CSCI-GA.3205 Applied Cryptography & Network Security

Department of Computer Science
New York University

PRESENTED BY DR. MAZDAK ZAMANI
mazdak.zamani@NYU.edu

Perfectly Secure
Semantically Secure

NYU

Perfectly Secret Encryption

- In the previous chapter we presented historical encryption schemes and showed how they can be broken with little computational effort.
- In this chapter, we look at the other extreme and study encryption schemes that are provably secure even against an adversary with unbounded computational power. Such schemes are called *perfectly secret*.

Where do random bits come from?

- In principle, one could generate a small number of random bits by hand, e.g., by flipping a fair coin.

Generating randomness

- Modern random-number generation proceeds in two steps.
  - First, a “pool” of high-entropy data is collected. (Think of entropy as a measure of unpredictability.)
  - Next, this high-entropy data is processed to yield a sequence of nearly independent and unbiased bits. This second step is necessary since high-entropy data is not necessarily uniform.

Source of unpredictable data

- One technique is to rely on external inputs, for example, delays between network events, hard-disk access times, keystrokes or mouse movements made by the user, and so on.
  - Such data is likely to be far from uniform, but if enough measurements are taken the resulting pool of data is expected to have sufficient entropy.
- More sophisticated approaches—which, by design, incorporate random-number generation more tightly into the system at the hardware level—have also been used.
  - Intel has recently developed a processor that includes a digital random-number generator on the processor chip.

A simple example of random-number generation

- Imagine that our high-entropy pool results from a sequence of biased coin flips, where “heads” occurs with probability p and “tails” with probability 1 − p.
  - The result of 1,000 such coin flips certainly has high entropy but is not close to uniform.
- We can obtain a uniform distribution by considering the coin flips in pairs: if we see a head followed by a tail then we output “0,” and if we see a tail followed by a head then we output “1.”
  - If we see two heads or two tails in a row, we output nothing, and simply move on to the next pair.
- The probability that any pair results in a “0” is $p \cdot (1-p)$, which is exactly equal to the probability that any pair results in a “1,” and we thus obtain a uniformly distributed output from our initial high-entropy pool.

“General-purpose” random-number generator,

- One should use a random-number generator that is designed for cryptographic use, rather than a “general-purpose” random-number generator, which is not suitable for cryptographic applications.
  - The rand() function in the C stdlib.h library is not cryptographically secure, and using it in cryptographic settings can have disastrous consequences.

NYU

Definitions

- An encryption scheme is defined by three algorithms Gen, Enc, and Dec, as well as a specification of a (finite) *message space* **M** with |**M**| > 1.
- The key-generation algorithm Gen is a probabilistic algorithm that outputs a key k chosen according to some distribution.
- We denote by **K** the (finite) key space, i.e., the set of all possible keys that can be output by Gen.
- The encryption algorithm Enc takes as input a key $k \in \mathbf{K}$ and a message $m \in \mathbf{M}$, and outputs a ciphertext c.

Definitions: Deterministic vs Probabilistic

- We now allow the encryption algorithm to be probabilistic
  - $\text{Enc}_k(m)$ might output a different ciphertext when run multiple times
- We write $c \leftarrow \text{Enc}_k(m)$ to denote the possibly probabilistic process by which message m is encrypted using key k to give ciphertext c.
- If Enc is deterministic, we may emphasize this by writing $c := \text{Enc}_k(m)$.

Definitions – Con'd

- Use the notation $x \leftarrow \mathbf{S}$ to denote uniform selection of x from a set **S**.)
- We let **C** denote the set of all possible ciphertexts that can be output by $\text{Enc}_k(m)$, for all possible choices of $k \in \mathbf{K}$ and $m \in \mathbf{M}$ (and for all random choices of Enc in case it is randomized).
- The decryption algorithm Dec takes as input a key $k \in \mathbf{K}$ and a ciphertext $c \in \mathbf{C}$ and outputs a message $m \in \mathbf{M}$.

