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

simple algorithms are good, complex algorithms are bad…
ex. one time pad is simple to implement
but complex to crack

More Characteristics

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

ex. 1 MB of data to encrypt; 1 bit changed, it would be unfortunate if that entire information was not decipherable anymore

Yet More Characteristics

- Ciphertext size should be same as plaintext size

do you want reasonable encruption
or do you want to maximize diffusion

- Encryption should maximize *confusion*
  – Relation between plaintext and ciphertext should be complex
- Encryption should maximize *diffusion*
  – Plaintext information should be distributed throughout ciphertext

you want to relationship between the plaintext and ciphertext to be very unclear
a monoalphabetic substritution is not good b/c did not get rid of letter frequency

Stream and Block Ciphers

- Stream ciphers convert one symbol of plaintext immediately into one symbol of ciphertext

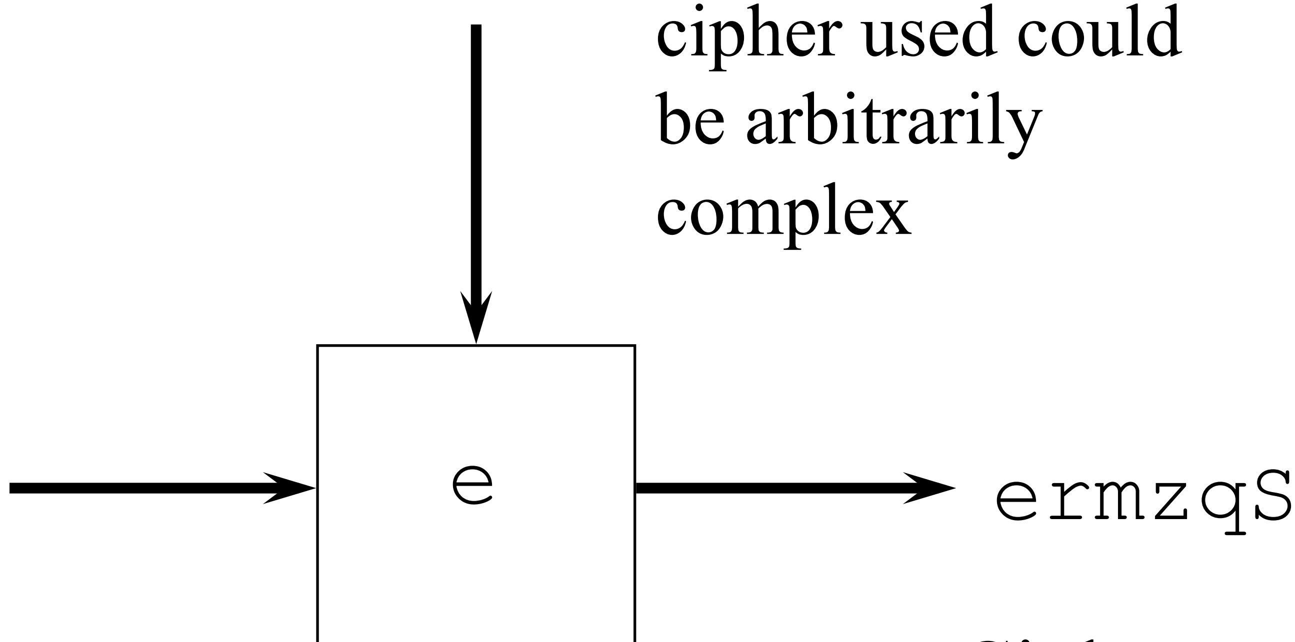
stream cipher means going through one symbol at a time

