رمزنگاری، امنیت اطلاعات و حریم خصوصی

ارائه: دکتر سیدعلی لاجوردی

بخش دوم

So far…

- Heuristic constructions; build, break, repeat, …
  - This isn't very satisfying

- Can we prove that some encryption scheme is secure?

- First need to define what we mean by "secure" in the first place…

Modern cryptography

- Historically, cryptography was an art
  - Heuristic design and analysis

- Starting in the early ‘80s, cryptography began to develop into more of a science

- Based on three principles that underpin most real-world cryptography today

Core principles of modern crypto

- Formal definitions
  - Precise, mathematical model and definition of what security means

- Assumptions
  - Clearly stated and unambiguous

- Proofs of security
  - Move away from design-break-patch cycle

Importance of definitions

- Definitions are essential for the design, analysis, and sound usage of crypto
- Developing a precise definition forces the designer to think about what they really want
  - What is essential and (sometimes more important) what is not
- If you don’t understand what you want to achieve, how can you possibly know when (or if) you have achieved it?
- Definitions enable meaningful analysis, evaluation, and comparison of schemes
  - Does a scheme satisfy the definition?
  - What definition does it satisfy?
- Definitions allow others to understand the security guarantees provided by a scheme
- Enables schemes to be used as components of a larger system (modularity)
- Enables one scheme to be substituted for another if they satisfy the same definition

Assumptions

- With few exceptions, cryptography currently requires computational assumptions
  - At least until we prove $P \neq NP$ (and even that would not be enough)

- Principle: any such assumptions must be made explicit

Importance of clear assumptions

- Allow researchers to (attempt to) validate assumptions by studying them
- Allow meaningful comparison between schemes based on different assumptions
  - Useful to understand minimal assumptions needed
- Practical implications if assumptions are wrong

- Enable proofs of security

Proofs of security

- Provide a rigorous proof that a construction satisfies a given definition under certain specified assumptions
  - Provides an iron-clad guarantee (relative to your definition and assumptions!)

- Proofs are crucial in cryptography, where there is a malicious attacker trying to “break” the scheme

Limitations?

- Cryptography still remains partly an art as well

- Proofs given an iron-clad guarantee of security
  - …relative to the definition and assumptions!

- Provably secure schemes can be broken!
  - If the definition does not correspond to the real-world threat model
  - If the assumption is invalid
  - If the implementation is flawed
- This does not detract from the importance of having formal definitions in place and giving proofs of security

