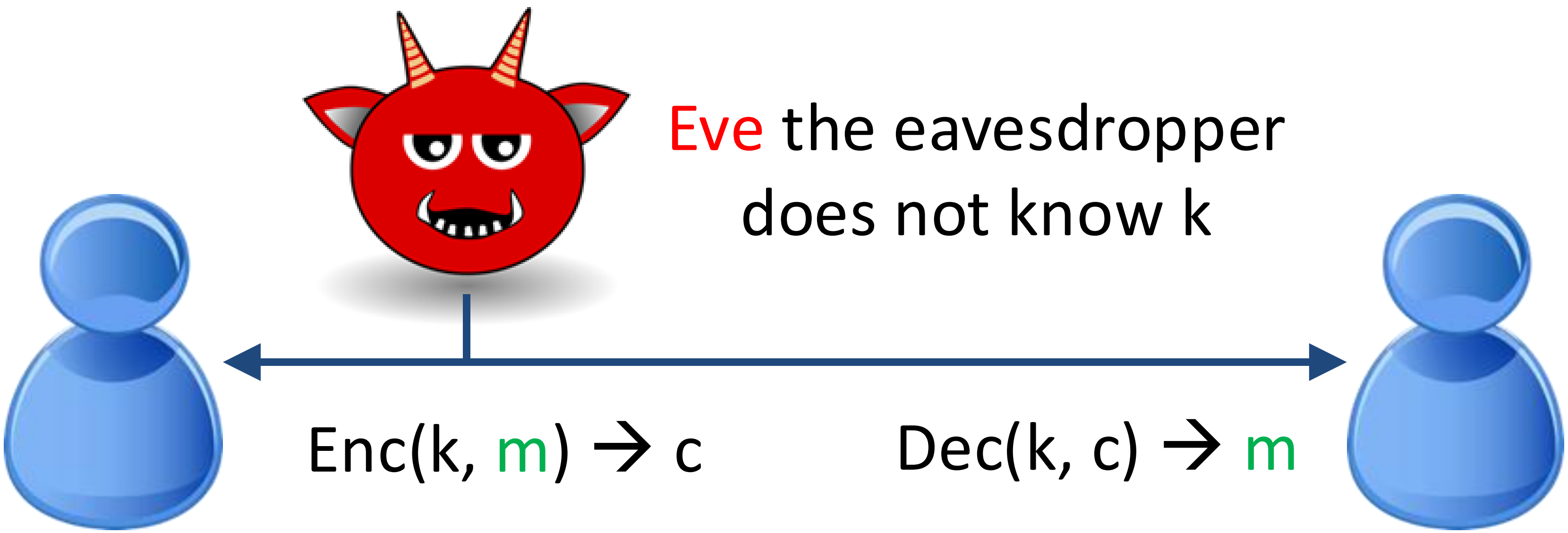
# Chapter 18 – Asymmetric Encryption and Key Exchange

University of Illinois

ECE 422/CS 461

# Recall Symmetric Encryption

- Allows two parties to exchange messages in private (an eavesdropper cannot read)
- Alice and Bob **must** share a secret key




Alice
knows k

m: message or plaintext
c: ciphertext

Bob
knows k

# Motivation for Today

- But how do Alice and Bob share a key?
  - Example scenario: visit a website for the first time?

- Two methods to the rescue!
  - Asymmetric encryption (public-key encryption)
  - Key exchange