- Block ciphers work on a given sized chunk of data at a time

Stream Ciphers

Key

Of course, actual cipher used could be arbitrarily complex



Plaintext

Encryption

Ciphertext

Advantages of Stream Ciphers

+ Speed of encryption and decryption
  - Each symbol encrypted as soon as it's available

very accessible, you can encrypt and decryt immdiately

+ 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

  attackers can insert some enrypted symboling
- Not good match for many common uses of cryptography

  encrypt paylod of IP packet - yeah you can modify it
- Some disadvantages can be mitigated by use of proper cryptographic mode

this key stream may be unpredictable
just need a simple algorithm

XOR - do it twice, get what you started with

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

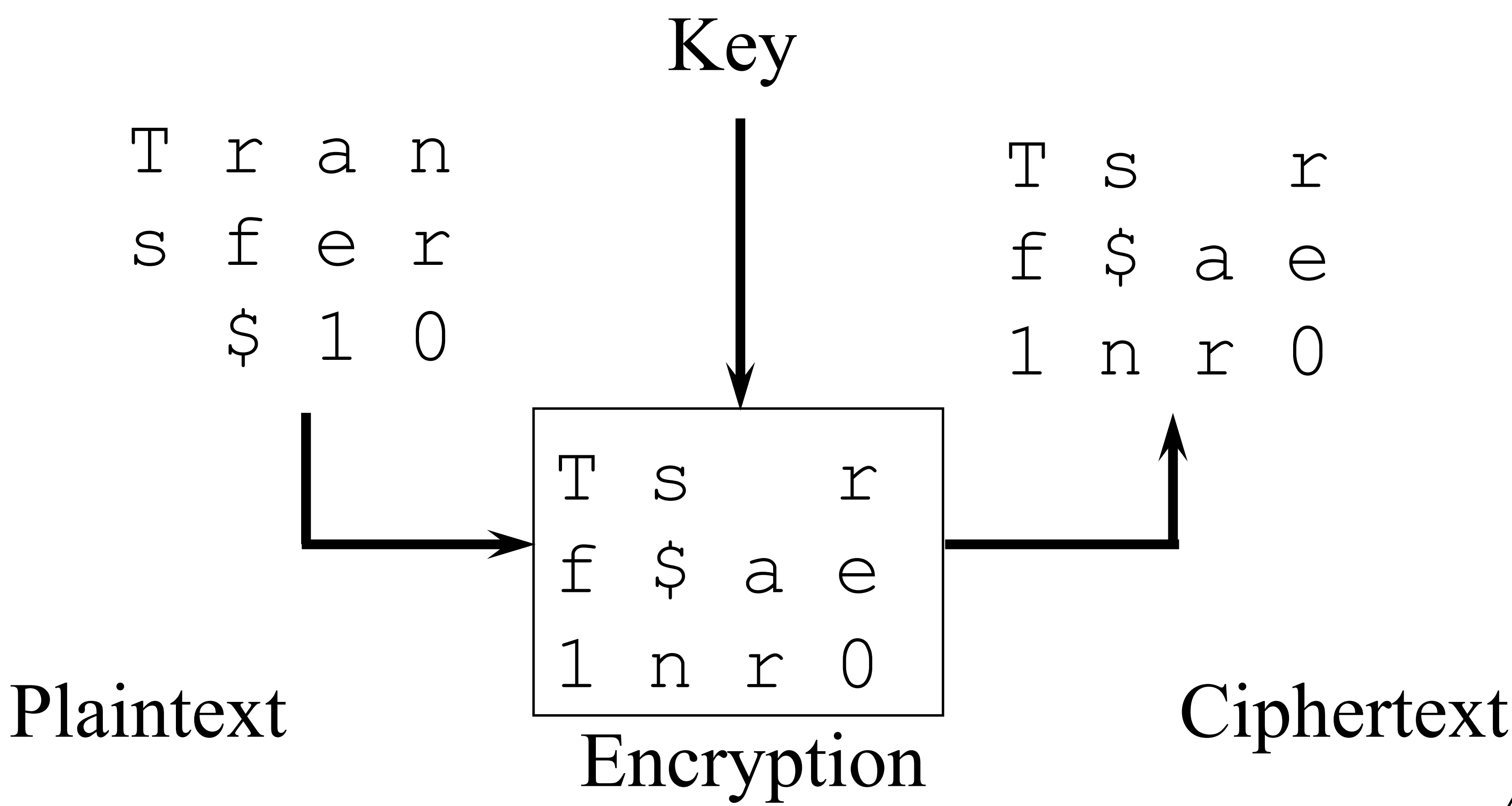
Characteristics of RC4

- Around 10x faster than DES
- Significant cryptographic weakness in its initial key stream

  implementation of RC4 incorrect; don' reuse key
  last year or 2: deprecated for RC4
  don't use RC3 any more