Defining secure encryption

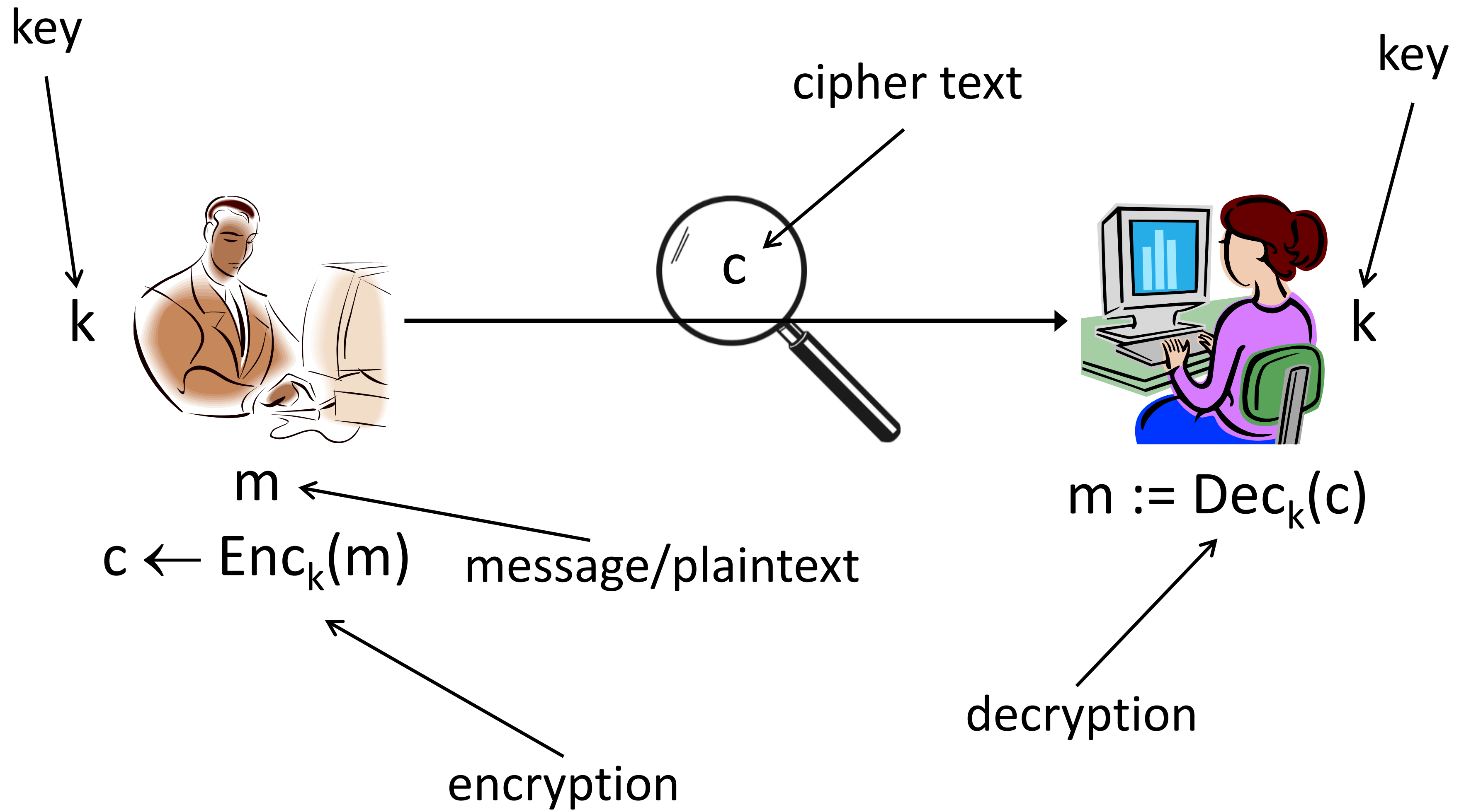
Crypto definitions (generally)

- Security guarantee/goal
  - What we want to achieve (or what we want to prevent the attacker from achieving)
- Threat model
  - What (real-world) capabilities the attacker is assumed to have

Recall

- A private-key encryption scheme is defined by a message space M and algorithms (Gen, Enc, Dec):
  - Gen (key-generation algorithm): generates k
  - Enc (encryption algorithm): takes key k and message $m \in M$ as input; outputs ciphertext c
    $$c \leftarrow Enck(m)$$
  - Dec (decryption algorithm): takes key k and ciphertext c as input; outputs m.
    $$m := Deck(c)$$

Private-key encryption



Threat models for encryption

- Ciphertext-only attack
  - One ciphertext or many?
- Known-plaintext attack
- Chosen-plaintext attack
- Chosen-ciphertext attack

Goal of secure encryption?

- How would you define what it means for encryption scheme (Gen, Enc, Dec) over message space M to be secure?
  - Against a (single) ciphertext-only attack

Secure encryption?

- “Impossible for the attacker to learn the key”
  - The key is a means to an end, not the end itself
  - Necessary (to some extent) but not sufficient
  - Easy to design an encryption scheme that hides the key completely, but is insecure
  - Can design schemes where most of the key is leaked, but the scheme is still secure
- “Impossible for the attacker to learn the plaintext from the ciphertext”
  - What if the attacker learns 90% of the plaintext?
- “Impossible for the attacker to learn any character of the plaintext from the ciphertext”
  - What if the attacker is able to learn (other) partial information about the plaintext?
  - What if the attacker guesses a character correctly, or happens to know it?

