SI110
Asymmetric Encryption

Major Brian Hawkins, USMC

U.S. Naval Academy

Fall AY 2018

Outline

Review

What we've learned about encryption so far...

- Symmetric encryption
  - A shared secret (key) is used to both encrypt and decrypt the message
  - Key distribution is a **huge** issue
- Caesar and Vigenere Ciphers
  - Simplistic ciphers that can easily be broken using brute force attacks or frequency analysis
- We didn't address it, but in practice...
  - Much more complex encryption algorithms are used that are not susceptible to these naïve attacks
  - But the key distribution problem remains...

Review Continued

One-time pads (OTPs)...

- Key as long (or longer) than message
- Key is NEVER used again
- ...is provable secure/unbreakable ...
- But the key distribution problem remains...

Asymmetric Cryptography

Asymmetric/Public-Key Cryptography – synonymous

What is it?

- Each party has two keys, known as a *key pair*
  - *public* key – shared with the world
  - *private* key – known only to you

What does it do?

The public and private keys are mathematically related such that:

- Something encrypted with a user's public key can only be decrypted with a user's private key
- Something encrypted with a user's private key can only be decrypted with a user's public key

What does it do?

Like symmetric encryption, asymmetric encryption protects some pillars of cyber security:

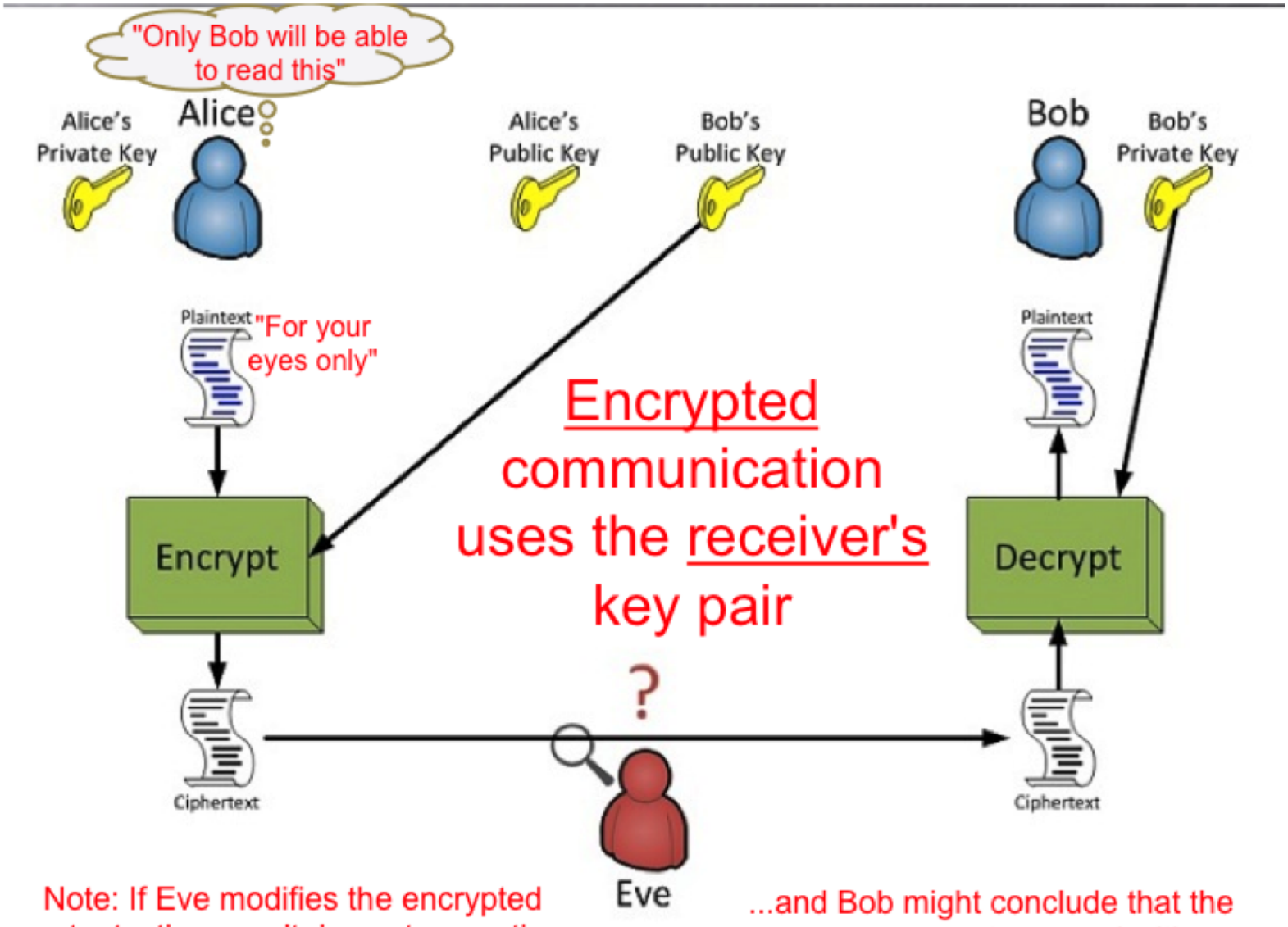
- Confidentiality
  - Given a CT, an adversary cannot tell what the original PT was
- Non-repudiation
  - Digital signatures provide the proof that a message originated from a particular individual/entity
- Depending on the implementation, can also be used to provide:
  - Authentication
  - Integrity