  – 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 depends on what kind of block ciphers this s

  • 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 cryptographic modes - encrypting an entire bank
  - 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

This is not good. It happened due to inconsistancy

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

mapped to the same in encrypted form

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?*

issues like this happen a lthe tiem

Another Problem With This Approach

What if these are transmissions representing deposits into bank accounts?

X13lu*m
#rdL04,

Insertion Attack!

Dor72m/
2ci;aE9

encryption happened
he doesn't need it all

insertion attack
added in things that don't exist

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

What if account 5610993 belongs to him?

So far, so good . . .

really hard to use cryptography right; but OpenSSL used very very widley

ECB (electronic codebook) is the default one to use (not that good lol)

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

more than one block of data or a stream of data
but protect us from insert attack

So What Mode Should We Have Used?

- Cipher Block Chaining (CBC) mode might be better

takes a bunch of encrypted blocks, and it ties them together using some sort of feedback
feedback can help us hide that and prevent insertion attacks

- 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

not entirely true, the actual affects are limited, good for error propagation

- Decryption is similar

What About the First Block?

b/c there is no previous block, so you could encrypt every first block was mapped to same thing. ex. germans made the mistake and biritish found out hwo they talked to each other.

- 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

IV is used with CBC - both are very important, make sure to understand these keywords

- 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

how to do this: use XOR with a random string + first block

- 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

use previous encrypted blk 0 and plaintext blk 1

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?

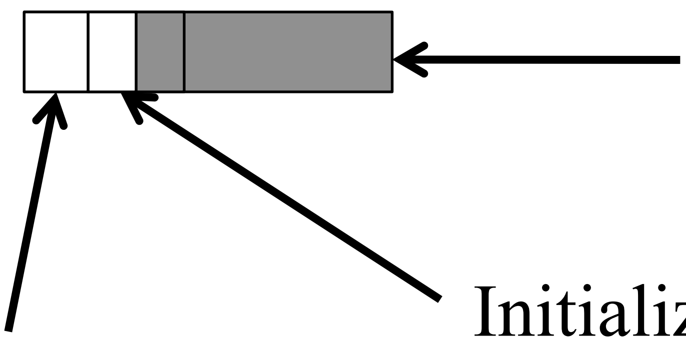
- First block received decrypts to

  $P = plaintext \oplus IV$ P is the plaintext XOR'ed with the IV, no problem if receiver knows IV

- $plaintext = P \oplus IV$
- No problem if receiver knows *IV* IV gets sent in plaintext
  - Typically, *IV* is sent in the message
- Subsequent blocks use standard CBC
  - So can be decrypted that way

when the second encrypted block comes in, we want to tru plaintext attacker could also flip a bit in the cipher text to mess the user up

An Example of IV Decryption



And XOR with the plaintext IV

IV is not related to the key
to decrypt, attacker needs the key,
not just the original message

ORIGINAL MESSAGE

1	1	0	1	0	0	0	1

0	1	0	0	1	1	0	0

Initialization Vector

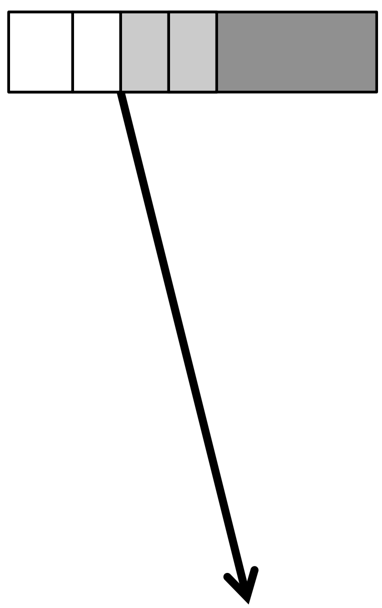
1	0	0	1	1	1	0	1

Plaintext

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)

just like you had a codebook

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

Both convert block to stream cipher

web browsers - security problem because someone was using a good cryptographic mode that wasn't really good for that browser.