The right definition

- “Regardless of any prior information the attacker has about the plaintext, the ciphertext should leak no additional information about the plaintext”
  - How to formalize?

Perfect secrecy

Perfect secrecy

- “Regardless of any prior information the attacker has about the plaintext, the ciphertext should leak no additional information about the plaintext”
- Attacker’s information about the plaintext = attacker knows the distribution of M
- Perfect secrecy: observing the ciphertext should not change the attacker’s knowledge about the distribution of M
- Encryption scheme (Gen, Enc, Dec) with message space M and ciphertext space C is perfectly secret if for every distribution over M, every $m \in M$, and every $c \in C$ with $\Pr[C=c] > 0$, it holds that

$$\Pr[M = m \mid C = c] = \Pr[M = m]$$

One-time pad

- Patented in 1917 by Vernam
  - Recent historical research indicates it was invented (at least) 35 years earlier

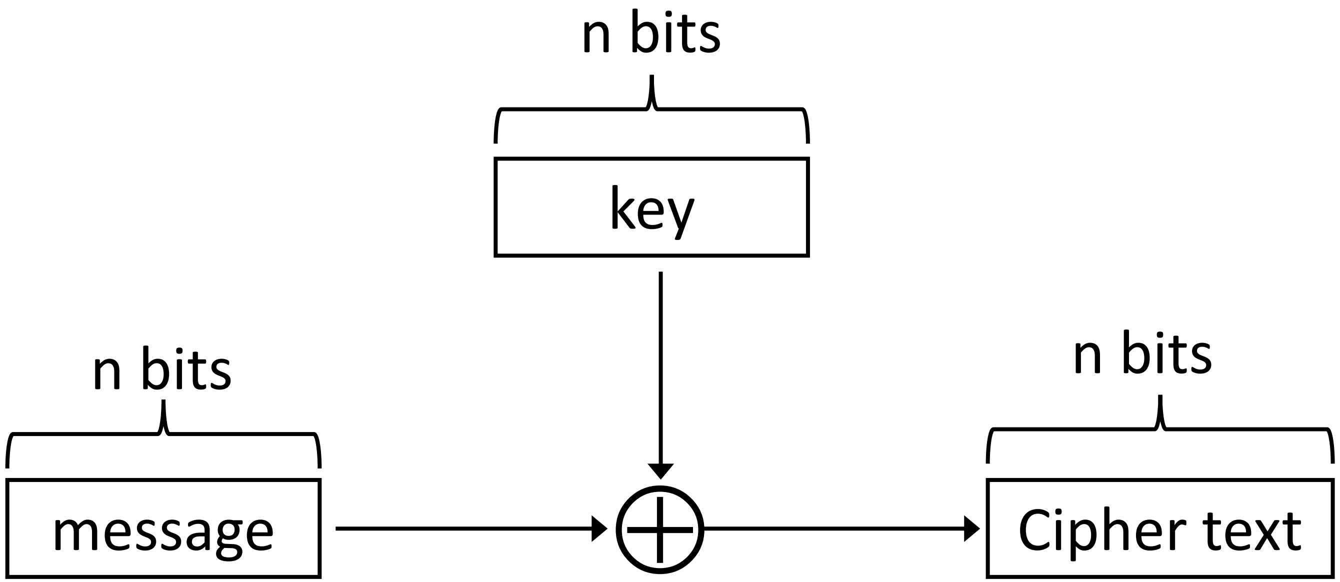
- Proven perfectly secret by Shannon (1949)

One-time pad (OTP)

- Let $\mathcal{M} = \{0,1\}^n$
- Gen: choose a uniform key $k \in \{0,1\}^n$
- $Enc_k(m) = k \oplus m$
- $Dec_k(c) = k \oplus c$