Definitions – Con'd

- We assume *perfect correctness*, meaning that for all $k \in \mathbf{K}$, $m \in \mathbf{M}$, and any ciphertext c output by $\text{Enc}_k(m)$, it holds that $\text{Dec}_k(c) = m$ with probability 1.
- Perfect correctness implies that we may assume Dec is deterministic without loss of generality, since $\text{Dec}_k(c)$ must give the same output every time it is run. We will thus write $m := \text{Dec}_k(c)$ to denote the process of decrypting ciphertext c using key k to yield the message m.

Probability distributions

- The distribution over **K** is the one defined by running Gen and taking the output. (Gen chooses a key uniformly from **K**)
- We let K be a random variable denoting the value of the key output by Gen; thus, for any $k \in \mathbf{K}$, $\Pr[K = k]$ denotes the probability that the key output by Gen is equal to k.
- Similarly, we let M be a random variable denoting the message being encrypted, so $\Pr[M = m]$ denotes the probability that the message takes on the value $m \in \mathbf{M}$.

Probability distributions – Con’d

- The probability distribution of the message is not determined by the encryption scheme itself, but instead reflects the likelihood of different messages being sent by the parties using the scheme, as well as an adversary’s uncertainty about what will be sent.
  - As an example, an adversary may know that the message will either be attack today or don’t attack. The adversary may even know (by other means) that with probability 0.7 the message will be a command to attack and with probability 0.3 the message will be a command not to attack. In this case, we have Pr[M = *attack today*] = 0.7 and Pr[M = *don’t attack*] = 0.3.

Probability distributions – Con’d

- *K* and *M* are assumed to be independent
  - What is being communicated by the parties is independent of the key they happen to share.
  - This makes sense, among other reasons, because the distribution over **K** is determined by the encryption scheme itself (since it is defined by Gen), while the distribution over **M** depends on the context in which the encryption scheme is being used.

Probability distributions – Con’d

- Fixing an encryption scheme and a distribution over **M** determines a distribution over the space of ciphertexts **C** given by choosing a key $k \in \mathbf{K}$ (according to Gen) and a message $m \in \mathbf{M}$ (according to the given distribution), and then computing the ciphertext $c \leftarrow \mathrm{Enc}_k(m)$.
- We let C be the random variable denoting the resulting ciphertext and so, for $c \in \mathbf{C}$, write $\Pr[C = c]$ to denote the probability that the ciphertext is equal to the fixed value c.

Example 2.1

We work through a simple example for the shift cipher (cf. Section 1.3). Here, by definition, we have $\mathcal{K} = \{0, \ldots, 25\}$ with $\Pr[K = k] = 1/26$ for each $k \in \mathcal{K}$.

Say we are given the following distribution over $\mathcal{M}$:

$$\Pr[M = \mathtt{a}] = 0.7 \quad \text{and} \quad \Pr[M = \mathtt{z}] = 0.3.$$

What is the probability that the ciphertext is B? There are only two ways this can occur: either $M = \mathtt{a}$ and $K = 1$, or $M = \mathtt{z}$ and $K = 2$. By independence of M and K, we have

$$\begin{aligned}\Pr[M = \mathtt{a} \wedge K = 1] &= \Pr[M = \mathtt{a}] \cdot \Pr[K = 1] \\ &= 0.7 \cdot \left(\frac{1}{26}\right).\end{aligned}$$

Similarly, $\Pr[M = \mathtt{z} \wedge K = 2] = 0.3 \cdot \left(\frac{1}{26}\right)$. Therefore,

$$\begin{aligned}\Pr[C = \mathtt{B}] &= \Pr[M = \mathtt{a} \wedge K = 1] + \Pr[M = \mathtt{z} \wedge K = 2] \\ &= 0.7 \cdot \left(\frac{1}{26}\right) + 0.3 \cdot \left(\frac{1}{26}\right) = 1/26.\end{aligned}$$

We can calculate conditional probabilities as well. For example, what is the probability that the message a was encrypted, given that we observe ciphertext B? Using Bayes' Theorem (Theorem A.8) we have

$$\begin{aligned}\Pr[M = \mathtt{a} \mid C = \mathtt{B}] &= \frac{\Pr[C = \mathtt{B} \mid M = \mathtt{a}] \cdot \Pr[M = \mathtt{a}]}{\Pr[C = \mathtt{B}]} \\ &= \frac{0.7 \cdot \Pr[C = \mathtt{B} \mid M = \mathtt{a}]}{1/26}.\end{aligned}$$

