CS 302 Computer Security and Privacy

Debayan Gupta

Lecture 1 or 2
January 22, 2019

Security of Symmetric Cryptography

Choosing a cryptosystem

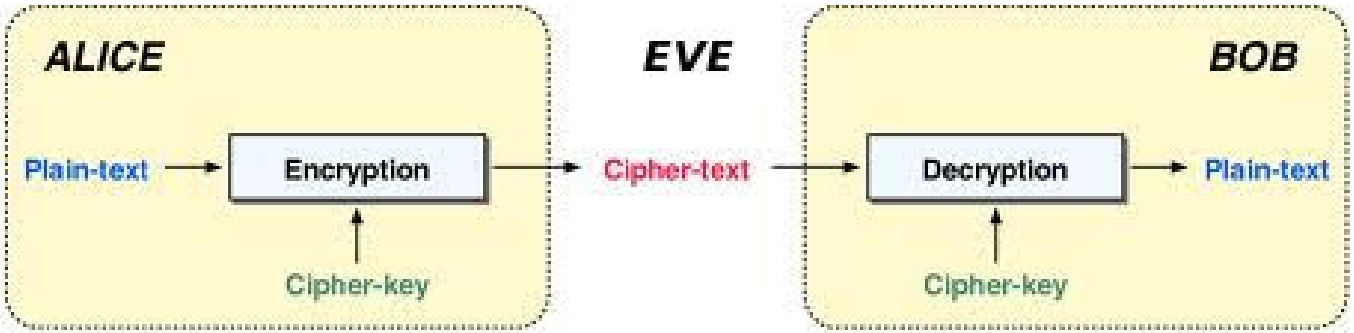

Image credit: Derived from image by Frank Kagan Gürkaynak,
`http://www.iis.ee.ethz.ch/~kgf/acacia/fig/alice_bob.png`

What cryptosystem should I use?

Desired features:

- Easy to use.
- Keeps Alice's message secure.
- Hard for Eve to break.

An analogy—choosing a car

Image credit:
`http://creativeracingcars.yolasite.com/resources/wp_Racing_Mustang_1280x800.jpg`

What car should I buy?

Desired features:

- Fast.
- Safe.
- Economical.
- Fun to drive.

Quantifying computational difficulty

Traditionally, cryptography was based on intuitive notions.

Recall the computational requirements for a symmetric cryptosystem:

Feasibility E and D, regarded as functions of two arguments, should be computable using a feasible amount of time and storage.

Security (weak) It should be difficult to find m given $c = E_k(m)$ without knowing k.

Goal: Quantify these notions.

Some important questions

1. What is an acceptable amount of time for computing E and D and on what computers?
2. What does it mean to "find" m?
   - Always?
   - Sometimes?
   - All bits of m?
   - Only some bits?
   - Some predicate of m, e.g., is m the message "I love you"?
3. What a priori knowledge does Eve have about m and k?
4. How can we choose k that is "unknown" to Eve?

Modern Cryptography

The goal of *Modern Cryptography* is to replace intuitive notions of security with mathematically precise, provable statements.

Goldwasser and Bellare follow the modern approach. We will look there for a careful treatment of the approach I will sketch here.

Giving precise answers to security questions

We next look at techniques for making precise statements about

1. Computational complexity.
2. Data confidentiality.
3. Knowledge of adversary.
4. Randomness.

Measuring computational difficulty

We want a notion of how much time is required to carry out a computational task.

Why not use actual running time?

- It depends on the speed of the computer as well as on the algorithm for computing the function.
- It varies from one input to another.
- It is difficult to analyze at a fine grained level of detail.

Role of complexity theory

Complexity theory allows one to make meaningful statements about the *asymptotic* difficulty of computational problems, independent of the particular computer or programming language.

Complexity measures *rate of growth* of worst-case running time as a function of the length n of the inputs.

An algorithm runs in time $T(n)$ if its running time on all but finitely many inputs x is at most $T(|x|)$.

An algorithm runs in *polynomial time* if it runs in time $p(n)$ for some polynomial function $p(n)$.

A function f is *polynomial time* if it is computable by some polynomial time algorithm.

Feasibility

The computational complexity of a cryptosystem measures how the time to encrypt and decrypt grows as a function of an underlying *security parameter* n.

Polynomial time functions are said to be *feasible*.

Feasibility is a minimal requirement.

In practice, we care about the actual run time for fixed values of the security parameter (such as $n = 512$).

Keeping data confidential

A naive claim of security: It is impossible for Eve to find the key.

This definition is both too strong and too weak.

Too strong We can't always prevent Eve from obtaining k.

- She can guess the key at random and will sometimes be right.
- She can try all possible keys, given enough time.

Too weak The goal of a cryptosystem is to keep m confidential. A system in which Eve can decrypt Alice's messages is totally insecure, whether or not she learns the key.