- Correctness:
  Deck(Enck(m)) = $k \oplus (k \oplus m)$
  $= (k \oplus k) \oplus m = m$

One-time pad



Perfect secrecy of one-time pad

- Note that any observed ciphertext can correspond to any message
- So, having observed a ciphertext, the attacker cannot conclude for certain which message was sent
- Theorem: The one-time pad is perfectly secret

One-time pad

- Several limitations
  - The key is as long as the message
  - Only secure if each key is used to encrypt a *single* message

$\Rightarrow$ Parties must share keys of (total) length equal to the (total) length of all the messages they might ever send

Using the same key twice?

- Completely insecure against a known-plaintext attack!
- Say c_1 (= $k \oplus m_1$)
  c_2 (= $k \oplus m_2$)
  and the attacker knows m_1
- Attacker can compute $k := c_1 \oplus m_1$
- Attacker can compute $m_2 := c_2 \oplus k$

Using the same key twice?

- Say $c_1 = k \oplus m_1$
  $c_2 = k \oplus m_2$

- Attacker can compute
  $c_1 \oplus c_2 = (k \oplus m_1) \oplus (k \oplus m_2) = m_1 \oplus m_2$

- This leaks information about m_1, m_2!

Using the same key twice?

- $m_1 \oplus m_2$ is information about m_1, m_2

- Is this significant?
  - No longer perfectly secret!
  - $m_1 \oplus m_2$ reveals where m_1, m_2 differ
  - Frequency analysis
  - Exploiting characteristics of ASCII…

One-time pad

- Drawbacks
  - Key as long the message
  - Only secure if each key is used to encrypt *once*
  - Trivially broken by a known-plaintext attack

- All these limitations are *inherent* for schemes achieving perfect secrecy
  - I.e., it’s not just a problem with the OTP

Optimality of the one-time pad

- Theorem: if (Gen, Enc, Dec) with message space $\mathcal{M}$ is perfectly secret, then $|\mathcal{K}| \geq |\mathcal{M}|$
- Intuition:
  - Given any ciphertext, try decrypting under every possible key in $\mathcal{K}$
  - This gives a list of up to $|\mathcal{K}|$ possible messages
  - If $|\mathcal{K}| < |\mathcal{M}|$, some message is not on the list
- Proof:
  - Assume $|\mathcal{K}| < |\mathcal{M}|$
  - Need to show that there is a distribution on $\mathcal{M}$, a message m, and a ciphertext c such that

$$\Pr[M=m \mid C=c] \neq \Pr[M=m]$$

Where do we stand?

- We defined the notion of perfect secrecy
- We proved that the one-time pad achieves it!
- We proved that the one-time pad is optimal!
  - E.g., we cannot improve the key length
- Are we done?

- Do better *by relaxing the definition*
  - But in a meaningful way…

Perfect secrecy

- Requires that *absolutely no information* about the plaintext is leaked, even to eavesdroppers *with unlimited computational power*
  - The definition has some inherent drawbacks
  - The definition seems unnecessarily strong…

Computational secrecy

- Would be ok if a scheme leaked information *with tiny probability* to eavesdroppers *with bounded computing resources/running time*
- I.e., we can relax perfect secrecy by
  - Allowing security to “fail” with tiny probability
  - Restricting attention to “efficient” attackers

Tiny probability of failure?

- Say security fails with probability 2^{-60}
  - Should we be concerned about this?
  - With probability > 2^{-60}, the sender and receiver will both be struck by lightning in the next year…
  - Something that occurs with probability 2^{-60}/sec is expected to occur once every 100 billion years

Bounded attackers?

- Consider brute-force search of key space; assume one key can be tested per clock cycle
- Desktop computer $\approx 2^{57}$ keys/year
- Supercomputer $\approx 10^{17}$ flops $\approx 2^{80}$ keys/year
- Supercomputer since Big Bang $\approx 2^{112}$ keys
  - Restricting attention to attackers limited to trying 2^{112} keys is fine!