Note that $\Pr[C = \mathtt{B} \mid M = \mathtt{a}] = 1/26$, since if $M = \mathtt{a}$ then the only way $C = \mathtt{B}$ can occur is if $K = 1$ (which occurs with probability $1/26$). We conclude that $\Pr[M = \mathtt{a} \mid C = \mathtt{B}] = 0.7$. ♢

NYU

Example 2.2

Consider the shift cipher again, but with the following distribution over $\mathcal{M}$:

$$\Pr[M = \texttt{kim}] = 0.5,\ \Pr[M = \texttt{ann}] = 0.2,\ \Pr[M = \texttt{boo}] = 0.3.$$

What is the probability that $C = \texttt{DQQ}$? The only way this ciphertext can occur is if $M = \texttt{ann}$ and $K = 3$, or $M = \texttt{boo}$ and $K = 2$, which happens with probability $0.2 \cdot 1/26 + 0.3 \cdot 1/26 = 1/52$.

So what is the probability that `ann` was encrypted, conditioned on observing the ciphertext `DQQ`? A calculation as above using Bayes' Theorem gives $\Pr[M = \texttt{ann} \mid C = \texttt{DQQ}] = 0.4$. ♢

Binary Cipher

- XOR encryption is a symmetrical encryption/decryption method based on the use of the logical/binary operator XOR (also called Exclusive Or , symbolized by ⊕). The XOR cipher uses as operands the plain text and the key (previously encoded in binary/bit string).
- The XOR operation takes 2 bits as input and returns one bit as output according to the following table: if the two bits are different, the result is 1, otherwise the result is 0.

A	B	A **xor** B
0	0	0
0	1	1
1	0	1
1	1	0

NYU

Binary Cipher

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

NYU

Binary Cipher - Example

- Message: ny ☾ 0110 1110 0111 1001
- Key: dc ☾ 0110 0100 0110 0011
- Binary Cipher? 0000 1010 0001 1010

NYU

Perfect secrecy

- The adversary knows the likelihood that different messages will be sent.
- This adversary also knows the encryption scheme being used.
- The only thing unknown to the adversary is the key shared by the parties.
- The adversary can eavesdrop on the parties' communication, and thus observe this ciphertext.
- For a scheme to be perfectly secret, observing this ciphertext should have no effect on the adversary's knowledge regarding the actual message that was sent.(Adversary learns absolutely nothing about the plaintext)

DEFINITION 2.3 *An encryption scheme* (Gen, Enc, Dec) *with message space* $\mathcal{M}$ *is* perfectly secret *if for every probability distribution over* $\mathcal{M}$*, every message* $m \in \mathcal{M}$*, and every ciphertext* $c \in \mathcal{C}$ *for which* $\Pr[C = c] > 0$*:*

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

We now give an equivalent formulation of perfect secrecy. Informally, this formulation requires that the probability distribution of the ciphertext does not depend on the plaintext, i.e., for any two messages $m, m' \in \mathcal{M}$ the distribution of the ciphertext when m is encrypted should be identical to the distribution of the ciphertext when m' is encrypted. Formally, for every $m, m' \in \mathcal{M}$, and every $c \in \mathcal{C}$,

$$\Pr[\mathsf{Enc}_K(m) = c] = \Pr[\mathsf{Enc}_K(m') = c] \tag{2.1}$$

Perfectly secret (Formally)

LEMMA 2.4 *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with message space* $\mathcal{M}$ *is perfectly secret if and only if Equation (2.1) holds for every* $m, m' \in \mathcal{M}$ *and every* $c \in \mathcal{C}$.

NYU

Perfect (adversarial) indistinguishability

- This definition is based on an *experiment* involving an adversary passively observing a ciphertext and then trying to guess which of two possible messages was encrypted.
- An adversary A first specifies two arbitrary messages m_0, $m_1 \in \mathbf{M}$.
- One of these two messages is chosen uniformly at random and encrypted using a random key; the resulting ciphertext is given to *A*.
- Finally, *A* outputs a “guess” as to which of the two messages was encrypted; A *succeeds* if it guesses correctly.
- An encryption scheme is *perfectly indistinguishable* if no adversary A can succeed with probability better than 1/2.