Always make sure your cryptography is used by someone who made it before LOL

Uses of Cryptography

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

Cryptography and Secrecy

if you can describe your secret in NP complete problems
you can prove that you know the secret
a zero knowledge proof are not widely used, they are neat

- 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?

  only I could have created it…
  - 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 <u>without</u> 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

cryptography is expensive

- If you don't need secrecy, can get the same effect

attacker couldn't check the checkum in the encrypted perons

  - 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

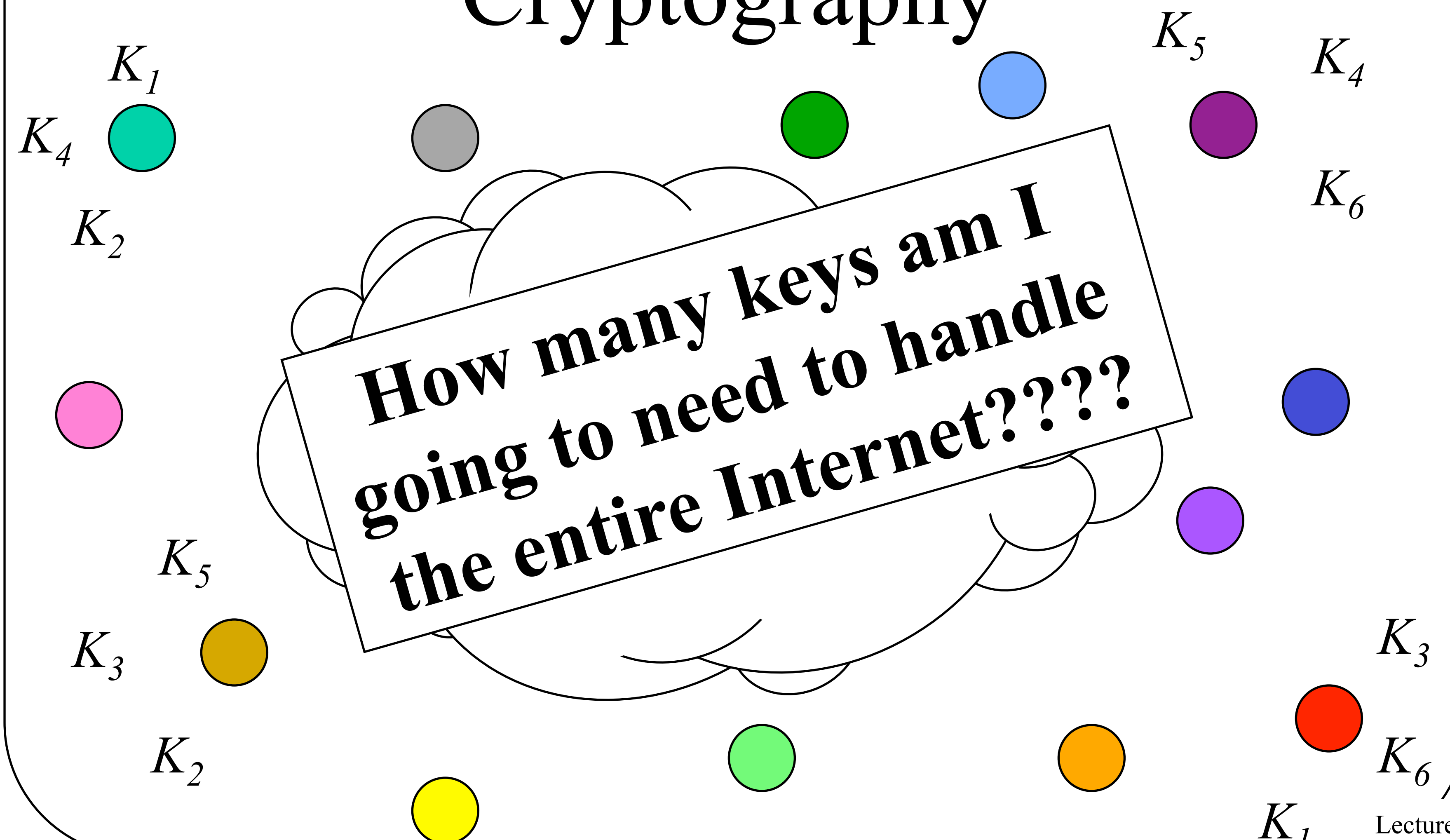
repudiation - say I appeared to have done something, but I did not