A more nuanced approach

Some compromises of decreasing difficulty for Eve:

Complete break Eve can find the key k.

- Can read all messages between Alice and Bob.
- Can send encrypted messages to Bob.

Complete message recovery Eve can decrypt all messages m.

- Can read all messages between Alice and Bob.
- Cannot encrypt her own messages to fool Bob.

Selected message recovery Eve can decrypt some subset $M \subseteq \mathcal{M}$ of messages. The larger M is, the more serious the compromise.

Attacks that do not always succeed

Eve does not always have to succeed to do damage.

Weak keys Eve can decrypt messages encrypted with keys from some subset $K \subseteq \mathcal{K}$ of "weak" keys.
The larger K is, the more serious the compromise.

Uncertain message recovery Eve can narrow down the possible plaintexts but is uncertain about the actual message.

Probabilistic algorithms Eve's attack may only succeed with some small probability.

Partial information Eve can discover some information about m.
Example: In many cryptosystems, she always learns the length of m.

What kinds of compromise are acceptable?

Eve's information

Until now, we've implicitly assumed that Eve knows nothing about the cryptosystem except for the ciphertext c.

In practice, Eve might know much more.

- She probably knows (or has a good idea) of the message distribution.
- She might have obtained several other ciphertexts.
- She might have learned the decryptions of earlier ciphertexts.
- She might have even chosen the earlier messages or ciphertexts herself.

This leads us to consider several attack scenarios.

Attack scenarios

Ciphertext-only attack Eve knows only c and tries to recover m.

Known plaintext attack Eve knows c and a sequence of plaintext-ciphertext pairs $(m_1, c_1), \ldots, (m_r, c_r)$ where $c \notin \{c_1, \ldots, c_r\}$. She tries to recover m.

Known plaintext attacks

A known plaintext attack can occur when

1. Alice uses the same key to encrypt several messages;
2. Eve later learns or successfully guesses the corresponding plaintexts.

Some ways that Eve learns plaintexts.

- The plaintext might be publicly revealed at a later time, e.g., sealed bid auctions.
- The plaintext might be guessable, e.g., an email header.
- Eve might later discover the decrypted message on Bob's computer.

Chosen text attack scenarios

Still stronger attack scenarios allow Eve to choose one element of a plaintext-ciphertext pair and obtain the other.

Chosen plaintext attack Like a known plaintext attack, except that Eve chooses messages $m_1, \ldots, m_r$ before getting c and Alice (or Bob) encrypts them for her.

Chosen ciphertext attack Like a known plaintext attack, except that Eve chooses ciphertexts $c_1, \ldots, c_r$ before getting c and Alice (or Bob) decrypts them for her.

Mixed chosen plaintext/chosen ciphertext attack Eve chooses some plaintexts and some ciphertexts and gets the corresponding decryptions or encryptions.

Why would Alice cooperate in a chosen plaintext attack?

- Eve might be authorized to generate messages that are then encrypted and sent to Bob, but she isn't authorized to read other people's messages.[1]
- Alice might be an internet server, not a person, that encrypts messages received in the course of carrying out a more complicated cryptographic protocol.[2]
- Eve might gain access to Alice's computer, perhaps only for a short time, when Alice steps away from her desk.

[1] Nothing we have said implies that Eve is unknown to Alice and Bob or that she isn't also a legitimate participant in the protocol.

[2] We will see such protocols later in the course.

Adaptive chosen text attack scenarios

Adaptive versions of chosen text protocols are when Eve chooses her texts one at a time after learning the response to her previous text.

Adaptive chosen plaintext attack Eve chooses the messages $m_1, m_2, \ldots$ one at a time rather than all at once. Thus, m_2 depends on (m_1, c_1), m_3 depends on both (m_1, c_1) and (m_2, c_2), etc.

Adaptive chosen ciphertext and adaptive mixed attacks are defined similarly.

Randomness in cryptography

Where do we assume randomness?

1. The message is drawn at random from some arbitrary probability distribution over the message space $\mathcal{M}$ which is part of Eve's *a priori* knowledge.
2. The secret key is chosen uniformly at random from the key space $\mathcal{K}$.
3. Eve has access to a source of randomness which she may use while attempting to break the system. For example, Eve can choose an element $k' \in \mathcal{K}$ at random. With probability $p = 1/|\mathcal{K}|$, her element k' is actually the correct key k.

Independence

The three sources of randomness are assumed to be *statistically independent*.

Eve's random numbers do not depend on (nor give any information about) the message or key used by Alice.

Alice's key does not depend on the particular message or vice versa.

Joint probability distribution

These multiple sources of randomness give rise to a *joint probability distribution* that assigns a well-defined probability to each triple (m, k, z), where m is a message, k a key, and z is the result of the random choices Eve makes during her computation.

