Computer Law and Ethics, COSC-3325, Lecture 3

Stefan Andrei

Reminder of the last lecture

- Privacy and Computer Technology
- “Big Brother is Watching You”
- Privacy Topics
- Protecting Privacy
- Communications

Overview of This Lecture

- How Pretty Good Privacy (PGP) works?
  - The Basics of Cryptography
  - Public key cryptography
  - How PGP works?
  - The Diffie-Hellman protocol
- This lecture is inspired from Chapter 1 of the *Introduction to Cryptography* in the PGP 6.5.1 documentation. Copyright © 1990-1999 Network Associates, Inc. and its Affiliated Companies.
  - Available online at www.pgpi.org/doc/pgpintro/, accessed on January 2, 2011

Background

- Pretty Good Privacy (PGP) is a data encryption and decryption computer program that provides cryptographic privacy and authentication for data communication.
- PGP is often used for signing, encrypting and decrypting texts, E-mails, files, directories and whole partitions to increase the security of e-mail communications.
- It was created by Philip Zimmermann in 1991.

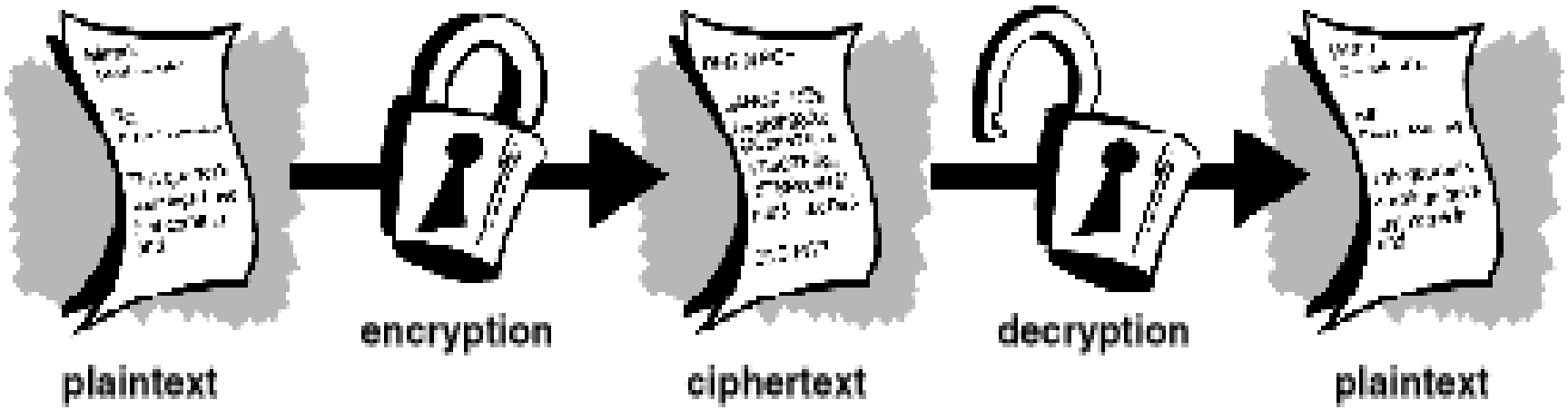
The Basics of Cryptography

- When Julius Caesar sent messages to his generals, he didn't trust his messengers.
- So he replaced every ‘A’ in his messages with a ‘D’, every ‘B’ with an ‘E’, and so on through the alphabet.
- Only someone who knew the "shift by 3" rule could decipher his messages.
- And so we begin.

Encryption and decryption

- Data that can be read and understood without any special measures is called *plaintext* or *cleartext.*
- The method of disguising plaintext in such a way as to hide its substance is called *encryption.*
- *Encrypting* plaintext results in unreadable gibberish called *ciphertext.*
- Encryption is used to ensure that information is hidden from anyone for whom it is not intended, even those who can see the encrypted data.
- The process of reverting ciphertext to its original plaintext is called *decryption.*

Encryption and decryption (cont)



What is cryptography?

- *Cryptography* is the science of using mathematics to encrypt and decrypt data.
- Cryptography enables one to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient.
- While cryptography is the science of securing data, *cryptanalysis* is the science of analyzing and breaking secure communication.

What is cryptology?

- Classical cryptanalysis involves an interesting combination of analytical reasoning, application of mathematical tools, pattern finding, patience, determination, and luck.
- Cryptanalysts are also called *attackers.*
- *Cryptology* embraces both cryptography and cryptanalysis.

Strong cryptography

- *"There are two kinds of cryptography in this world: cryptography that will stop your kid sister from reading your files, and cryptography that will stop major governments from reading your files.*
- *PGP is about the latter."*
- --Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C.
- Cryptography can be *strong* or *weak*, as explained above.

Strong cryptography (cont)

- *Cryptographic strength* is measured in the time and resources it would require to recover the plaintext.
- The result of *strong cryptography* is ciphertext that is very difficult to decipher without possession of the appropriate decoding tool.
- How difficult? Given all of today's computing power and available time - even a billion computers doing a billion checks a second - it is not possible to decipher the result of strong cryptography before the end of the universe.

