# Network Security 2: Cryptography 2

Lecturer: Venkat Arun

# Recap:

- Symmetric encryption uses a common secret key k. Anybody who knows k can encrypt/decrypt messages. To everybody else, the message is unintelligible
- Diffie-Hellman key exchange allows two strangers to use a non-private communication channel to establish a shared secret k
  - While the channel does not have to be private, it should disallow man-in-the-middle (MITM) attacks
  - Most internet paths allow for MITM attacks
- **Today:** Public key encryption allows for a communication channel to be authenticated

# Public key encryption

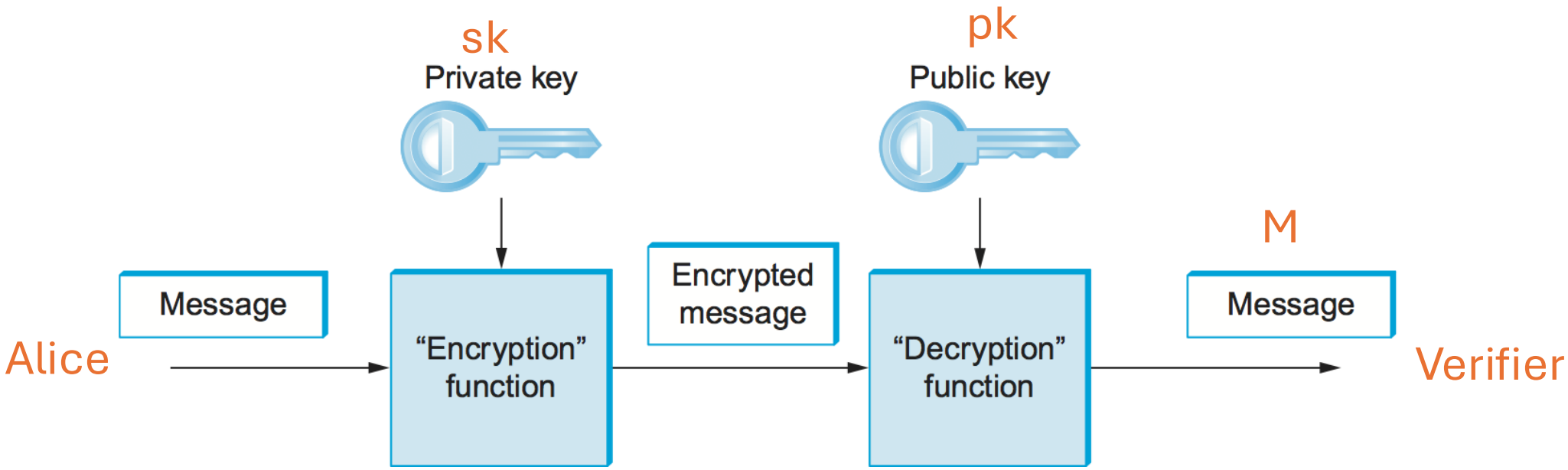
- Alice creates a public/private key pair: (pk, sk)
- She publishes pk to the entire world saying “Hello, you know me as Alice, this is my public key”
- She keeps the private key sk, secret which gives her special abilities
- A computationally limited adversary cannot derive pk from sk, even though it is possible with enough compute

# Case in point

For example, the following is my public key that I use on GitHub and to SSH into things. I have absolutely no qualms about sharing it:

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDsxUpCsXSz96uC68Ex6hM6pOyYliAuH0XVsR/hNAbXk5cK8xvnpe+ono4HfvhlO6Bd4YpsFH7SkH88BRrbSkglLDWI4r0QCq5SoRsJG32Rgo7eB1tzA0Iu2bdGOlifTaI9mudeS8CpQptn+a8PCObDGgrvgKeCIGSU50QJSHyAPUYKsQu3XHfstTVsNIHkSNpwhFTToddGPMZPKNyp4VHAlZygCSGvmTGxADYwLKBunAWYFmP1n5vSVyhU4MJER7QMiw+nQM9UxtxKa9m61NKrK2zElMnQQVKaiXjT4E+ZFMKC6MJwJYk2u46VFTBzYmlxEmBUMhztsU4qYp/OYCtTQE53lfGZJ6DRyf0GIybQIubWQQNcWbBQkfCo+KnDog0nbSQWP8dJvHOStBHFD6XyZzBUgWQ01Aju4I6/HhOzS+sBxJa9pu4DQgemSZkVhP1D6TA2jXU6ZAB/tRatOkS9PvVk8cks3Nszg08ROPQXBkH1/3V3bfTRHxLzSY/Sr7yGxTZQBZ6knfQ7kT84yCSub0KHcwvARm+ux5SyhxP7c9gbMyUj+Vo//E8g/w2npcNefMMRu8UKIeBywCTohEGqKYL1ap0eh45ACnr/icjbIJzZFQz0tXB5dhy2GEotoSkr9iIabvDaNGDTAtZ6PVqUK+7Zwg6lb2pYmh5i99kO1w== venkat@utexas.edu