How does it work?

Encryption Process

1. PT is encrypted using the recipient's public key by the sender
   - Recall that this is available to anyone
2. CT is transmitted to the recipient
3. CT is decrypted by the recipient using the recipient's private key
   - Only the recipient's private key will decrypt the message
   - Only the recipient should possess this private key

Graphically. . .



Authenticated Communication

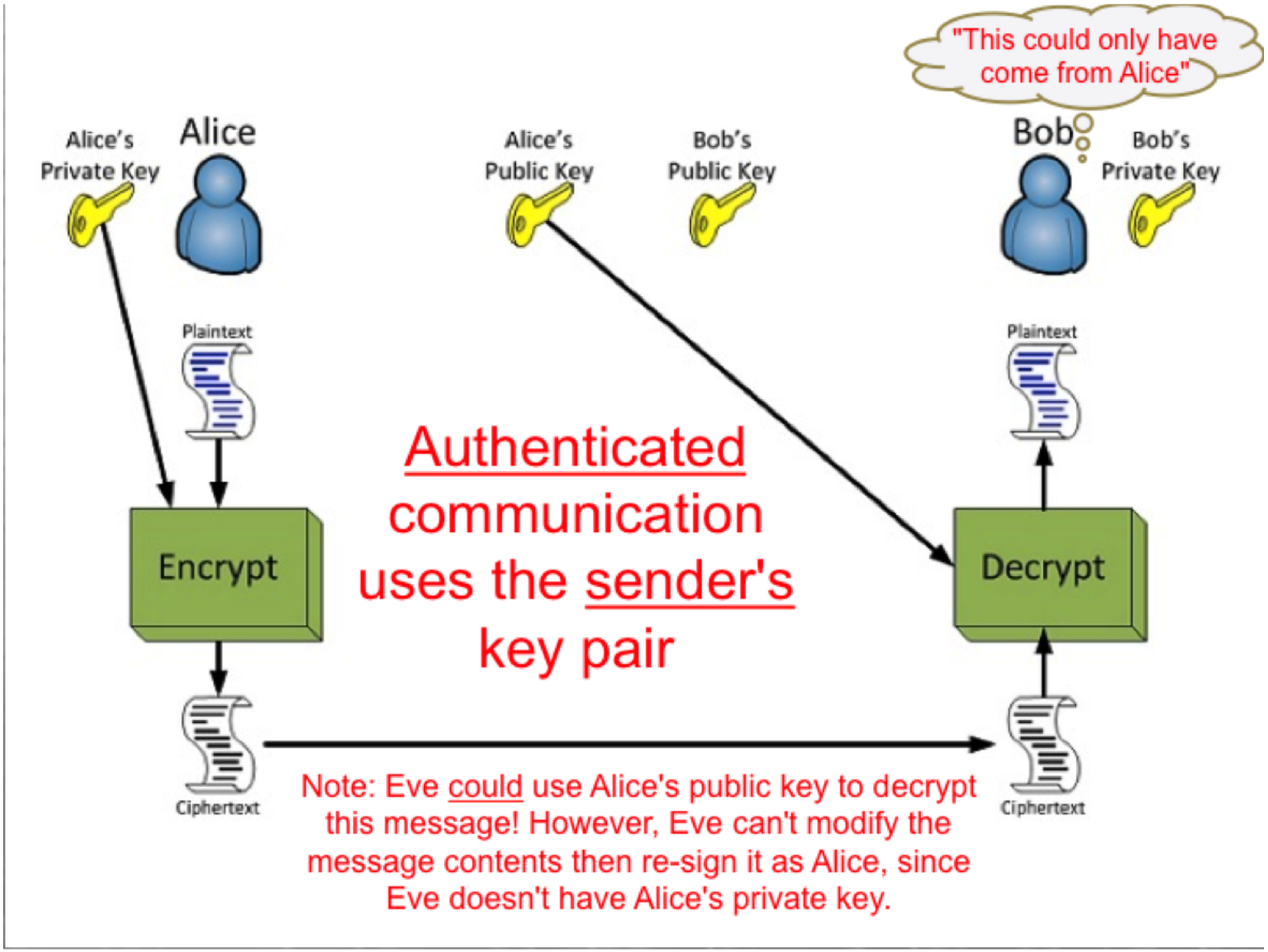
How can we be sure a message could only have come from one individual?

