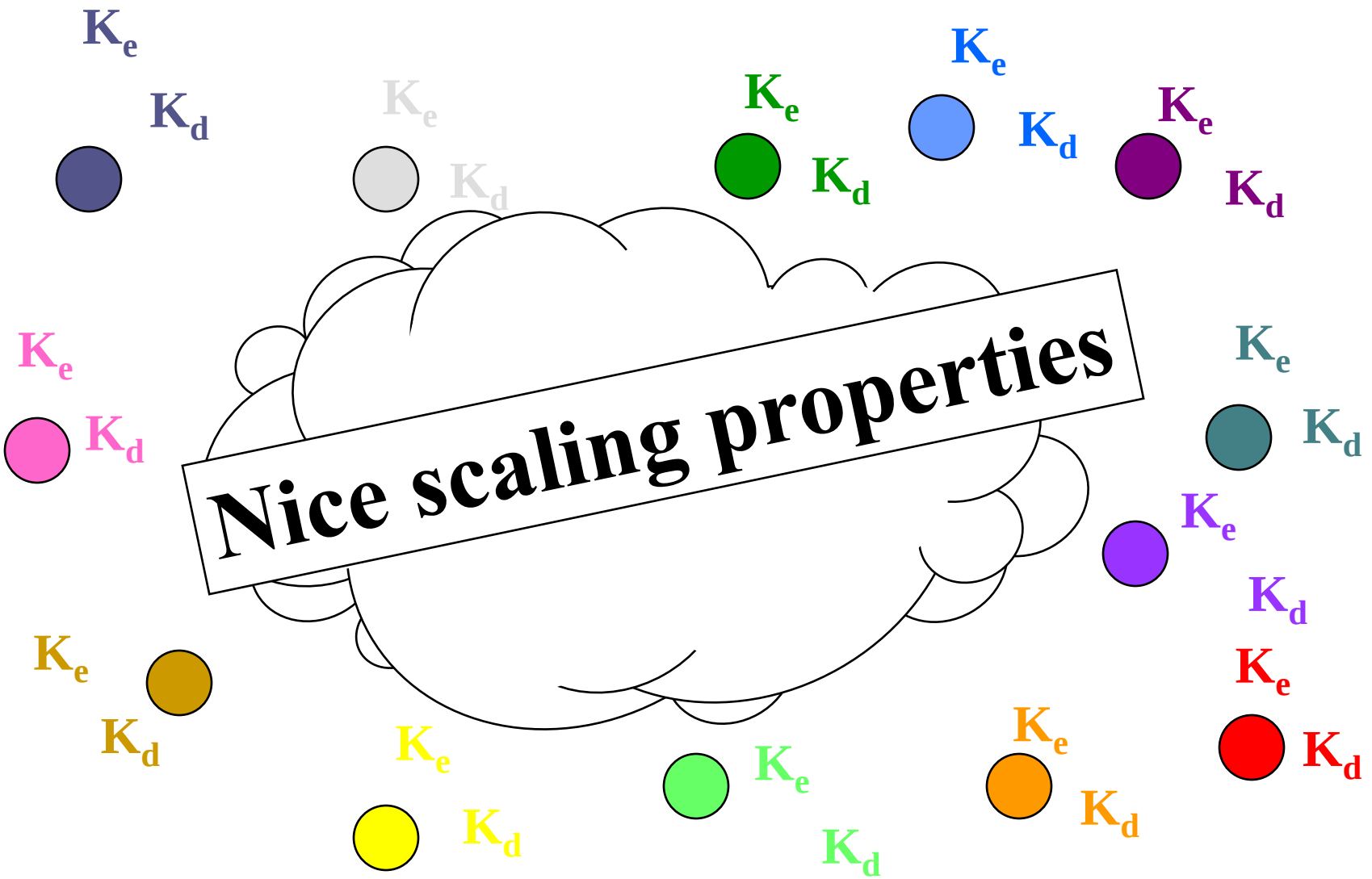
Authentication With Shared Keys

- 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
- The longer the key, the more expensive the encryption and decryption

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

it's Alice's key

Only Bob can decrypt it

Only Alice could have created it

Alice

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

K_S

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

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

Bob

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

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

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