# Public key encryption: ability #1 (signing)




- Useless for encryption
- By encrypting a message, Alice can "sign" it, since only someone with sk could have produced an encrypted message, that when decrypted with pk, will produce M

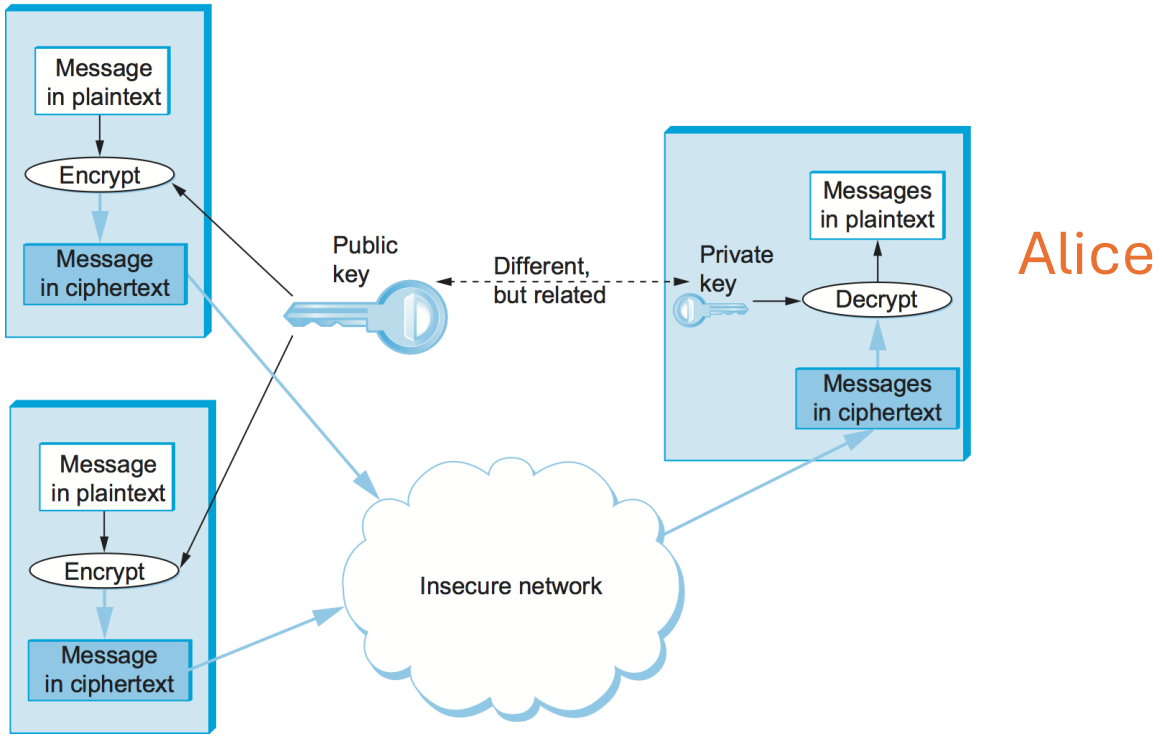
# Example: Use in SSH

- You may have created a public/private key pair for SSHing into machines. This is what is happening.
- You create (pk, sk) and give the machine pk through some trusted means (for example, using your Amazom AWS account which was authenticated using your password)
- The machine picks a large random number c and sends it to you
- You encrypt c using sk to produce s. You send s to the machine
- The machine decrypts s using pk. If the result equals c, it knows that you have sk

# Public key encryption: ability #2 (assymetric encryption)

Stranger on the internet

Stranger on the internet




- Anyone who wants to communicate securely with Alice can encrypt their message m using Alice's pk
- Only Alice will be able to decrypt the resulting ciphertext

# RSA Cryptosystem

The following slides about the RSA cryptosystem are to give you a flavor of how it works.