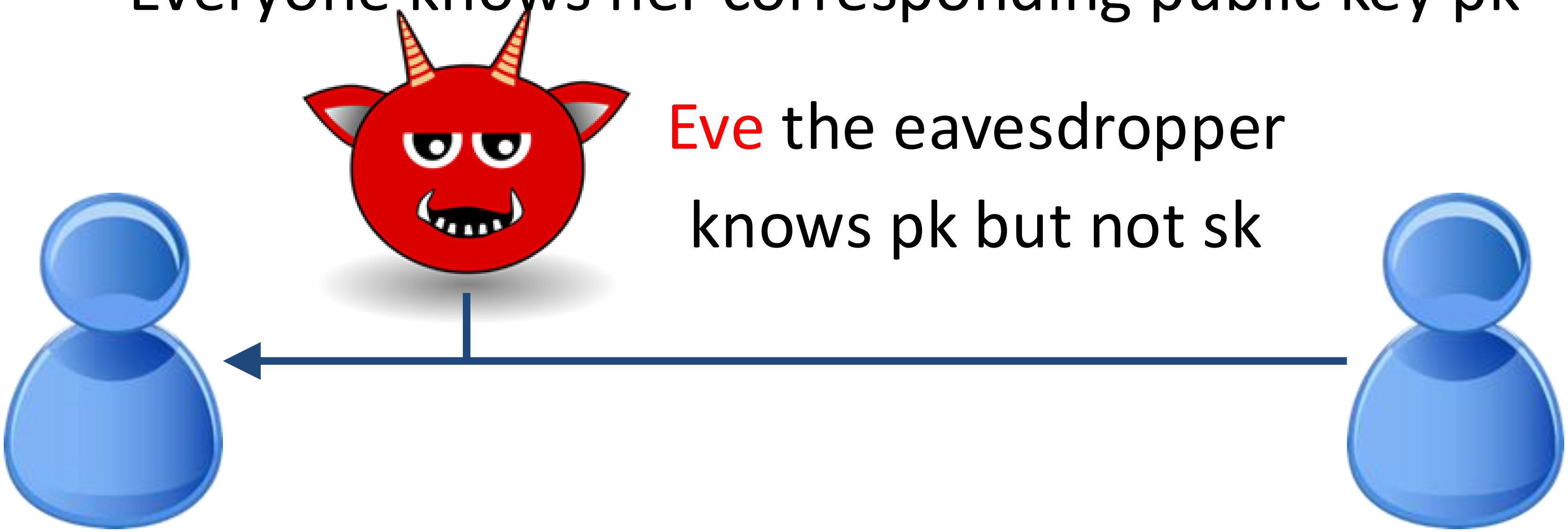
# Goals

- By the end of this chapter you should
  - Know the interfaces and security definitions of asymmetric encryption and key exchange
  - Demonstrate RSA encryption and Diffie-Hellman key exchange on toy examples
  - Understand how they establish symmetric keys
  - Understand man-in-the-middle attacks

# Asymmetric Encryption (Public-Key Encryption)

# Asymmetric Encryption

- Allows **anyone** to send messages to **Alice** in private (an eavesdropper cannot read)
  - Alice has a private key sk
  - Everyone knows her corresponding public key pk



Eve the eavesdropper

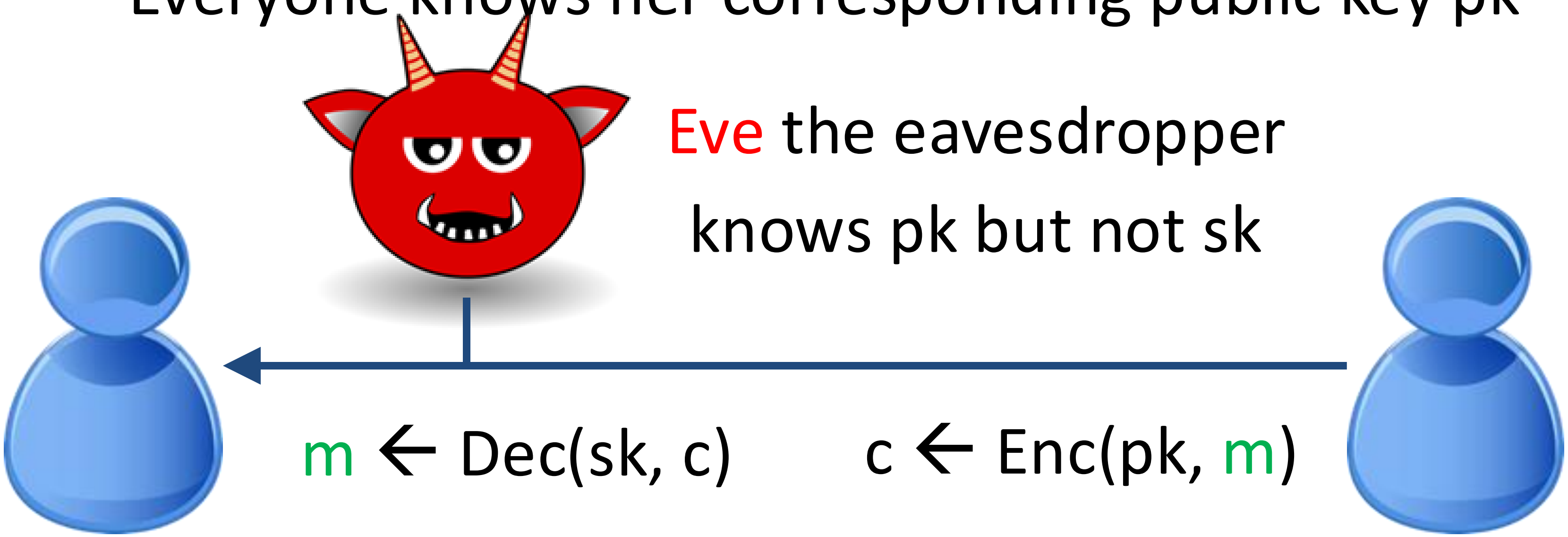
knows pk but not sk

Alice
knows sk

Bob

# Asymmetric Encryption

- Allows **anyone** to send messages to **Alice** in private (an eavesdropper cannot read)
  - Alice has a private key sk
  - Everyone knows her corresponding public key pk