- Modern key spaces: 2^{128} keys or more…

Roadmap

- We will give an alternate (but equivalent) definition of perfect secrecy
  - Using a randomized experiment
- That definition has a natural relaxation

Perfect indistinguishability

- Π = (Gen, Enc, Dec), message space $\mathcal{M}$
- Informally:
  - Two messages m_0, m_1; one is chosen and encrypted (using unknown k) to give $c \leftarrow \text{Enc}_k(m_b)$
  - Adversary A is given c and tries to determine which message was encrypted
  - Π is perfectly indistinguishable if *no* A can guess correctly with probability *any better than ½*

Perfect indistinguishability

- Claim: Π is perfectly indistinguishable if and only if Π is perfectly secret
  - I.e., perfect indistinguishability is just an alternate definition of perfect secrecy

Encryption and plaintext length

- In practice, we want encryption schemes that can encrypt arbitrary-length messages
- Encryption does not hide the plaintext length (in general)
  - The definition takes this into account by requiring m_0, m_1 to have the same length
- But beware that leaking plaintext length can often lead to problems in the real world!
  - Obvious examples…
  - Database searches
  - Encrypting compressed data

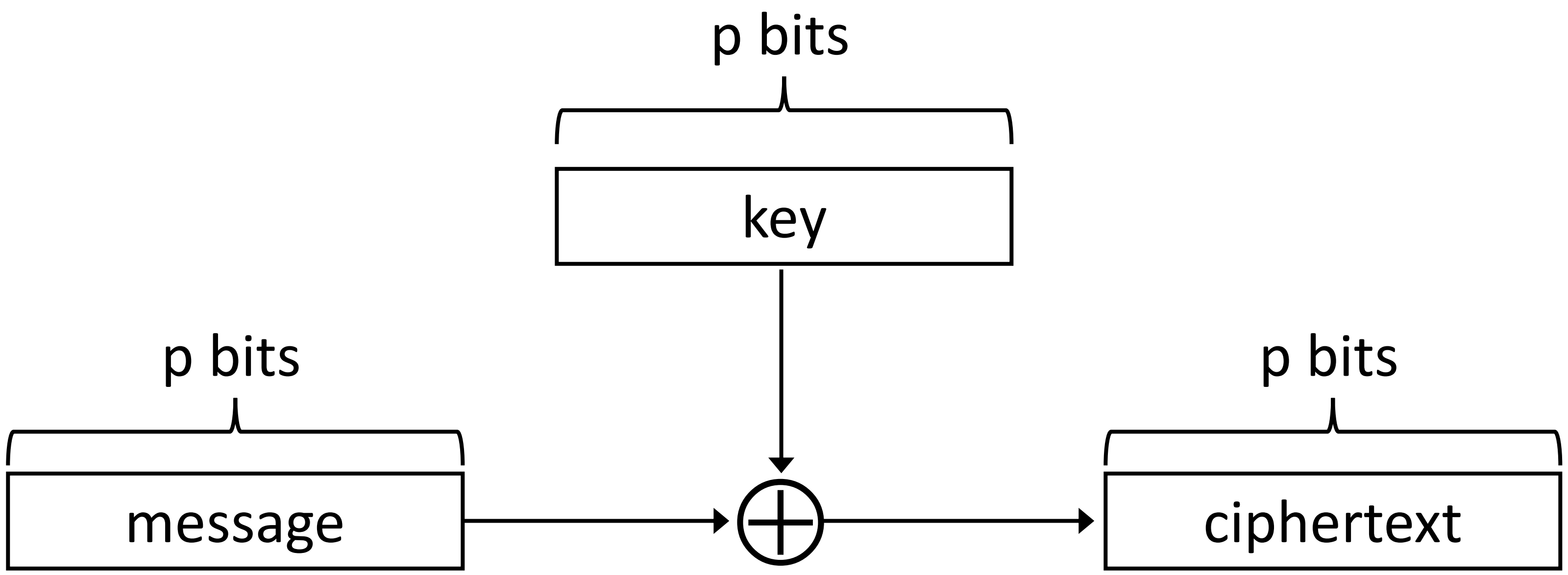
Where things stand

- We saw that there are some inherent limitations if we want perfect secrecy
  - In particular, key must be as long as the message