PGP protection

- One would think, then, that strong cryptography would hold up rather well against even an extremely determined cryptanalyst.
- Who's really to say? No one has proven that the strongest encryption obtainable today will hold up under tomorrow's computing power.
- However, the strong cryptography employed by PGP is the best available today.
- Vigilance and conservatism will protect us better, however, than claims of impenetrability.

How does cryptography work?

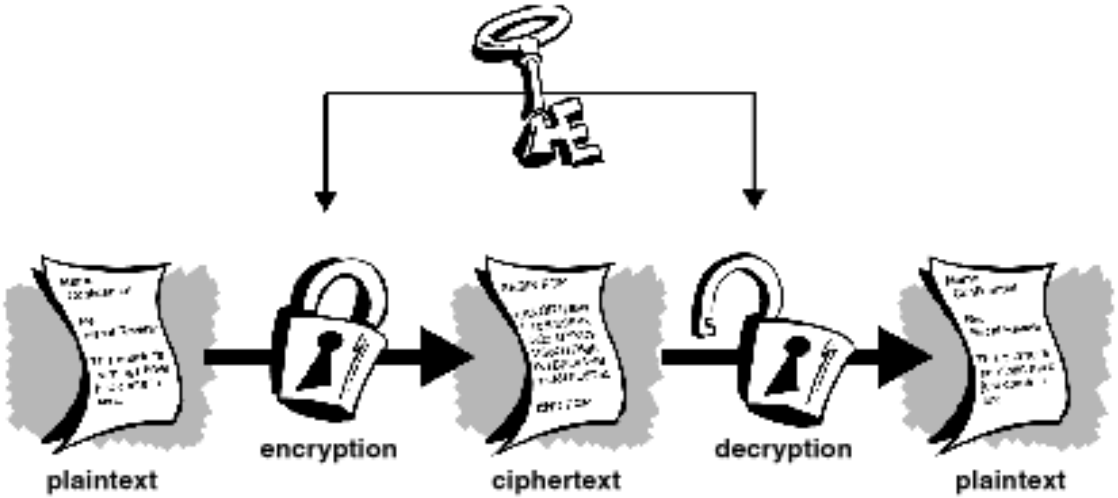
- A *cryptographic algorithm*, or *cipher*, is a mathematical function used in the encryption and decryption process.
- A cryptographic algorithm works in combination with a *key — a word, number,* or *phrase* – to encrypt the plaintext.
- The same plaintext encrypts to different ciphertext with different keys.

Security of the encrypted data

- The security of encrypted data is entirely dependent on two things:
  - the strength of the cryptographic algorithm and
  - the secrecy of the key.
- A cryptographic algorithm, plus all possible keys and all the protocols that make it work comprise a *cryptosystem.*
- *PGP* is a cryptosystem.

Conventional cryptography

- In conventional cryptography, also called *secret-key* or *symmetric key* encryption, one key is used both for encryption and decryption.
- The Data Encryption Standard (DES) is an example of a conventional cryptosystem that was widely employed by the Federal Government.



Caesar's Cipher

- An extremely simple example of conventional cryptography is a substitution cipher.
- A substitution cipher substitutes one piece of information for another.
- This is most frequently done by offsetting letters of the alphabet.
- Two examples are Captain Midnight's Secret Decoder Ring, which you may have owned when you were a kid, and Julius Caesar's cipher.
- In both cases, the algorithm is to offset the alphabet and the key is the number of characters to offset it.

Example

- If we encode the word "SECRET" using Caesar's key value of 3, we offset the alphabet so that the 3rd letter down (D) begins the alphabet.
- So starting with

  ABCDEFGHIJKLMNOPQRSTUVWXYZ
- and sliding everything up by 3, you get

  DEFGHIJKLMNOPQRSTUVWXYZABC
- where D=A, E=B, F=C, and so on.

Example (cont)

- Using this scheme, the plaintext,

  "SECRET"
- encrypts as

  "VHFUHW."
- To allow someone else to read the ciphertext, one tells them that the key is 3.
- Obviously, this is exceedingly weak cryptography by today's standards, but hey, it worked for Caesar, and it illustrates how conventional cryptography works.

Key management and conventional encryption

- Conventional encryption has benefits.
- It is very fast.
- It is especially useful for encrypting data that is not *going anywhere*.
- However, conventional encryption alone as a means for transmitting secure data can be quite expensive simply due to the difficulty of secure key distribution.

Example

- Recall a character from your favorite spy movie: the person with a locked briefcase handcuffed to his or her wrist.
- What is in the briefcase, anyway?
- It is probably not the missile launch code / biotoxin formula / invasion plan itself.
- It is the *key* that will decrypt the secret data.

Secure communication

- For a sender and recipient to communicate securely using conventional encryption, they must agree upon a key and keep it secret between themselves.
- If they are in different physical locations, they must trust a courier, the Bat Phone, or some other secure communication medium to prevent the disclosure of the secret key during transmission.

Hiding the key …

- Anyone who overhears or intercepts the key in transit can later read, modify, and forge all information encrypted or authenticated with that key.
- From DES to Captain Midnight's Secret Decoder Ring, the persistent problem with conventional encryption is *key distribution*: how do you get the key to the recipient without someone intercepting it?