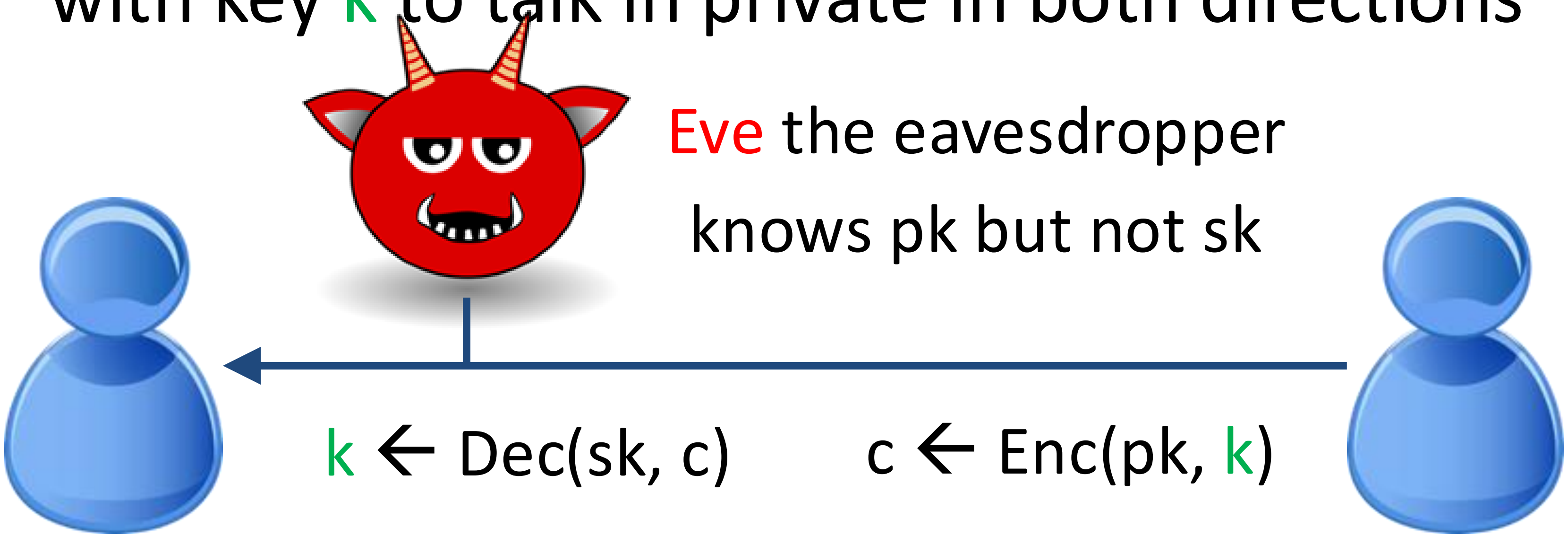

Alice
knows sk

m: message or plaintext
c: ciphertext

Bob

# Hybrid Encryption

- Bob sends a symmetric key k to Alice under asymmetric encryption (Eve cannot read k)
- Alice & Bob can now use symmetric encryption with key k to talk in private in both directions