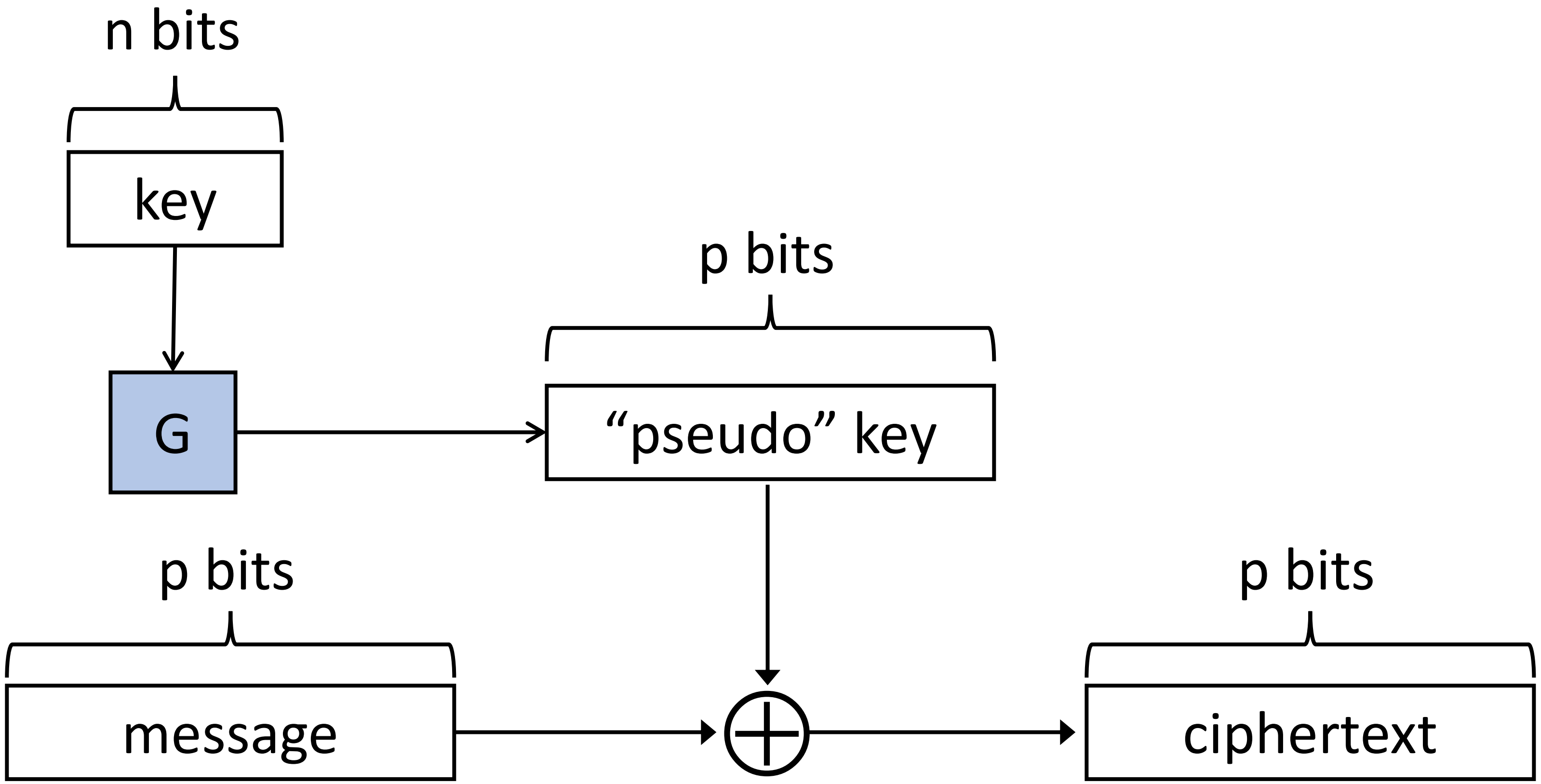
- We defined computational secrecy, a relaxed notion of security

- Does that definition allow us to overcome prior limitations?

