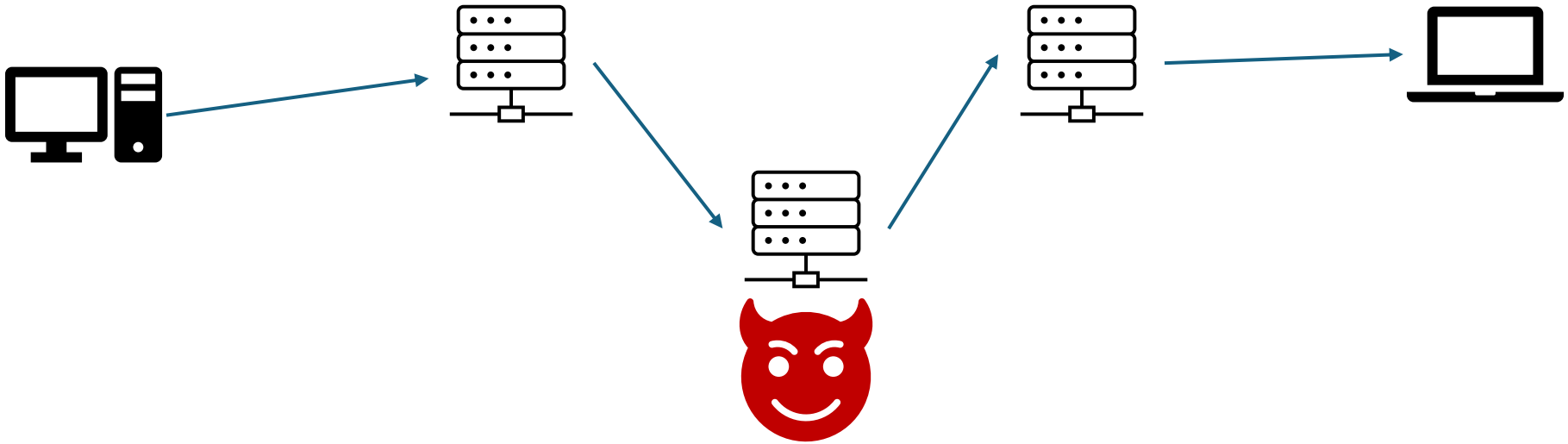
Network Security 1: Cryptography

Symmetric and asymmetric cryptography

Lecturer: Venkat Arun

Chapter 8.2

Why cryptography?



Since packets go through untrusted router, the person owning that router can snoop on your traffic.

This means, when you put your password to login to your bank account for instance, someone on the same WiFi network as you may be able to read your password