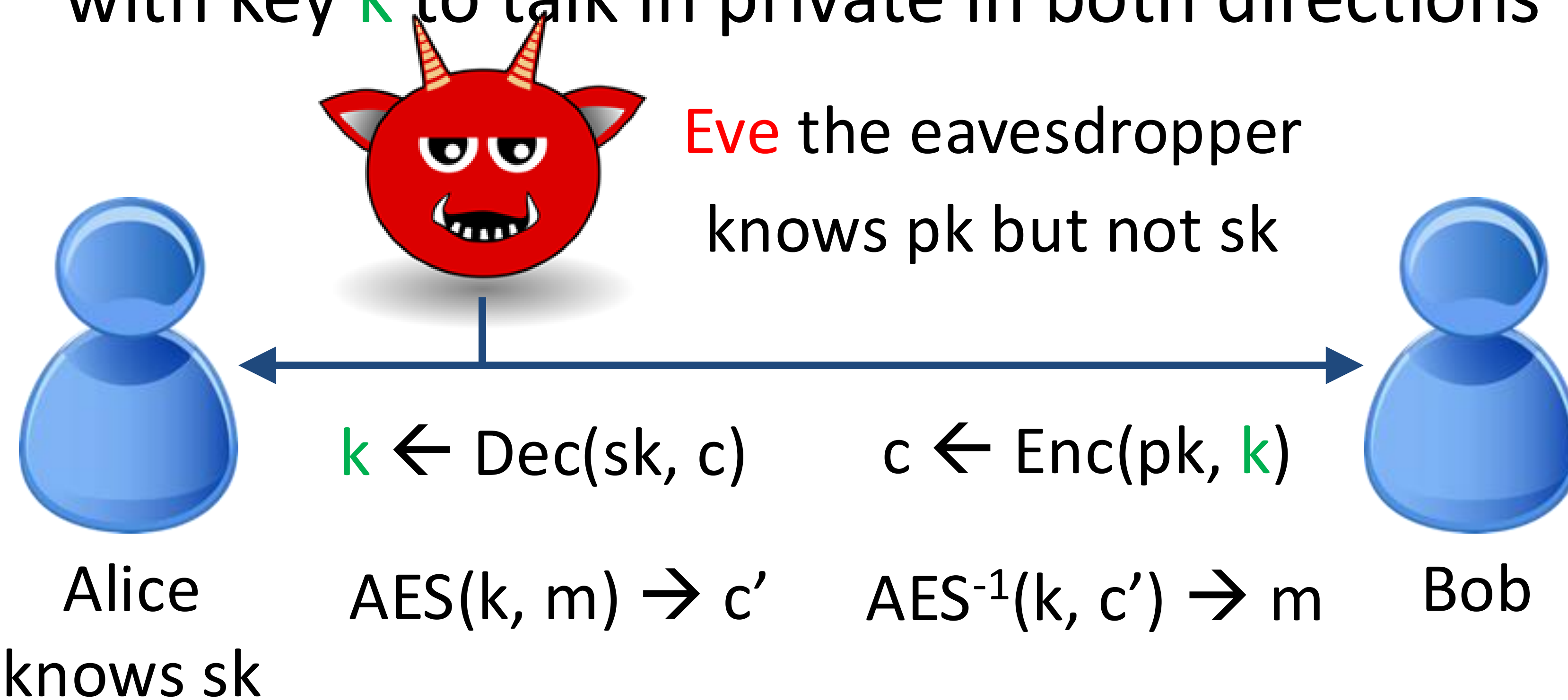

Alice
knows sk

m: message or plaintext
c: ciphertext

Bob

# Hybrid Encryption

- Bob sends a symmetric key k to Alice under asymmetric encryption (Eve cannot read k)
- Alice & Bob can now use symmetric encryption with key k to talk in private in both directions



# Asymmetric Encryption Interface

- KeyGen() → (pk, sk)
  - Publish public key pk and keep private key sk private
  - sk needs to have enough entropy
- Enc(pk, m) → c
  - Everyone can encrypt
- Dec(sk, c) → m
  - Only the key owner can decrypt

# Security: IND-CPA

– We invoke KeyGen() → (pk, sk)
– Eve can ask for encryptions of any messages
– Eve picks two messages $m_0$ and $m_1$ of equal length
– We flip a coin b ← {0, 1} and give Eve Enc(pk, $m_b$)
– Eve can ask for encryptions of any messages
– Eve guesses b. Insecure iff Eve wins with 0.5 + ε probability

# RSA [Rivest-Shamir-Adleman, 1978]

- Most widely used asymmetric encryption
  - Previously invented by Clifford Cocks of British intelligence in 1973; remained classified until 1997


# How (Plain) RSA Works

- Key generation
  - Pick two large (1024-bit) random primes p and q
  - Compute modulus $N = pq$
  - Pick integers e and d such that $ed \equiv 1 \mod (p-1)(q-1)$
  - Public key pk = (e, N). Private key sk = (d, N).