In this course, you are expected to understand the basic mechanisms, but not the details of modular arithmetic and number theory

# RSA Cryptosystem

- Private key: two large randomly chosen prime numbers p and q
- Public key: n = p * q

Primer on modular arithmetic:

$$x^{\phi(n)} = 1 \; mod \; n$$

where $\phi(n)$ is Euler's totient function

if gcd(x, n) = 1 (i.e. they are "relatively prime")

If n = pq, then $\phi(n) = (p - 1)(q - 1)$

# RSA Cryptosystem

- Private key: two large randomly chosen prime numbers p and q
- Public key: n = p * q

**Assumption:** If n is generated as above, then $\phi(n)$ is hard to compute

**Note:** If you can factor n, you can compute $\phi(n)$, but factoring is not necessary for this

# Encryption using RSA

- Private key: two large randomly chosen prime numbers p and q
- Public key: n = p * q

**Encryption:**

- Pick a message 1 < m < n such that m != p and m != q
- e is a commonly agreed constant.
  - It picked to be coprime with $\phi(n)$ and, for efficiency reasons, small. $e = 2^{16} + 1$ is often used
- Compute $s = m^e \ (mod\ n)$. This is the encrypted message

**Decryption:**

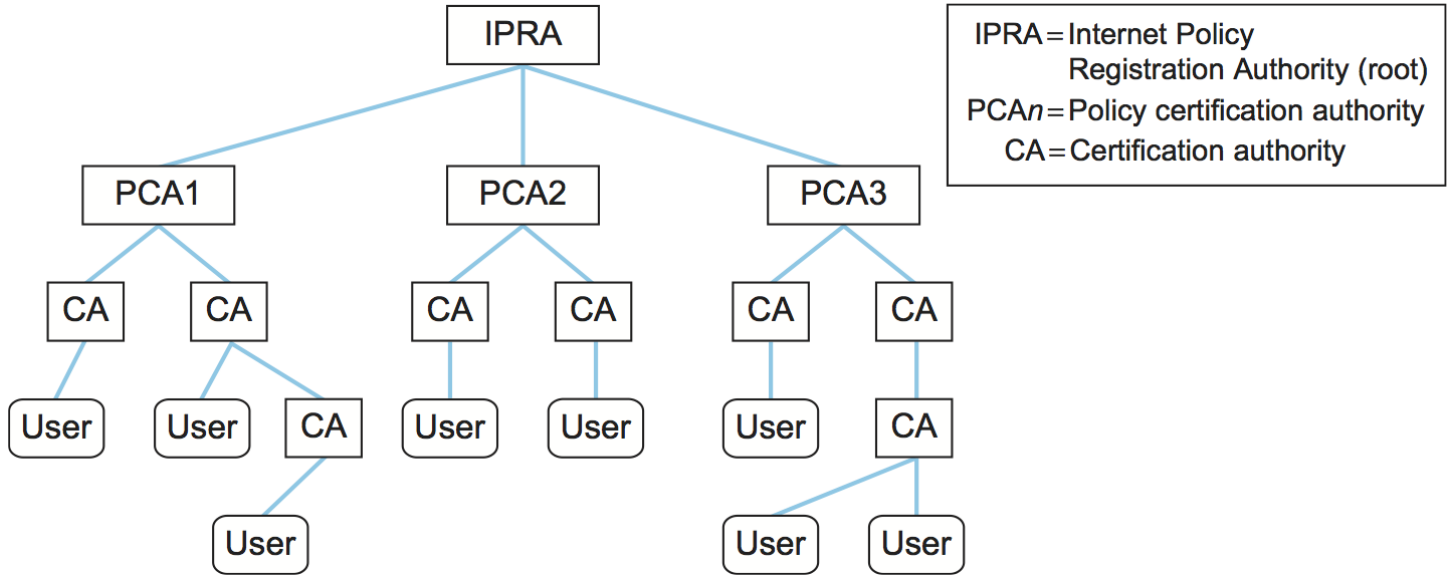
- Pre-compute d such that $d\ e \left(mod\ \phi(n)\right) = 1$
- Decrypt $m' = s^d \ (mod\ n)$

# Key Distribution

The following material is fully a part of this course

# How do I know someone's public key?

- In SSH, you tell the machine your public key using a "trusted mechanism", for example, using a web interface
- How do you secure the web interface?
- The internet has a tree of trusted authorities
- You know the public key of IPRA through "magic". It signs the key for PCA, which signs the public key for the CA, which signs the public key of individual websites



# The common case: TLA, your OS and the browser

- You computer comes pre-installed with an OS
- The OS comes pre-installed with a web browser
- The web browser comes pre-installed with a list of trusted certificate authorities (CA)
- Every time you go to a website using “HTTPS”, use HTTP running over TLS running over TCP
- TLS asks the website for how it would like to certify itself. The website returns its public key, signed by a CA.
- You already know the public key of the CA, so if the signature check passes, you ask the website to authenticate itself with its own private key. If it is able to do so, you trust it
- There are many omitted details here. We will discuss that in the next lecture.

# Next lecture

- Practical considerations in web security
  - Whom and what do we trust?
  - How can we minimize that trust?
  - How do we do encryption fast?
  - How will quantum computers change things?