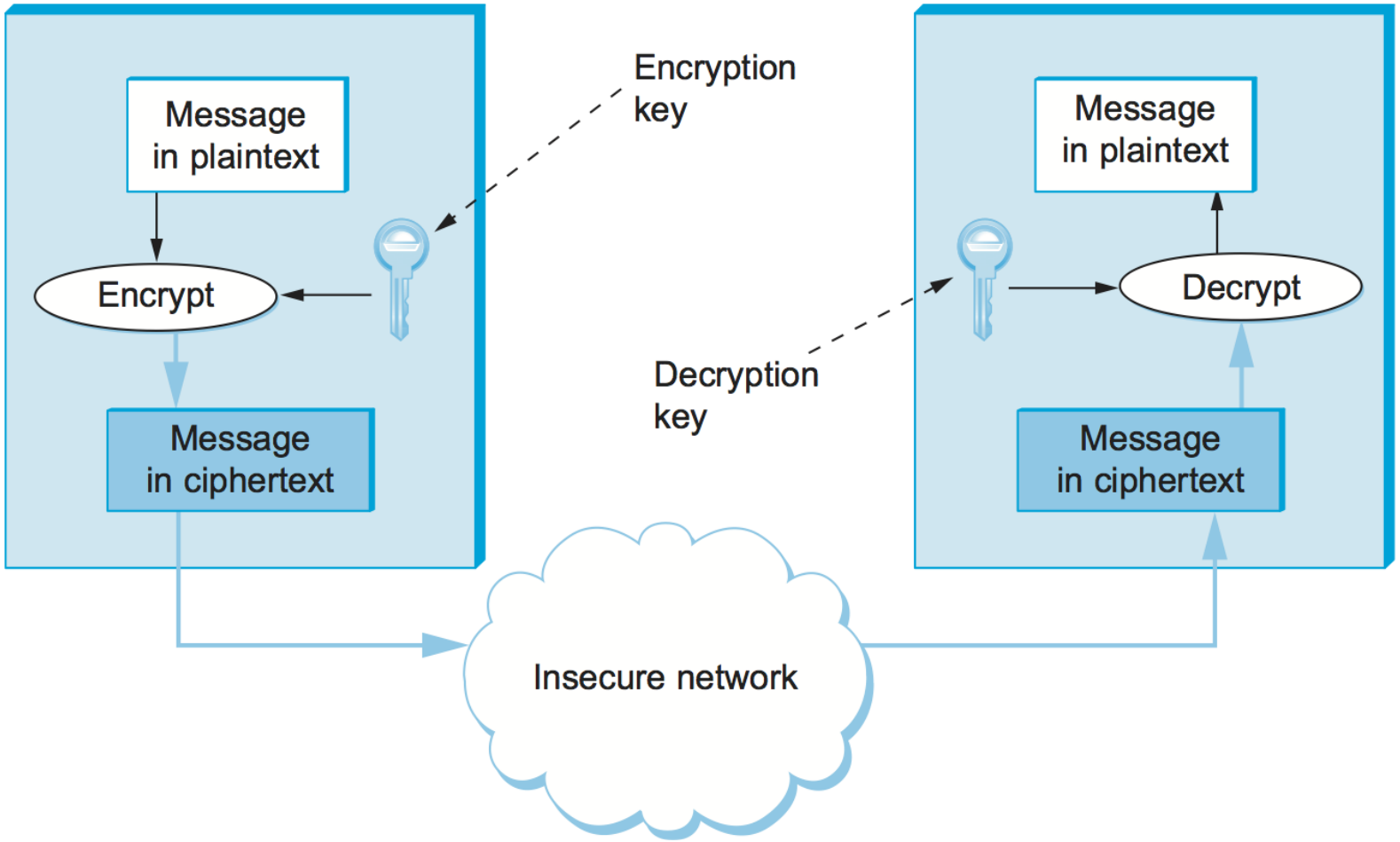
This is prevented by encrypting the password so that even if someone is snooping on all of your communication, they cannot understand what you are saying.

What cryptography does not protect

- It is important to understand what cryptography does and does not protect.
- For instance, we do not encrypt IP addresses since the routers need to know where to send your packets. Encryption also does not hide *how much* data you are sending to each IP address.
- Mechanisms like ToR partially address these concerns. Perfect solutions do not exist.
- **Perfect (almost) solutions do exist to hide *what* is being said, but not *to whom* or *how much***
- We will discuss more cases later

Symmetric encryption

In symmetric encryption, the same key is used for encryption and decryption



Example of symmetric encryption

```
Message:    H  E  L  L  O
            +  +  +  +  +
Key:        9  2  12 8  20
               =
Ciphertext: Q  G  X  T  I
```

- Algorithm: Add the key to the letter. E.g. ‘A’ + 1 = ‘B’
- If the key is as large as the message and each number is randomly chosen from 0-25, this encryption is *impossible to break*, since the probability of a ciphertext being say ‘D’ is entirely independent of what the message was
- This is called the *one-time pad*

You cannot reuse the key in a one-time pad

```
Message:    H  E  L  L  O
            +  +  +  +  +
Key:        9  2  12 8  20
               =
Ciphertext: Q  G  X  T  I
```

```
Message:    W  O  R  L  D
            +  +  +  +  +
Key:        9  2  12 8  20
               =
Ciphertext: F  Q  D  T  X
```

- If you re-use the same key for encrypting two messages, it is possible to crack the code. For instance, the fact that the fourth letter in both ciphertexts is the same tells us that the fourth letter in the two messages was also the same.
- We also know the difference between every pair of letters in the two messages. For instance, we know the difference between the first letters in the two messages is ‘Q’ – ‘F’ = 11
- Since we know the pattern of letters in English, we can decrypt the messages