  - Example: $N = 3*11=33$, $(p-1)(q-1) = 20$, $e = 3$, $d = 7$

# How (Plain) RSA Works

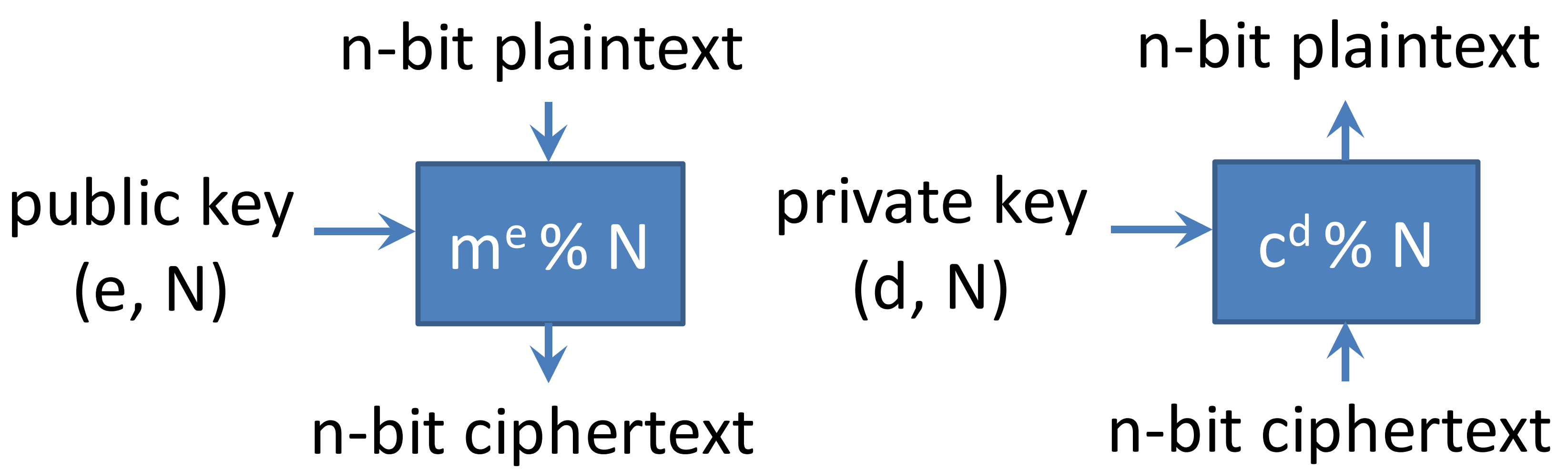
- Key generation
  - $N = pq$, $ed \equiv 1 \bmod (p-1)(q-1)$, $pk = (e, N)$, $sk = (d, N)$

- Encryption: $c = m^e \bmod N$
- Decryption: $m = c^d \bmod N$

- Correctness: $(m^e)^d \bmod N \rightarrow m$
  - Can be proved using number theory

# Security of RSA

- We believe RSA encryption is a pseudorandom permutation (PRP)
  - Why do we believe so?
  - Because no one can break it after >40 years



# Security of RSA

- We believe RSA encryption is a pseudorandom permutation (PRP)
  - Why do we believe so?
  - Because no one can break it after >40 years
- Current best attack on RSA is to factor N = pq
  - Integer factorization has been a hard problem for mathematicians for centuries
  - But no known reduction from RSA to factoring

# RSA

- Still the most used asymmetric encryption today and still believed to be secure

- However, no longer a top recommendation

# RSA No Longer Recommended

- Plain RSA is deterministic, not IND-CPA secure
  - There are ways to make RSA randomized and IND-CPA, but they are complicated and error-prone

- Numerous common pitfalls
  - p, q must be kept private, besides d
  - p, q must be random, insecure to use $pq_1$ and $pq_2$
    - But e need not be random, $e = 2^{16}+1 = 65537$ is good

# RSA No Longer Recommended

- Plain RSA is deterministic, not IND-CPA secure
- Numerous common pitfalls
- Very slow
- Need long keys: > 2048-bit modulus N
  - Most other crypto schemes use 128 or 256 bit keys

# What Are Recommended?

- For asymmetric encryption, use elliptic curve encryption
  - Randomized by default
  - Much faster than RSA
  - Shorter key (160 to 256 bits)

- For establishing a symmetric key, use a *key exchange* protocol (also based on elliptic curves)

# Key Exchange

# Some Basic Number/Group Theory

- Pick a large prime p. {1, 2, 3, …, p-1} form a group G under modular multiplication
  - Multiplication $x \bullet y$ is defined as $x \bullet y \bmod p$
  - (Closure) If x, y are in G, so is $x \bullet y$
  - (Associativity): $(x \bullet y) \bullet z = x \bullet (y \bullet z)$
  - (Identity): there exists e s.t. $e \bullet x = x \bullet e = x$
    - e=1 here
  - (Inverse): For each x, there exists $x^{-1}$ s.t. $x \bullet x^{-1} = e$