Recall: one-time pad



“Pseudo” one-time pad



Pseudo one-time pad

- Let G be a deterministic algorithm, with $|G(k)| = p(|k|)$
- $Gen(1^n)$: output uniform n-bit key k
  - Security parameter n $\Rightarrow$ message space $\{0,1\}^{p(n)}$
- $Enc_k(m)$: output $G(k) \oplus m$
- $Dec_k(c)$: output $G(k) \oplus c$

- Correctness follows as in the OTP…

Have we gained anything?

- YES: the pseudo-OTP has a key shorter than the message
  - n bits vs. p(n) bits
- The fact that the parties *internally* generate a p(n)-bit temporary string to encrypt/decrypt is **irrelevant**
  - The *key* is what the parties share *in advance*
  - Parties do not store the p(n)-bit temporary value
- Security of pseudo-OTP?

Definitions, proofs, and assumptions

- We've *defined* computational secrecy
- Our goal is to *prove* that the pseudo-OTP meets that definition
- We cannot prove this unconditionally
  - Beyond our current technique
  - Anyway, security clearly depends on G
- *Can* prove security based on *the assumption* that G is a pseudorandom generator

Proof by reduction

1. Assume G is a pseudorandom generator
2. Assume toward a contradiction that there is an efficient attacker A who "breaks" the pseudo-OTP scheme (as per the definition)
3. Use A as a subroutine to build an efficient D that "breaks" pseudorandomness of G
   - By assumption, no such D exists!

   $\Rightarrow$No such A can exist

❖If G is a pseudorandom generator, then the pseudo one-time pad Π is EAV-secure (i.e., computationally indistinguishable)

Keyed functions

- Let $F: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ be an efficient, deterministic algorithm
  - Define $F_k(x) = F(k, x)$
  - The first input is called the *key*
  - Security parameter = key length = n

- F is *pseudorandom* if F_k (for uniform k) is indistinguishable from a random function on the same domain/range

Block ciphers

- Block ciphers are practical constructions of pseudorandom permutations

- No asymptotics: F: $\{0,1\}^n$ x $\{0,1\}^m \rightarrow \{0,1\}^m$ for fixed n, m
  - n = “key length”
  - m = “block length”

- Hard to distinguish F_k from uniform $f \in \mathrm{Perm}_m$ *even for attackers running in time* $\approx 2^n$

AES

- Advanced encryption standard (AES)
  - Key length = 128, 192, or 256 bits
  - Block length = 128 bits