Authenticated Communication

1. The sender encrypts their message using their private key.
2. The sender transmits the message to the recipient.
3. The recipient then decrypts the message using the sender's public key.

Note that an eavesdropper *could* decrypt the message using the sender's public key – but they couldn't modify the message and send it to the recipient because they don't have the sender's private key!

Graphically. . .



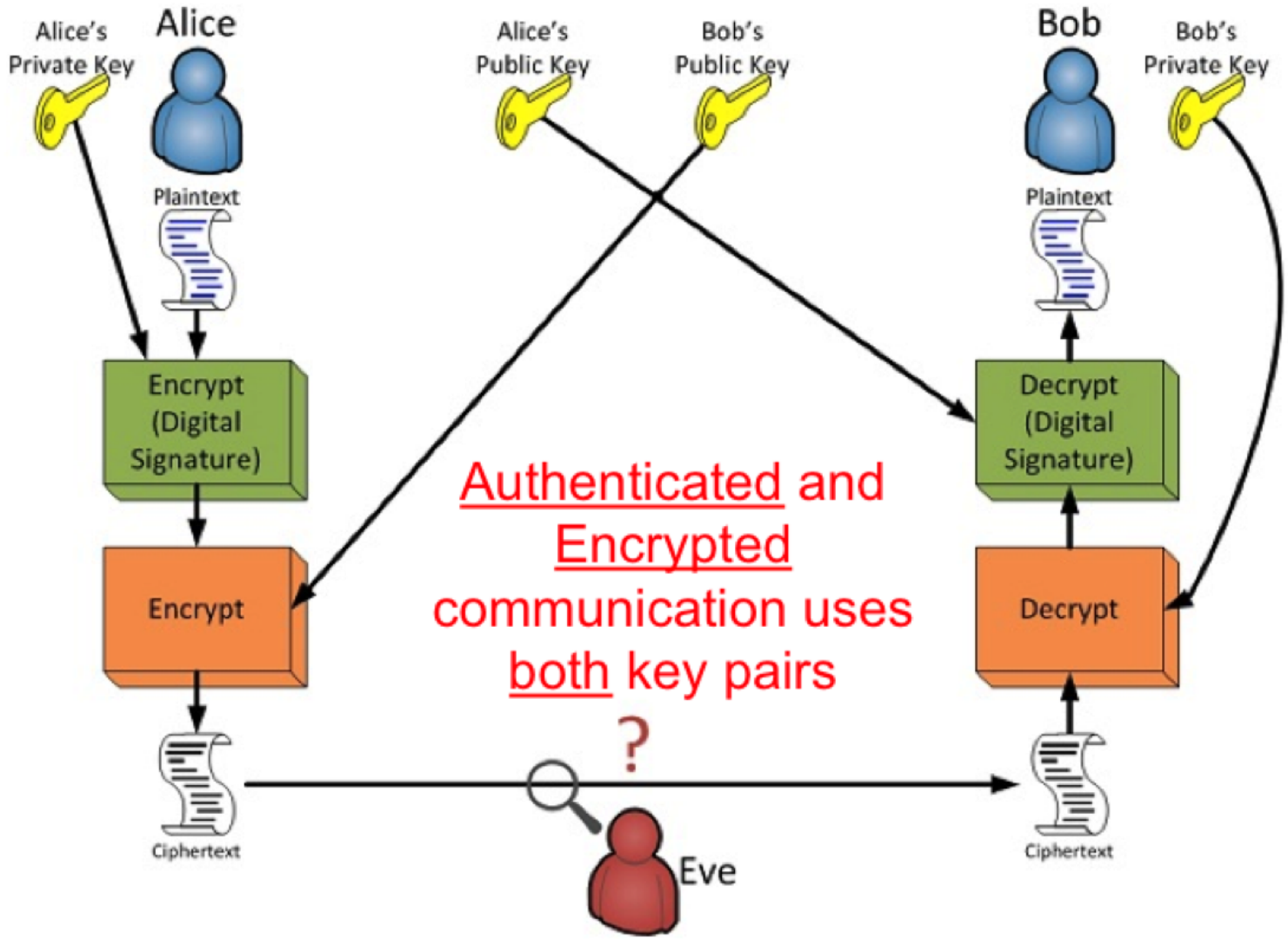
With our powers combined...