# Some Basic Number/Group Theory

- Pick a large prime p. {1, 2, 3, …, p-1} form a group G under modular multiplication
- Can define exponentiation (again, modulo p)
  - $g^a = g \bullet g \bullet g \bullet \ldots \bullet g$ (repeated a times, modulo p)
  - Note that $(g^a)^b = (g^b)^a$
- There are many other groups, e.g., elliptic curve

- We will use a cyclic group G and a generator g
  - You don't need to understand these for CS461

# Important Assumptions

- The discrete-log assumption: given $g$ and $g^a$ for a random $a$, infeasible to find $a$
  - Easy for real numbers (continuous case)

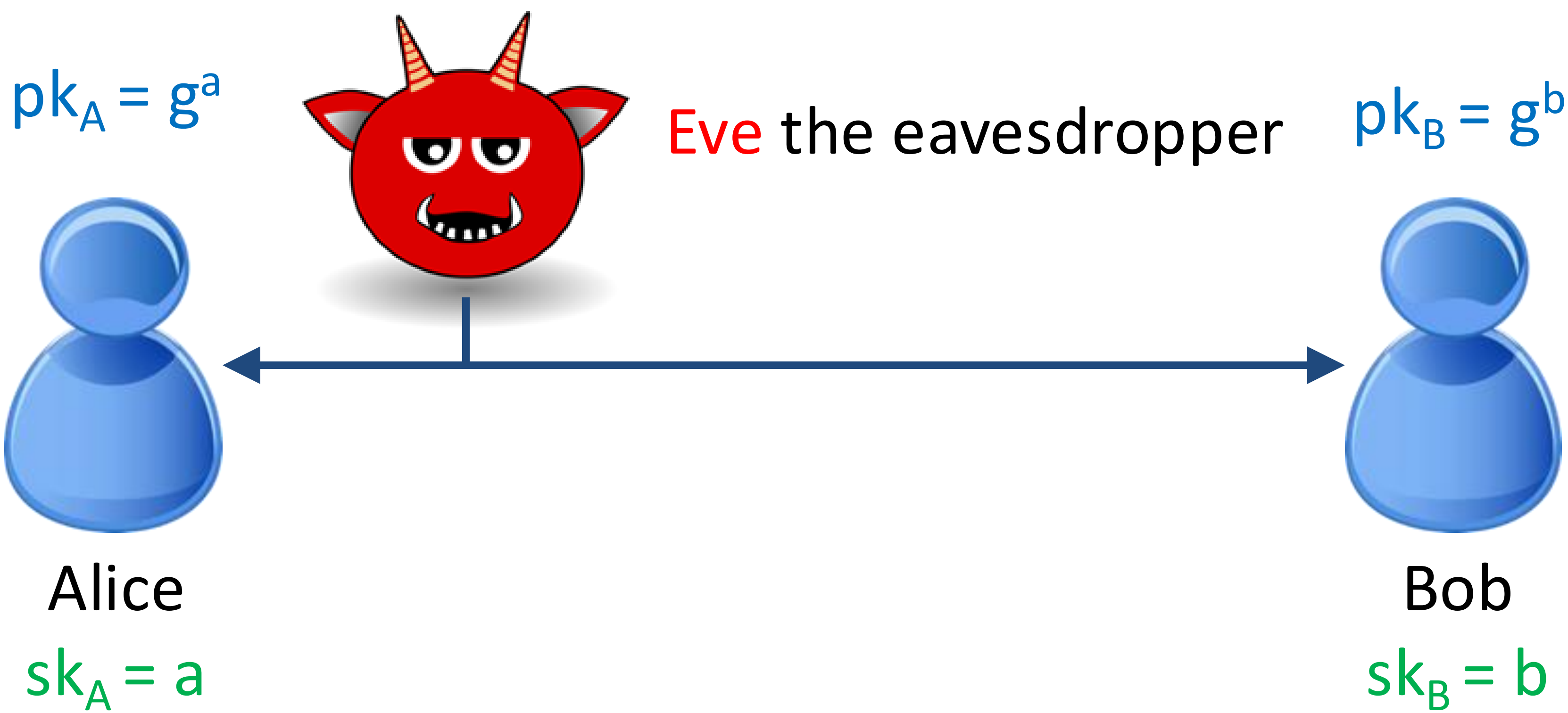
- The Diffie-Hellman assumption: given $g$, $g^a$ and $g^b$, infeasible to find $g^{ab}$
  - Stronger assumption than the discrete-log

# Key Exchange [Diffie-Hellman, 1976]

- "New Directions in Cryptography"
  - Previously invented by Malcolm Williamson of British intelligence in 1969; remained classified until 1997