- Will discuss details later in the course

- Available in standard crypto libraries
- No real reason to use anything else

Message integrity

Secrecy vs. integrity

- So far we have been concerned with ensuring *secrecy* of communication

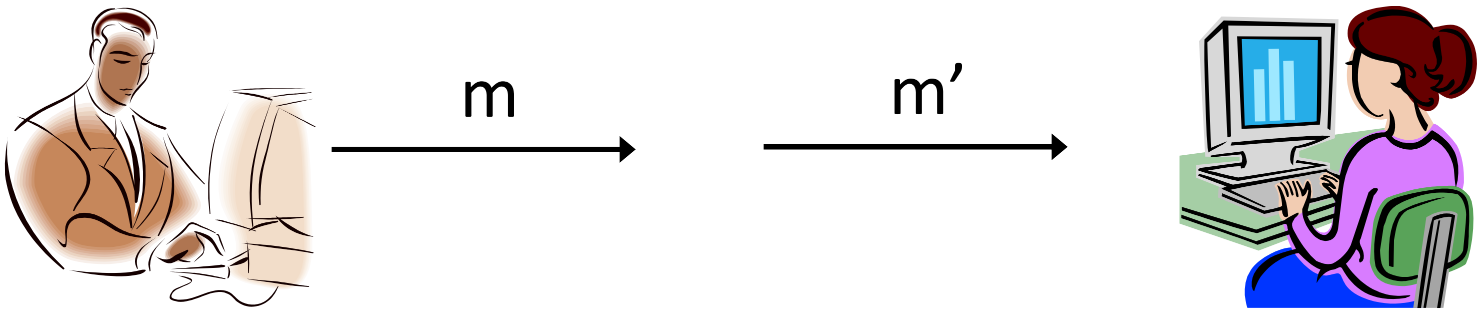
- What about *integrity*?
  - I.e., ensuring that a received message originated from the intended party, and was not modified

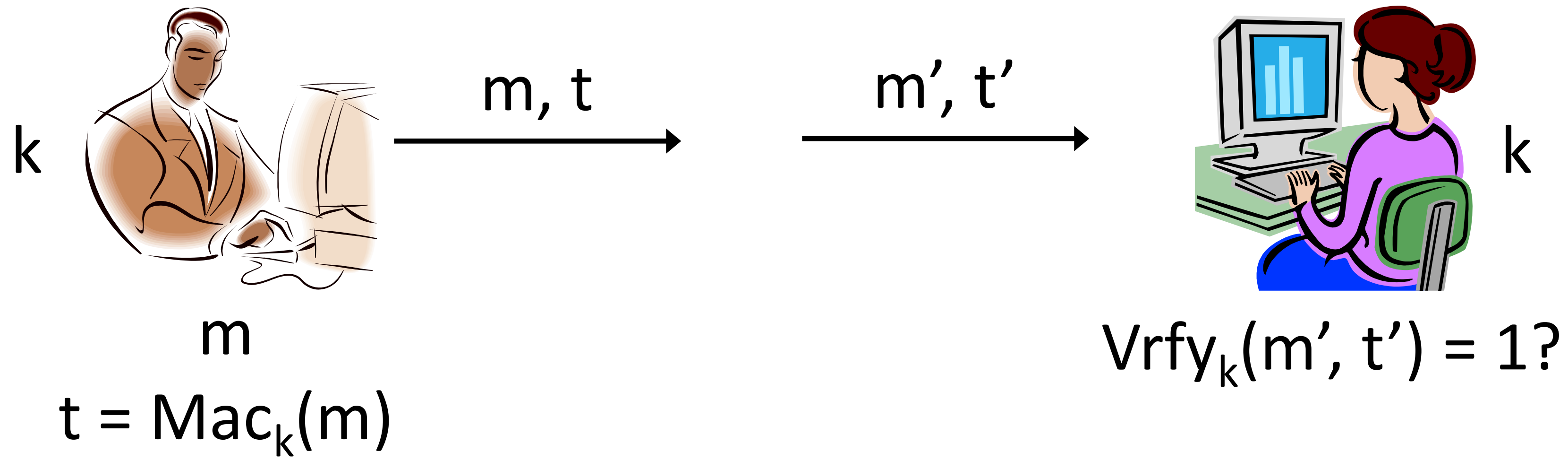
- Standard error-correction not enough!
  - The right tool is a *message authentication code*

Passive attacks vs. active attacks

- So far we have been considered only *passive* (i.e., eavesdropping) attacks
  - Attacker simply observes the channel (even if it might also carry out a chosen-plaintext attack)

- In the setting of integrity, we explicitly consider *active* attacks
  - Attacker has full control over the channel


m
m’


k
m, t
m', t'
k
m
t = Mac_k(m)
Vrfy_k(m', t') = 1?