In order to send a message that's unreadable by eavesdroppers and ensure that the message could only have come from the sender, we can combine the two encryption methods!

1. The sender encrypts the message with their own private key, then encrypts it with the recipient's public key
2. The sender then transmits the message to the recipient
3. To decrypt, the recipient reverses the process
4. First, they decrypt the message with their private key
5. Then, they decrypt the message with the sender's public key

This ensures confidentiality of the data in transit, as well as ensures that we know that the message could only have been sent by the sender

Graphically. . .



RSA Encryption

The classic example of asymmetric encryption: the RSA (Rivest-Shamir-Adleman) cryptosystem.

Preliminaries

- Used for both encryption and digital signatures (later)
- Its security is based on the *discrete logarithm* problem
  - Short story: factoring large numbers is very, very inefficient on computers
- Employs two keys:
  - Public key: (e, n)
  - Private key: (d, n)
  - $n = pq$, where p, q are large primes (100s of digits long)
  - e and d are multiplicative inverses modulo $(p-1)(q-1)$, which is vital for this scheme to work
  - Public key is released to the world, while the private key is kept secret
    - p, q also need to be kept secret, or d can be computed by an adversary

RSA continued

In order to use RSA, we generate an RSA key-pair (`OpenSSL` is an open-source tool to do so)

How it works

- Assume our public key is (e, n) and our private key is (d, n).
- Further assume the message is m, and the encrypted message is c.
- To send an encrypted message to us, a sender computes:

$$c \equiv m^e (\mathrm{mod} n)$$

- The sender sends us c.
- To decrypt, the we compute:

$$m \equiv c^d (\mathrm{mod} n) \equiv m^{ed} (\mathrm{mod} n)$$

RSA security

How good is it?

- While there are a few bad e/d choices, most of the security of RSA lies in the length of n.
- If n is 300 bits or shorter, it can be factored into p, q in a few hours on a laptop
- 512-bit n have been factored using special purpose machines in several weeks time (by really smart individuals)
- Largest known n that has been factored is 768 bits
- Typically RSA n are 1024, 2048, or 4096 bits long. 2048 is generally the minimum bit-length of "modern" moduli n.

Digital Signatures

Digital signatures are a way that we can verify that a message both came from the individual we think it did, and that their message hasn't been modified

How does it work?

- Sender computes the *hash* of their message, and appends it to the end of their message
  - Recall: a hash of a message is a one-way function that produces a number
  - This will provide integrity
- Sender "signs" the message $m|h(m)$ by encrypting with their private key, where $h(m)$ is the hash of m
  - This provides non-repudiation
- The sender then sends this to the recipient
  - The message need not be encrypted using the recipient's public key, but can be if desired.

Digital Signatures Continued

This scheme provides non-repudiation and integrity because...

Non-repudiation

- This is guaranteed because the recipient will decrypt using the sender's public key.
- If it decrypts properly, it could only have come from the sender
- This is because only the sender should possess their private key

Integrity

- The recipient calculates the hash of the message received, and compares it to the hash value the sender appended to the message.
- If they're the same, then the message hasn't been modified
- Integrity is guaranteed, because if the message were modified in transit, then the hash value of the message would no longer match the hash value appended to the message

Suppose you see my public key somewhere on the Internet. . . How do you know that I am who I say that I am?

Questions

Questions?