– Key distribution can be a problem

symmetric properties allows users to repudiate what they encrypted

– 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

brute force attacks can be possible if your key size changed

- Well known symmetric cipher
- Developed in 1977, still much used
  - Shouldn't be, for anything serious
- Block encryption, using substitutions, permutations, table lookups

NSA wants to be the only place where they can crack ciphers

  - With multiple *rounds* multiple rounds for the security you want
  - Each round is repeated application of operations like cipherblock chaining 2^55 keys
- Only serious problem based on short key

finding flaw in DES: there are a few keys that are needed to be avoided brute force attacks feasible on DES

The Advanced Encryption Standard

- A relatively new cryptographic algorithm
- Intended to be the replacement for DES
- Chosen by NIST

  open src this shit, allow everyone to compete and look at the best submissions
  - Through an open competition
- Chosen cipher was originally called Rijndael

  Dutcher
  - Developed by Dutch researchers
  - Uses combination of permutation and substitution

  128 or 256 bit key

Increased Popularity of AES

- Gradually replacing DES
  - As was intended

gradually replace DEFS

- 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

multiple problems: what if we only do a few rounds, but not all the rounds?
attacks are possible then

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

what looks like an impractical attack yesterday becomes serious tomorrow.

- Attacks on crypto only get better over time, never worse

Public Key Encryption Systems

- The encrypter and decrypter have different keys

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

Cipher = encrypt(key, plaintext)

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

plaintext = decrypt(key, cipher)

- 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

expensive computational operation

- One key is kept secret by the owner
- The other is made public to the world

public key - share it with the world

- If you want to send an encrypted message to someone, encrypt with his public key