Public key cryptography

- The problems of key distribution are solved by *public key cryptography*, the concept of which was introduced by Whitfield Diffie and Martin Hellman in 1975.
- There is now evidence that the British Secret Service invented it a few years before Diffie and Hellman, but kept it a military secret - and did nothing with it.
- [J H Ellis: The Possibility of Secure Non-Secret Digital Encryption, *CESG Report*, January 1970]

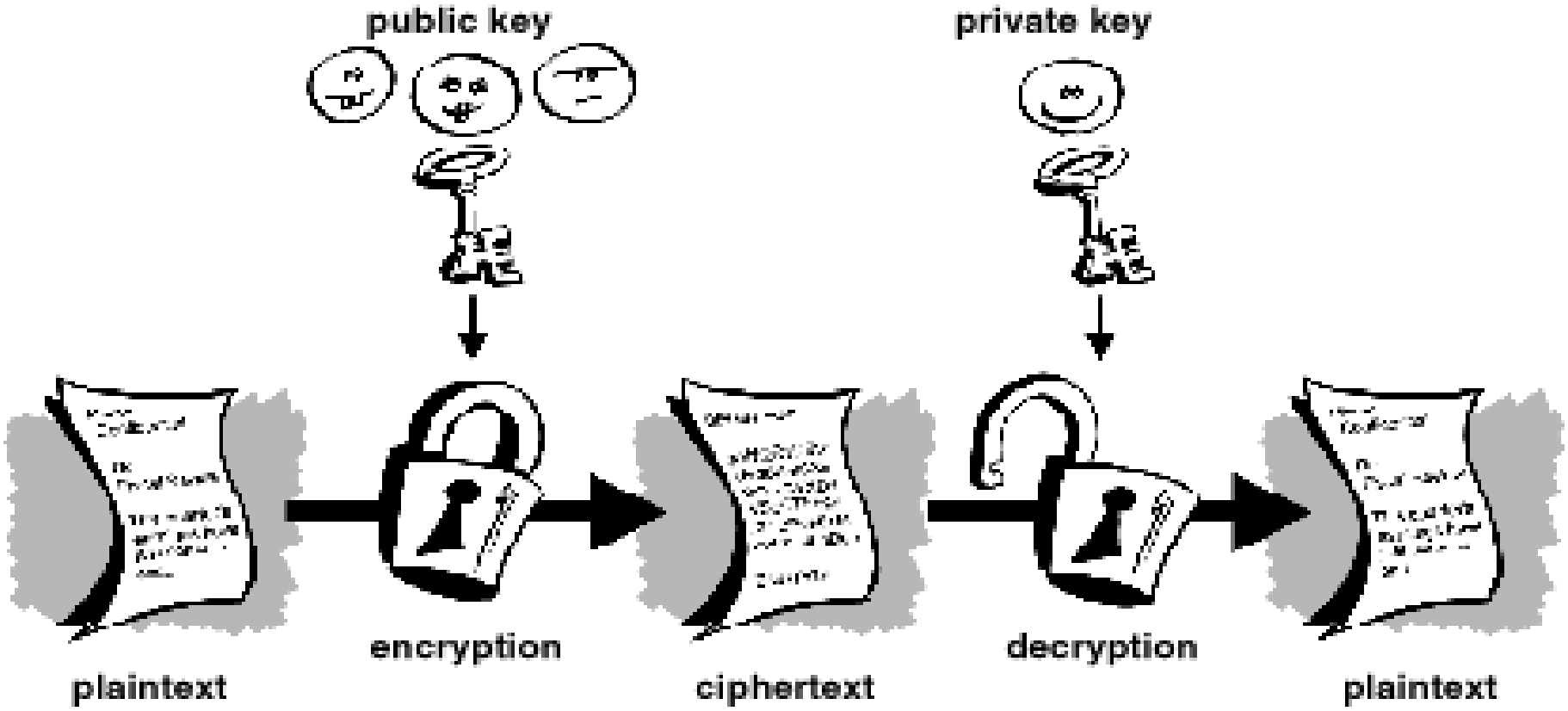
Main idea of public key cryptography

- Public key cryptography is an asymmetric scheme that uses a *pair* of keys for encryption: a *public key*, which encrypts data, and a corresponding *private*, or *secret key* for decryption.
- You publish your public key to the world while keeping your private key secret.
- Anyone with a copy of your public key can then encrypt information that only you can read.
- Even people you have never met.

How feasible is this idea?

- It is computationally infeasible to deduce the private key from the public key.
- Anyone who has a public key can encrypt information but cannot decrypt it.
- Only the person who has the corresponding private key can decrypt the information.

Public key encryption



The primary benefit of public key cryptography …

- … is that it allows people who have no preexisting security arrangement to exchange messages securely.
- The need for sender and receiver to share secret keys via some secure channel is eliminated;
  - all communications involve only public keys, and no private key is ever transmitted or shared.

Examples of public-key cryptosystems

- Elgamal (named for its inventor, Taher Elgamal),
- RSA (named for its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman),
- Diffie-Hellman (named, you guessed it, for its inventors), and
- DSA, the Digital Signature Algorithm (invented by David Kravitz).