Symmetric encryption that reuses keys

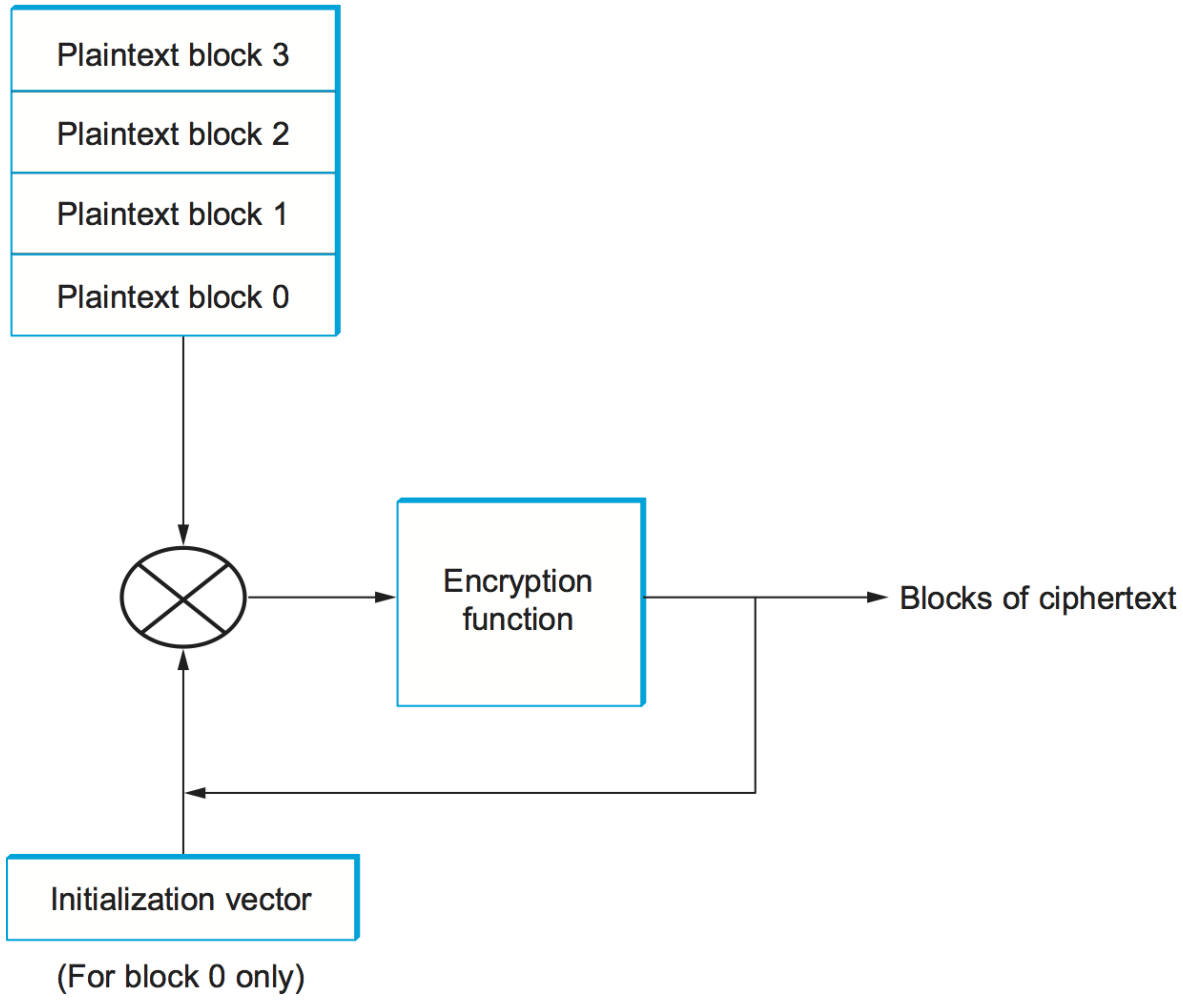
- Fundamentally, it is impossible to prevent decryption (by an adversary with a computer that can run exponential time algorithms) if the key is shorter than the message
- However, having very large keys is impractical. I'd prefer to have a small key or password and be able to encrypt large messages with it
- Symmetric encryption mechanisms that allow for this exist
- They work by using more complicated scrambling techniques than a simple addition per letter
- They guard against decryption by assuming that the adversary does not have a sufficiently powerful computer. If they did, encryption is not possible

AES: The standard symmetric encryption algorithm

- Here is an animation of how it works: https://legacy.cryptool.org/en/cto/aes-animation

- You just need to understand that it is trying to scramble the ciphertext with the key as much as possible. AES provides few theoretical guarantees. A few things to note:
  - All operations are invertible if you know the key. That is how decryption works
  - AddSubBytes is the only non-linear operation. The other 3 are all linear (affine, to be precise). Linear operations serve to mix the bytes. Non-linearity ensures that the nine rounds don't collapse into one effective round that would be easy to break (ignore this sentence if you have not taken linear algebra)
- The assumption is that to decrypt a message encrypted with a 128-bit key, the adversary will need $O(2^{128})$ operations to decode it.