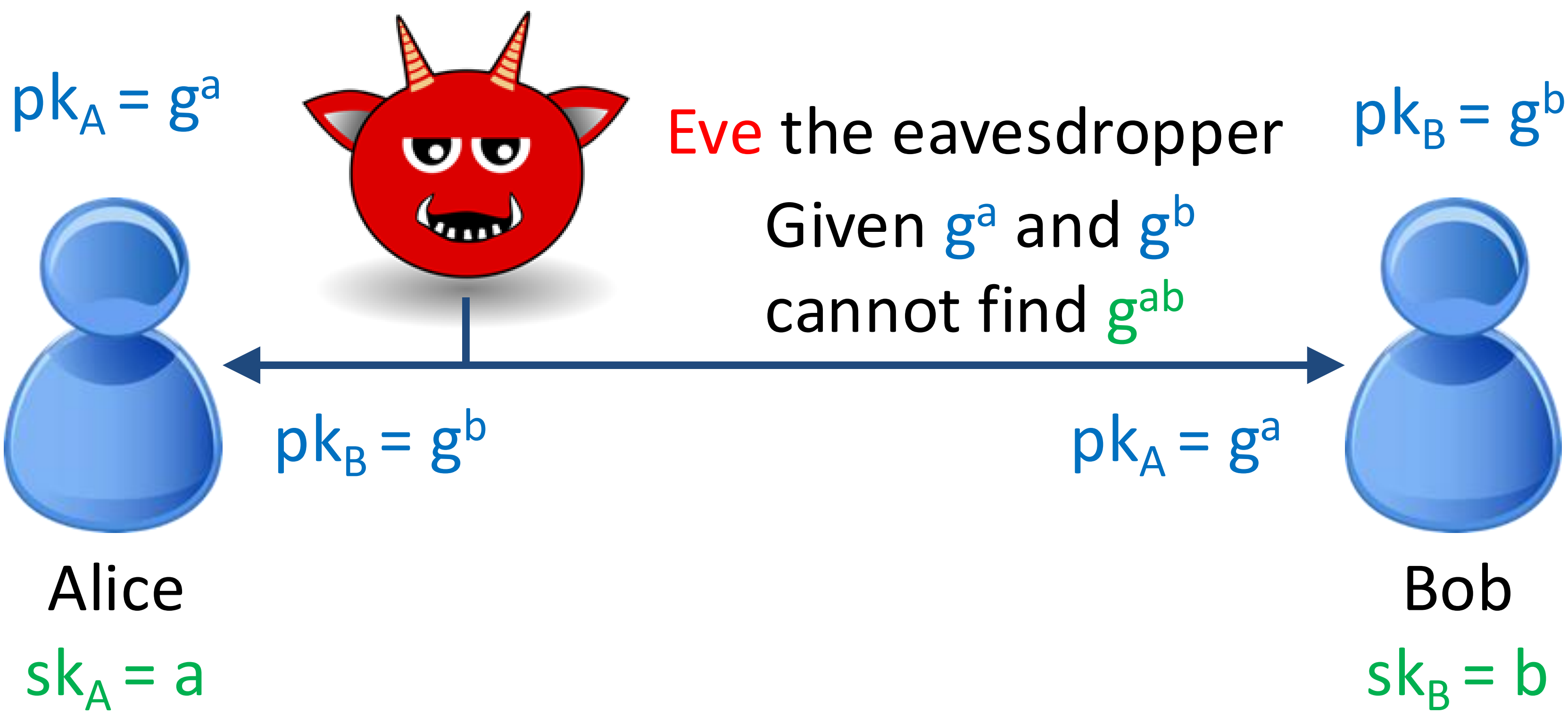

# Diffie-Hellman Key Exchange

- Alice generates a random private key $a$ and the computes her corresponding public key $g^a$
- Bob similarly generates $b$ and $g^b$



# Diffie-Hellman Key Exchange

- Alice and Bob exchange their public keys
- Alice computes $(g^b)^a$ and Bob computes $(g^a)^b$
- Now Alice and Bob share a secret key $g^{ab}$