Technological revolution

- Because conventional cryptography was once the only available means for relaying secret information, the expense of secure channels and key distribution relegated its use only to those who could afford it, such as governments and large banks.
- Public key encryption is the technological revolution that provides strong cryptography to the adult masses.
- Remember the courier with the locked briefcase handcuffed to his wrist? Public-key encryption puts him out of business (probably to his relief).

How PGP works?

- PGP combines some of the best features of both conventional and public key cryptography.
- PGP is a *hybrid cryptosystem*.
- When a user encrypts plaintext with PGP, PGP first compresses the plaintext.
- Data compression saves modem transmission time and disk space and, more importantly, strengthens cryptographic security.

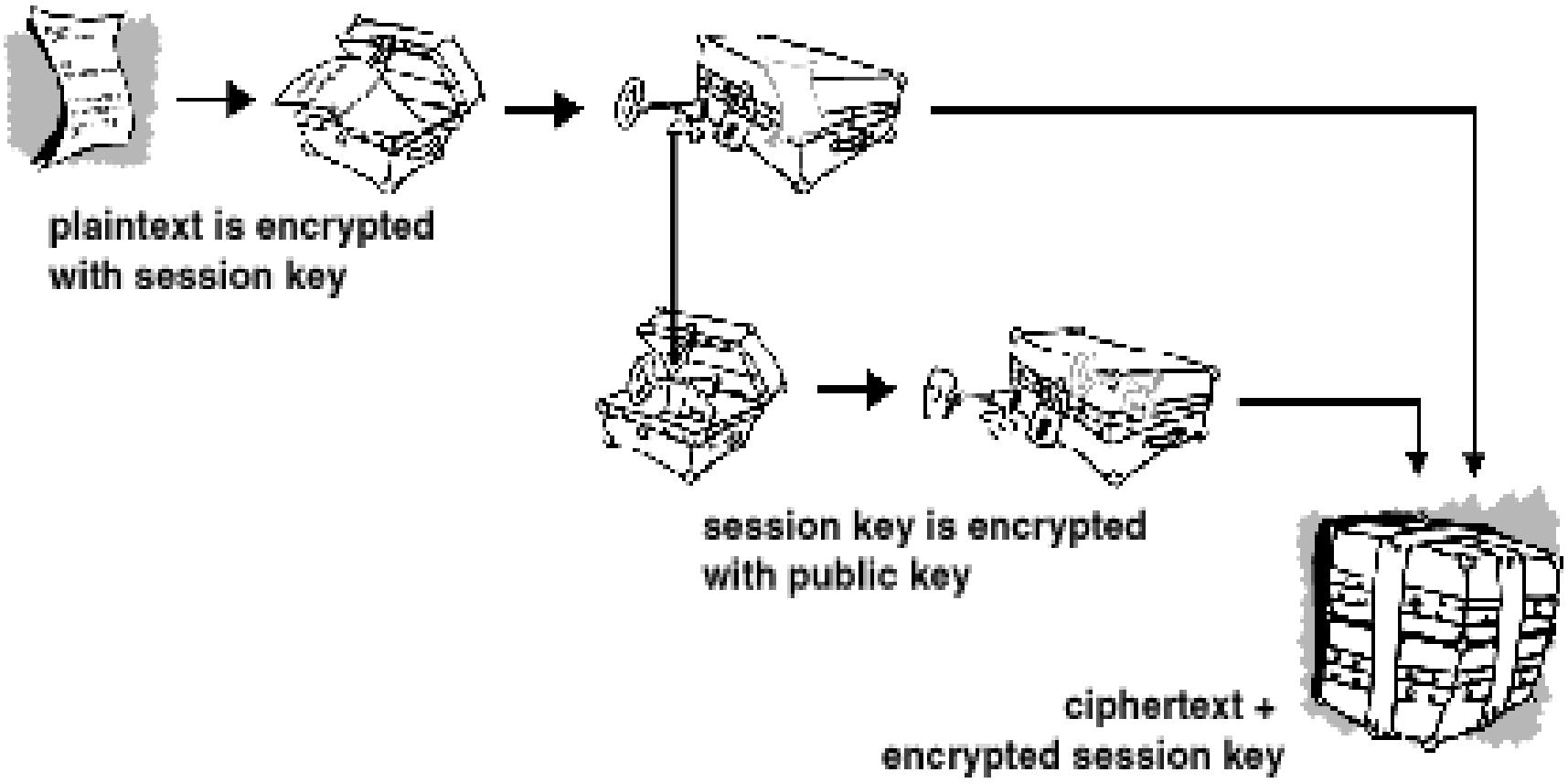
How PGP works? (cont)

- Most cryptanalysis techniques exploit patterns found in the plaintext to crack the cipher.
- Compression reduces these patterns in the plaintext, thereby greatly enhancing resistance to cryptanalysis.
- PGP then creates a *session key*, which is a one-time-only secret key.