encrypt with public key

  – Only he has private key to decrypt

his own private key

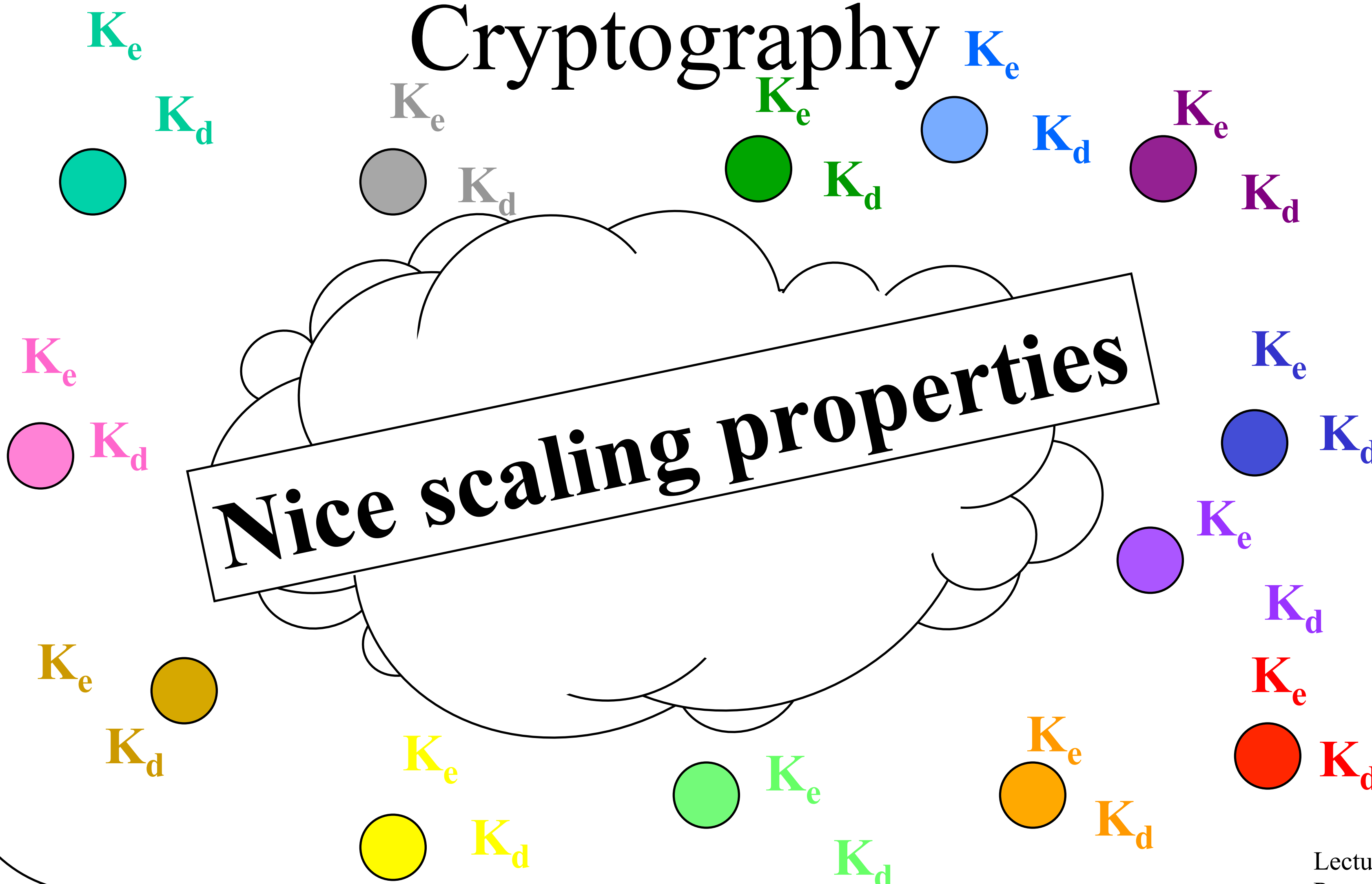
Authentication With Shared Keys

- 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

problems with distributing their public keys

- To communicate via shared key cryptography, key must be distributed
  - In trusted fashion

weakness of public key is distribution

- 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?

- <u>Conjectured</u> 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

It's Alice's key

Only Bob can decrypt it

Only Alice could have created it

Alice

K_{EA} K_{DA}

K_{EB}

Bob

K_{EB} K_{DB}

K_{EA}

K_S

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

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

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

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

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#l
^o,a 02

K_S

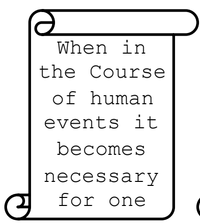
When in
the Course
of human
events it
becomes
necessary
for one

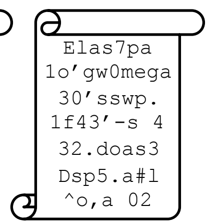
Signatures With Public Key Cryptography

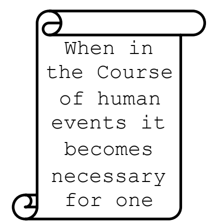
- Signer encrypts document with his private key
- Receiver checks validity by decrypting with signer's public key
- Only signer has the private key
  – So no trusted third party required
- But receiver must be certain that he has the right public key

For Example,

K_d







K_e

Alice's public key

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