# Advantages over RSA

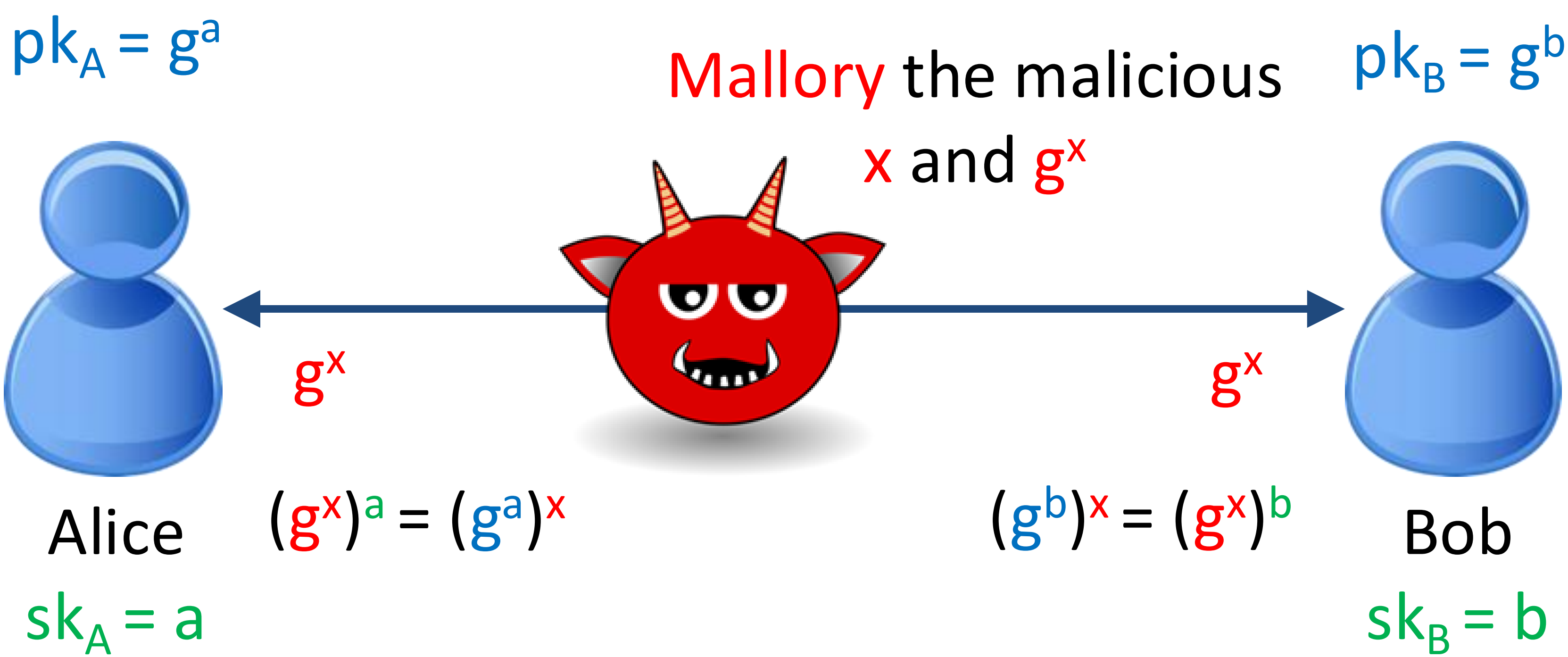
- RSA is slow and needs long (2048-bit) keys
- Diffie-Hellman with elliptic curve group is much faster and has 256-bit keys
  - The version we presented relies on number theory and hence is slow and needs long (2048-bit) keys
- Can provide forward secrecy: if secret keys are stolen, prior communication remains secure
  - May discuss this further in network security

# Man-in-the-Middle (MitM) Attacks

- Asymmetric encryption and key exchange give two ways to establish a symmetric key
- However, both are susceptible to MitM

# Man-in-the-Middle (MitM) Attacks

- Asymmetric encryption and key exchange give two ways to establish a symmetric key
- However, both are susceptible to MitM



# Man-in-the-Middle (MitM) Attacks

- Asymmetric encryption and key exchange give two ways to establish a symmetric key
- However, both are susceptible to MitM
- For them to work against MitM, at least one party's public key needs to be **certified**
  - Next lecture and in network security

# Applications of Encryption

- Confidential communication over Internet
  - e.g., HTTPS, end-to-end encryption
- Confidentiality of external storage
  - Disk encryption, phone encryption
  - Business store user information encrypted “at rest”
- **Key** question: how is the key stored/protected?
  - If derived from password, strength of encryption ← entropy of key ← entropy of password (enough?)
  - If established using Diffie-Hellman, MitM?

# Summary

- Two ways to establish symmetric keys
- Asymmetric encryption:
  - c = Enc(pk, m), m = Dec(sk, c)
  - Security defn is still IND-CPA, randomized
  - Use elliptic curve encryption (RSA OK)
- Diffie-Hellman key exchange:
  - Exchange $g^a$, $g^b$; get shared $g^{ab}$
- Always think about the **key** question when using encryption (and cryptography in general)