How PGP works? (cont)

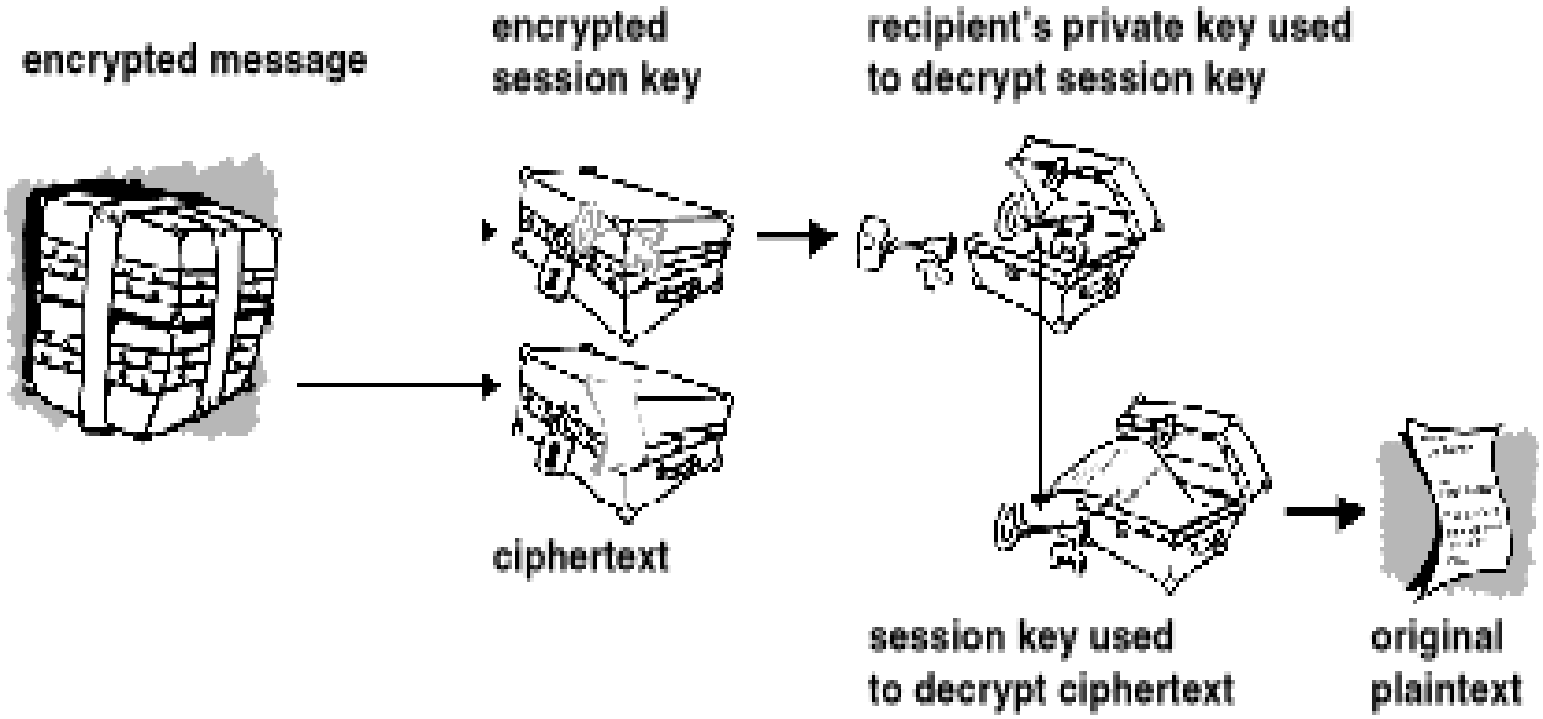
- This key is a random number generated from the random movements of your mouse and the keystrokes you type.
- This session key works with a very secure, fast conventional encryption algorithm to encrypt the plaintext; the result is ciphertext.
- Once the data is encrypted, the session key is then encrypted to the recipient's public key.
- This public key-encrypted session key is transmitted along with the ciphertext to the recipient.

How PGP works? (cont)



Decryption works in the reverse

- The recipient's copy of PGP uses his or her private key to recover the temporary session key, which PGP then uses to decrypt the conventionally-encrypted ciphertext.



Comments about PGP method

- The combination of the two encryption methods combines the convenience of public key encryption with the speed of conventional encryption.
- Conventional encryption is about 1,000 times faster than public key encryption.
- Public key encryption in turn provides a solution to key distribution and data transmission issues.
- Used together, performance and key distribution are improved without any sacrifice in security.

Keys

- A key is a value that works with a cryptographic algorithm to produce a specific ciphertext.
- Keys are basically really, really, really big numbers.
- Key size is measured in bits; the number representing a 1024-bit key is darn huge.
- In public key cryptography, the bigger the key, the more secure the ciphertext.

Keys (cont)

- However, public key size and conventional cryptography's secret key size are totally unrelated.
- A conventional 80-bit key has the equivalent strength of a 1024-bit public key.
- A conventional 128-bit key is equivalent to a 3000-bit public key.
- Again, the bigger the key, the more secure, but the algorithms used for each type of cryptography are very different and thus comparison is like that of apples to oranges.

Keys (cont)

- While the public and private keys are mathematically related, it's very difficult to derive the private key given only the public key.
- However, deriving the private key is always possible given enough time and computing power.
- This makes it very important to pick keys of the right size; large enough to be secure, but small enough to be applied fairly quickly.
- Additionally, you need to consider who might be trying to read your files, how determined they are, how much time they have, and what their resources might be.