The independence assumption implies that

$$\Pr[\mathrm{m}, \mathrm{k}, \mathrm{z}] = \Pr[\mathrm{m}] \times \Pr[\mathrm{k}] \times \Pr[\mathrm{z}]$$

where

- $\Pr[\mathrm{m}]$ is the probability that m is the chosen message,
- $\Pr[\mathrm{k}]$ is the probability that k is the chosen key,
- $\Pr[\mathrm{z}]$ is the probability that z represents Eve's random choices.

Eve's success probability

The joint distribution gives rise to an overall success probability for Eve (once we decide on what it means for an attack to succeed).

We want Eve's success probability to be "small".

Here, "small" is measured relative to a *security parameter* n, which you can think of as the key length.

Definition

A function f is *negligible* if for every polynomial $p(\cdot)$ there exists an N such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.

We require that the success probability be a negligible function of the security parameter.

Computational security

Putting this all together, we get a general notion of computational security.

Definition

A cryptosystem is *computationally secure* relative to a notion of compromise if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, when given as input the security parameter n and all of the information available to Eve, the algorithm succeeds in compromising the cryptosystem with success probability that is negligible in n.

Practical security considerations

In practice, the important tradeoff is between the amount of time that Alice and Bob are willing to spend to use the cryptosystem versus what a determined adversary might be willing to spend to break the system.

Asymptotic complexity results will not tell us how to set the security parameter for a system, but they may inform us about how much security improvement we can expect as the key length increases.

Probability Theory

Probability distributions and events

We give a quick overview of probability theory.

A *discrete probability distribution* p assigns a real number $p_\omega \in [0, 1]$ to each element ω of a probability space Ω such that

$$\sum_{\omega \in \Omega} p_\omega = 1.$$

An *event* E is a subset of Ω. The probability of E is

$$\Pr[\mathrm{E}] = \sum_{\omega \in \mathrm{E}} \mathrm{p}_\omega.$$

Random variables

A *random variable* is a function $X : \Omega \to \mathcal{X}$, where $\mathcal{X}$ is a set.

We think of X as describing a random choice according to distribution p.

Let $x \in \mathcal{X}$. Event $X = x$ means that the outcome of choice X is x.

Formally, the event $X = x$ is the set $\{\omega \in \Omega \mid X(\omega) = x\}$.
Its probability is therefore

$$\Pr[\mathrm{x} = \mathrm{X}] = \sum_{\omega:\mathrm{X}(\omega)=\mathrm{x}} \mathrm{p}_\omega.$$

We sometimes ambiguously write x to denote the event $X = x$.

Experiments

Sometimes m denotes the random variable that describes the experiment of Alice choosing a message $m \in \mathcal{M}$ according to the assumed message distribution.

Other times, m denotes a particular message in set $\mathcal{M}$.

Hopefully, which meaning is intended will be clear from context.

Conditional probability

Let E and F be events and assume $\Pr[\mathrm{F}] \neq 0$. The conditional probability of E given F is defined by

$$\Pr[\mathrm{E}|\mathrm{F}] = \Pr[\mathrm{E} \cap \mathrm{F}]/\Pr[\mathrm{F}].$$

Intuitively, it is the probability that E holds given that F is known to hold.

Example:

Ω	p
1	.2
2	.2
3	.3
4	.1
5	.2

$E = \{1, 2, 3\}$, $F = \{2, 3, 4\}$.

$\Pr[\mathrm{E}] = .7$

$\Pr[\mathrm{F}] = .6$

$\Pr[\mathrm{E} \cap \mathrm{F}] = .5$

$\Pr[\mathrm{E}|\mathrm{F}] = .2/.6 + .3/.6 = 5/6.$

Statistical independence

Formally, events E and F are *statistically independent* if $\Pr[\mathrm{E}|\mathrm{F}] = \Pr[\mathrm{E}]$.

An equivalent definition is that $\Pr[\mathrm{E} \cap \mathrm{F}] = \Pr[\mathrm{E}] \cdot \Pr[\mathrm{F}]$.

This is easily seen by dividing both sides by $\Pr[\mathrm{F}]$ and applying the definition of $\Pr[\mathrm{E}|\mathrm{F}]$.

Perfect Secrecy

Information-theoretic security

We have been discussing *computational security* – what can Eve learn with a certain success probability in a limited amount of time.

We now turn to *information-theoretic security*, where we remove the limits on Eve and just look at the question from an information-theoretic perspective.
What information is contained in the data at Eve's disposal?

A cryptosystem is *information-theoretically secure* if $\Pr[\mathrm{m}] = \Pr[\mathrm{m}|\mathrm{c}]$. Thus, c gives no information about m.

This is equivalent to saying that m and c are *statistically independent*.

We also call this *perfect secrecy*.