Block-wise encryption (see 8.2.1)



A somewhat formal definitions of security

We believe the following to be true for AES

- Suppose Alice has her symmetric key k. Bob is allowed to ask her to encrypt any set of messages m_1, m_2, m_3 … he likes and tell him $E_k(m_1), E_k(m_2), \ldots$
- Then he asks her to encrypt messages X and Y
- Alice returns either $E_k(X), E_k(Y)$ or $E_k(Y), E_k(X)$ with 50% probability each
- If E is a good algorithm, Bob cannot figure in out which order she returned it to him any better than randomly guessing
- This is an extremely strong guarantee/assumption of security

How do we distribute public keys?

Limitations of symmetric encryption

- German Enigma machines were an example of symmetric encryption
- During the WWII, a codebook was distributed every month to all the people receiving encoded messages
- The codebook will specify what key to use each day
- If the codebook got leaked, *anyone* could decrypt the messages
- This sometimes happened

The Diffie Hellman Key exchange

The Diffie Hellman key exchange offers a magical ability

Two strangers can stand in a crowded room and talk to each other loudly so anyone who cares can overhear

Yet, nobody else can understand what these two are saying

Key assumption about discrete logarithms

AES just handwaves to say that it is secure. The DH key exchange in contrast is based on a more precisely stated assumption:

Let p be a large, publicly known, prime number

Let g be an integer picked uniformly at random from the range $2, p$

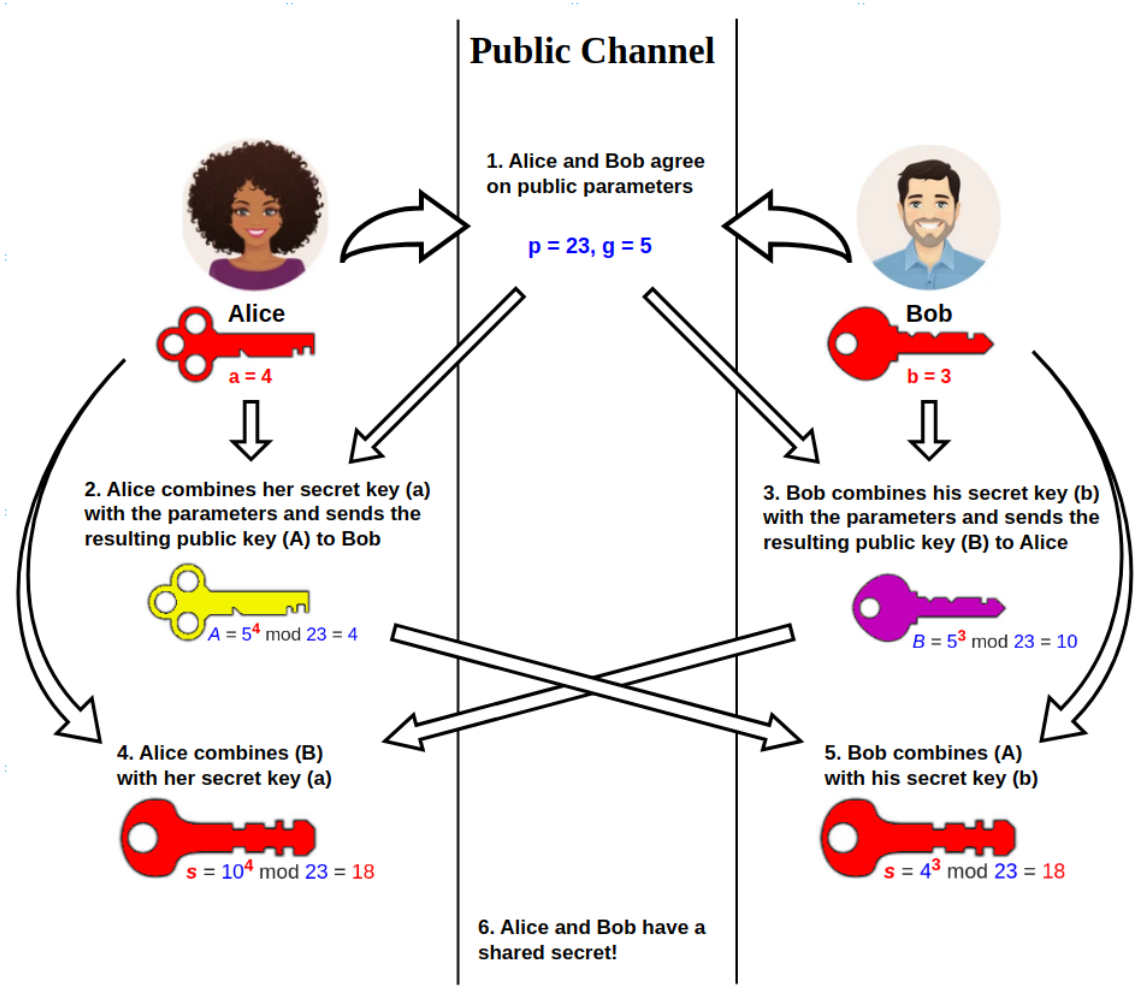
Given $h = g^x \ mod \ p$

It is not possible to recover x even if g and p are known. Note: a unique m satisfying this equation exists, but cannot be found in polynomial time

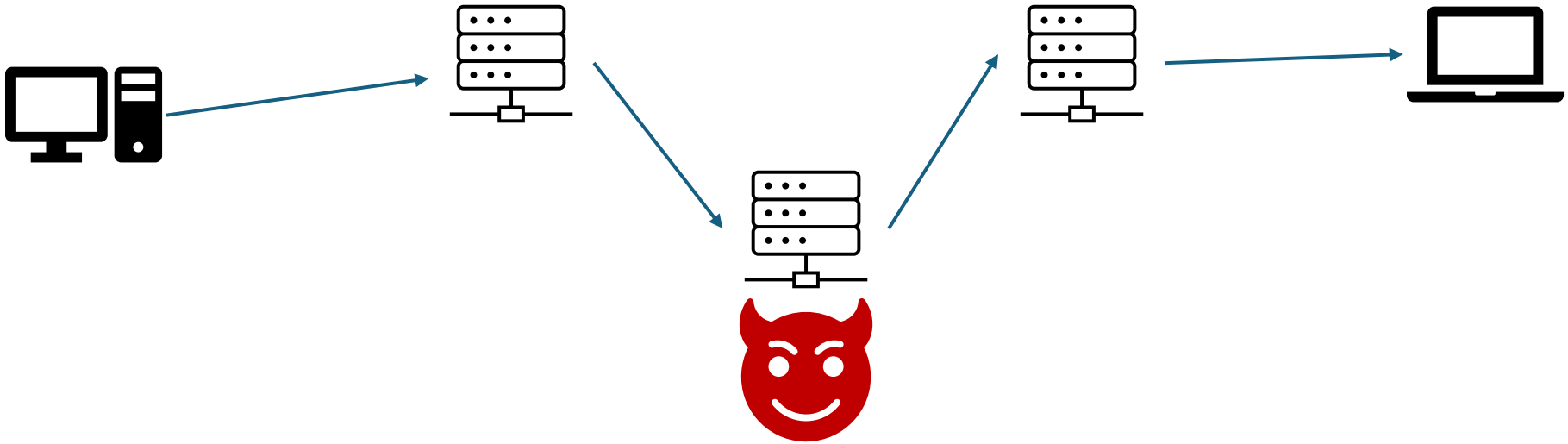
Stronger assumption: Given, $h_1 = g^x \ mod \ p$ and $h_2 = g^y mod \ p$, it is not possible to compute g^{xy} in polynomial time

The DH key exchange

- Input: nothing. They just need to agree on public parameters p and g
- Output: a shared key known only to Alice and Bob and not to any eavesdroppers



Man in the middle attack (MITM)



Suppose, instead of just eavesdropping on the conversation, the adversary can intercept, modify or even send entirely new packets. Can they get the plaintext of the two conversing parties even if they exchanged keys using the DH exchange? How?