Keys (cont)

- Larger keys will be cryptographically secure for a longer period of time.
- If what you want to encrypt needs to be hidden for many years, you might want to use a very large key.
- Of course, who knows how long it will take to determine your key using tomorrow's faster, more efficient computers?
- There was a time when a 56-bit symmetric key was considered extremely safe.

Keys (cont)

- Keys are stored in encrypted form.
- PGP stores the keys in two files on your hard disk; one for public keys and one for private keys.
- These files are called *keyrings*.
- As you use PGP, you will typically add the public keys of your recipients to your public keyring.
- Your private keys are stored on your private keyring.
- If you lose your private keyring, you will be unable to decrypt any information encrypted to keys on that ring.

The Diffie-Hellman protocol

- Although Diffie–Hellman key agreement itself is an *anonymous (non-authenticated)* key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide perfect forward secrecy in Transport Layer Security's ephemeral modes (referred to as EDH or DHE depending on the cipher suite).

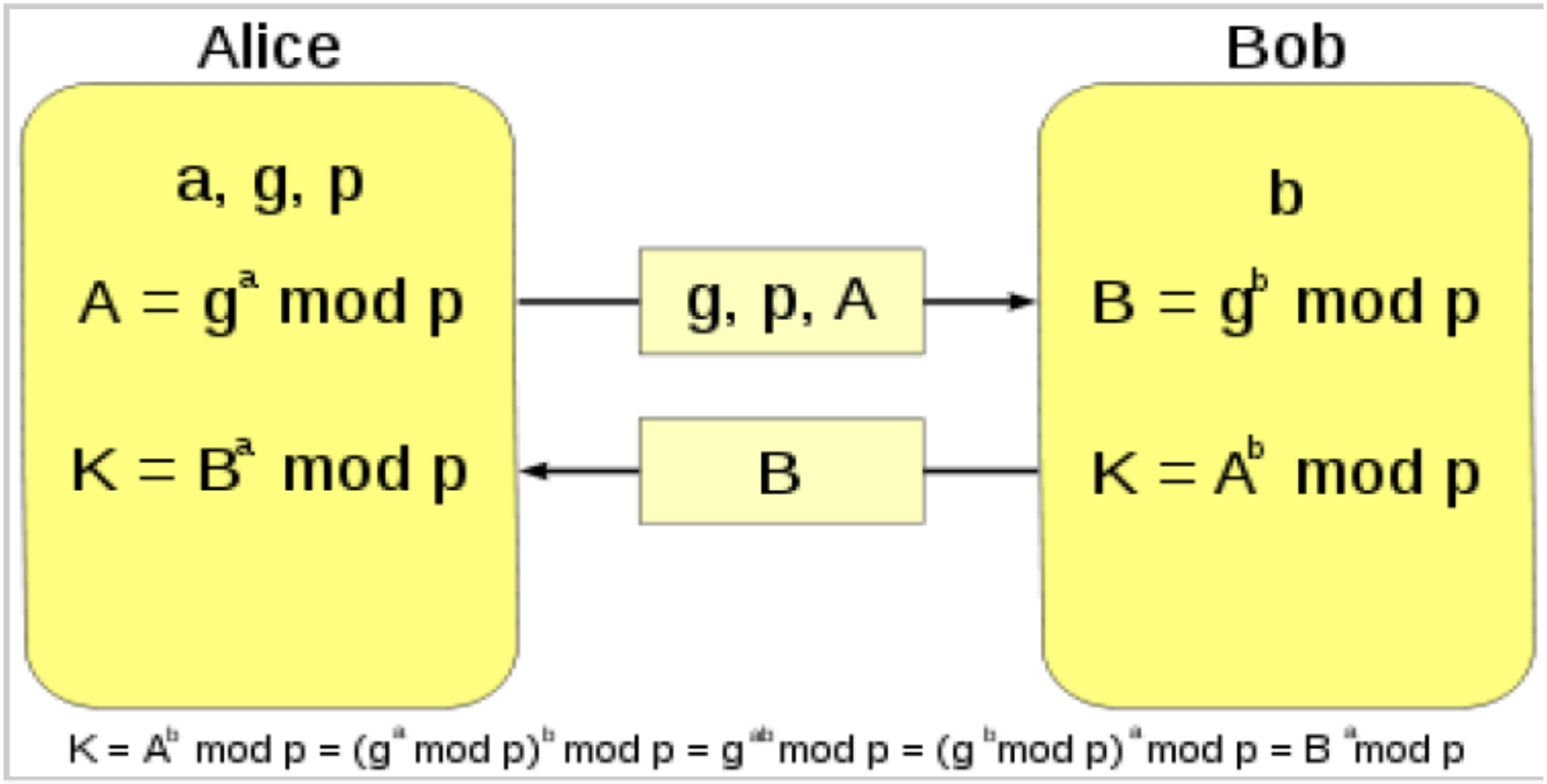
Description

- Diffie–Hellman protocol establishes a shared secret key that can be used for secret communications by exchanging data over a public network.
- The information to be sent is converted to (positive) integers.
- The simplest, and original, implementation of the protocol uses the multiplicative group of integers modulo *p*, where *p* is prime and *g* is primitive root mod *p.*

What does 'mod' operator mean?

- '*g* mod *p*' means the remainder of the division of *g* to *p*.
- **Example 1:**
  - Let *g* = 101 and *p* = 13.
  - Then *g* mod *p* = 10 because 101 = 13 * 7 + 10.
- **Example 2:**
  - Let *g* = 5, a = 3, and *p* = 7.
  - Then g^a mod *p* = 125 mod 7 = 6 because 125 = 7 * 17 + 6.

The Diffie-Hellman key exchange



Example of the protocol

Alice				Bob		
Secret	Public	Calculates	Sends	Calculates	Public	Secret
a	p, g		p,g $\longrightarrow$			b
a	p, g, A	$g^a \bmod p = A$	A $\longrightarrow$		p, g	b
a	p, g, A		$\longleftarrow$ B	$g^b \bmod p = B$	p, g, A, B	b
a, **s**	p, g, A, B	$B^a \bmod p = s$		$A^b \bmod p = s$	p, g, A, B	b, **s**

Example description step by step

- Non-secret values are in green, and secret values are in **boldface red**:

1. Alice and Bob agree to use a prime number $p=23$ and base $g=5$.
2. Alice chooses a secret integer, say, $\mathbf{a=6}$, then sends Bob $A = g^a \bmod p$
   - $A = 5^6 \bmod 23$
   - $A = \mathbf{15{,}625} \bmod 23$
   - $A = 8$

Example description step by step (cont)

3. Bob chooses a secret integer, say, ***b*=15**, then sends Alice B = g^b mod p
   - B = 5^{15} mod 23
   - B = **30,517,578,125** mod 23
   - B = 19
4. Alice computes **s** = B^a mod p
   - **s** = 19^6 mod 23
   - **s** = **47,045,881** mod 23
   - **s** = **2**

Example description step by step (cont)

5. Bob computes **s** = A^a mod p
   - **s** = 8^{15} mod 23
   - **s** = **35,184,372,088,832** mod 23
   - **s** = **2**
6. Alice and Bob now share a secret: **s** = **2**. This is because of multiplication commutativity, that is, 6*15 is the same as 15*6.

Example description step by step (cont)

- Somebody who had known both private (6, 15) integers might also have calculated **s** as follows:
  - **s** = 5^{6*15} mod 23
  - **s** = 5^{15*6} mod 23
  - **s** = 5^{90} mod 23
  - **s** = 807,793,566,946,316,088,741,610,050,849,573,099,185,363,389,551,639,556,884,765,625 mod 23
  - **s** = **2**

Comments about the protocol

- Both Alice and Bob have arrived at the same value, because g^{ab} and g^{ba} are equal mod p.
- Note that only a, b and $g^{ab} = g^{ba}$ mod p are kept secret.
- All the other values – p, g, g^a mod p, and g^b mod p – are sent in the clear.
- Once Alice and Bob compute the shared secret, they can use it as an encryption key, known only to them, for sending messages across the same open communications channel.

Comments about the protocol (cont)

- Of course, much larger values of *a, b,* and *p* would be needed to make this example secure, since it is easy to try all the possible values of g^{ab} mod *23.*
- There are only 23 possible integers as the result of mod 23.
- If *p* were a prime of at least 300 digits, and *a* and *b* were at least 100 digits long, then even the best algorithms known today could not find *a* given only *g, p,* g^b mod *p* and g^a mod *p,* even using all of mankind's computing power.

Comments about the protocol (cont)

- The problem is known as the discrete logarithm problem.
- Note that *g* need not be large at all, and in practice is usually either *2* or *5.*

A general description of the protocol

1. Alice and Bob agree on a finite cyclic group *G* and a generating element *g* in *G.* (This is usually done long before the rest of the protocol; *g* is assumed to be known by all attackers.)
2. Alice picks a random natural number *a* and sends g^a to Bob.
3. Bob picks a random natural number *b* and sends g^b to Alice.
4. Alice computes $(g^b)^a$.
5. Bob computes $(g^a)^b$.

- Both Alice and Bob are now in possession of the group element g^{ab}, which can serve as the shared secret key. The values of $(g^b)^a$ and $(g^a)^b$ are the same because groups are power associative.

How secure is this protocol?

- Eve is an eavesdropper - she watches what is sent between Alice and Bob, but she does not alter the contents of their communications.
- Let us identify what Alice, Bob, and Eve know.
- Let s = shared secret key. s = 2
- Let g = public base. g = 5
- Let p = public (prime) number. p = 23
- Let a = Alice's private key. a = 6
- Let A = Alice's public key. $A = g^a \bmod p = 8$
- Let b = Bob's private key. b = 15
- Let B = Bob's public key. $B = g^b \bmod p = 19$

What Alice knows?

- It should be difficult for Alice to solve for Bob's private key.

Alice	
knows	doesn't know
p = 23	b = ?
base g = 5	
a = 6	
$A = 5^6 \bmod 23 = 8$	
$B = 5^b \bmod 23 = 19$	
$s = 19^6 \bmod 23 = 2$	
$s = 8^b \bmod 23 = 2$	
$s = 19^6 \bmod 23 = 8^b \bmod 23$	
s = 2	

What Bob knows?

- It should be difficult for Bob to solve for Alice's private key.

Bob	
knows	doesn't know
p = 23	a = ?
base g = 5	
b = 15	
$B = 5^{15} \bmod 23 = 19$	
$A = 5^{a} \bmod 23 = 8$	
$s = 8^{15} \bmod 23 = 2$	
$s = 19^{a} \bmod 23 = 2$	
$s = 8^{15} \bmod 23 = 19^{a} \bmod 23$	
s = 2	

What Eva knows?

- If it isn't difficult for Alice to solve for Bob's private key (or vice versa), Eve may simply substitute her own private / public key pair, plug Bob's public key into her private key, produce a fake shared secret key, and solve for Bob's private key (and use that to solve for the shared secret key.

Eve	
knows	doesn't know
p = 23	a = ?
base g = 5	b = ?
	s = ?
$A = 5^a \bmod 23 = 8$	
$B = 5^b \bmod 23 = 19$	
$s = 19^a \bmod 23$	
$s = 8^b \bmod 23$	
$s = 19^a \bmod 23 = 8^b \bmod 23$	

How secure is this protocol?

- Eve may attempt to choose a public / private key pair that will make it easy for her to solve for Bob's private key.
- However, the protocol is considered secure against eavesdroppers if *p* and *g* are chosen properly.
- The eavesdropper ("Eve") would have to solve the Diffie–Hellman problem to obtain g^{ab}.
- This is currently considered difficult.

How to make the protocol secure?

- An efficient algorithm to solve the discrete logarithm problem would make it easy to compute *a* or *b* and solve *the Diffie–Hellman problem*, making this and many other public key cryptosystems insecure.
- The order of *G* should be prime or have a large prime factor to prevent use of the *Pohlig–Hellman* algorithm to obtain *a* or *b.*
- For this reason, a Sophie Germain prime *q* is sometimes used to calculate *p=2q+1,* called a safe prime, since the order of *G* is then only divisible by *2* and *q.*
- *g* is then sometimes chosen to generate the order *q* subgroup of *G*, rather than *G*, so that the Legendre symbol of g^a never reveals the low order bit of *a.*

How to make the protocol secure? (cont)

- If Alice and Bob use random number generators whose outputs are not completely random and can be predicted to some extent, then Eve's task is much easier.
- The secret integers *a* and *b* are discarded at the end of the session.
- Therefore, Diffie–Hellman key exchange by itself trivially achieves perfect forward secrecy because no long-term private keying material exists to be disclosed.

The man-in-the-middle attacks

- In the original description, the Diffie–Hellman exchange by itself does not provide authentication of the communicating parties and is thus vulnerable to a man-in-the-middle attack.
- A person in the middle may establish two distinct Diffie–Hellman key exchanges, one with Alice and the other with Bob, effectively masquerading as Alice to Bob, and vice versa, allowing the attacker to decrypt (and read or store) then re-encrypt the messages passed between them.
- A method to authenticate the communicating parties to each other is generally needed to prevent this type of attack.

Password-authenticated key agreement

- When Alice and Bob share a password, they may use a password-authenticated key agreement (PAKE) form of Diffie–Hellman to prevent man-in-the-middle attacks.
- One simple scheme is to make the generator *g* the password.
- A feature of these schemes is that an attacker can only test one specific password on each iteration with the other party, and so the system provides good security with relatively weak passwords.
- This approach is described in ITU-T Recommendation X.1035, which is used by the G.hn home networking standard (this is beyond the scope on this course).

Public key

- It is also possible to use Diffie–Hellman as part of a public key infrastructure.
- Alice's public key is simply (g^a, g, p).
- To send her a message Bob chooses a random b, and then sends Alice g^b (unencrypted) together with the message encrypted with symmetric key $(g^a)^b$.
- Only Alice can decrypt the message because only she has a.
- A preshared public key also prevents man-in-the-middle attacks.

Public key (cont)

- In practice, Diffie–Hellman is not used in this way, with RSA being the dominant public key algorithm.
- This is largely used for historical and commercial reasons, namely that RSA created a Certificate Authority that became Verisign.
- Diffie–Hellman cannot be used to sign certificates, although the ElGamal and DSA signature algorithms are related to it.
- However, it is related to MQV, STS and the IKE component of the IPsec protocol suite for securing Internet Protocol communications.

Conclusion about PGP

- Pretty Good Privacy (PGP) is a data encryption and decryption computer program that provides cryptographic privacy and authentication for data communication.
- Diffie–Hellman protocol establishes a shared secret key that can be used for secret communications by exchanging data over a public network.
- The secure protocols use a password-authenticated key agreement (PAKE) form of Diffie–Hellman to prevent man-in-the-middle attacks.

Summary

- How Pretty Good Privacy (PGP) works?
  - This lecture is inspired from Chapter 1 of the *Introduction to Cryptography* in the PGP 6.5.1 documentation. Copyright © 1990-1999 Network Associates, Inc. and its Affiliated Companies.
  - Available online at http://www.pgpi.org/doc/pgpintro/, accessed on January 2, 2011

Coming up next

- Issues about computer malware
- Based on the paper:
  - Eugene H. Spafford: Privacy and Security: Remembrances of Things Pest, *Communications of the ACM*, August 2010, vol. 53, no. 8

Thank you for your attention!

Questions?