The adversarial indistinguishability experiment

1. *The adversary* $\mathcal{A}$ *outputs a pair of messages* $m_0, m_1 \in \mathcal{M}$.
2. *A key* k *is generated using* Gen, *and a uniform bit* $b \in \{0,1\}$ *is chosen. Ciphertext* $c \leftarrow \mathsf{Enc}_k(m_b)$ *is computed and given to* $\mathcal{A}$. *We refer to* c *as the* challenge ciphertext.
3. $\mathcal{A}$ *outputs a bit* b'.
4. *The output of the experiment is defined to be* 1 *if* $b' = b$, *and* 0 *otherwise. We write* $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1$ *if the output of the experiment is* 1 *and in this case we say that* $\mathcal{A}$ succeeds.

As noted earlier, it is trivial for $\mathcal{A}$ to succeed with probability $1/2$ by outputting a random guess. Perfect indistinguishability requires that it is impossible for any $\mathcal{A}$ to do better.

DEFINITION 2.5 *Encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with message space* $\mathcal{M}$ *is* perfectly indistinguishable *if for every* $\mathcal{A}$ *it holds that*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1\right] = \frac{1}{2}.$$

The following lemma states that Definition 2.5 is equivalent to Definition 2.3. We leave the proof of the lemma as Exercise 2.5.

LEMMA 2.6 *Encryption scheme* Π *is perfectly secret if and only if it is perfectly indistinguishable.*

The One-Time Pad

- In 1917, Vernam patented a perfectly secret encryption scheme now called the *one-time pad*.
- At the time Vernam proposed the scheme, there was no proof that it was perfectly secret; in fact, there was not yet a notion of what perfect secrecy was.
- Approximately 25 years later, however, Shannon introduced the definition of perfect secrecy and demonstrated that the one-time pad achieves that level of security.

The one-time pad encryption scheme

CONSTRUCTION 2.8

Fix an integer $\ell > 0$. The message space $\mathcal{M}$, key space $\mathcal{K}$, and ciphertext space $\mathcal{C}$ are all equal to $\{0,1\}^\ell$ (the set of all binary strings of length ℓ).

- Gen: the key-generation algorithm chooses a key from $\mathcal{K} = \{0,1\}^\ell$ according to the uniform distribution (i.e., each of the 2^ℓ strings in the space is chosen as the key with probability exactly $2^{-\ell}$).
- Enc: given a key $k \in \{0,1\}^\ell$ and a message $m \in \{0,1\}^\ell$, the encryption algorithm outputs the ciphertext $c := k \oplus m$.
- Dec: given a key $k \in \{0,1\}^\ell$ and a ciphertext $c \in \{0,1\}^\ell$, the decryption algorithm outputs the message $m := k \oplus c$.

Drawbacks of the one-time pad

- The key is as long as the message.
- the one-time pad—as the name indicates—is only secure if used once (with the same key).

- Limitations of Perfect Secrecy
  - Any perfectly secret encryption scheme must have a key space that is at least as large as the message space

THEOREM 2.10 *If* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a perfectly secret encryption scheme with message space* $\mathcal{M}$ *and key space* $\mathcal{K}$*, then* $|\mathcal{K}| \geq |\mathcal{M}|$.

NYU

Shannon's Theorem

THEOREM 2.11 (Shannon's theorem) *Let* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme with message space* $\mathcal{M}$*, for which* $|\mathcal{M}| = |\mathcal{K}| = |\mathcal{C}|$*. The scheme is perfectly secret if and only if:*

1. *Every key* $k \in \mathcal{K}$ *is chosen with (equal) probability* $1/|\mathcal{K}|$ *by algorithm* Gen.
2. *For every* $m \in \mathcal{M}$ *and every* $c \in \mathcal{C}$*, there exists a unique key* $k \in \mathcal{K}$ *such that* $\mathsf{Enc}_k(m)$ *outputs* c.

NYU

Computational ciphers and semantic security

- As we have seen in Shannon's theorem, the only way to achieve perfect security is to have keys that are as long as messages. However, this is quite impractical.
- The only way around Shannon's theorem is to relax our security requirements.
- The way we shall do this is to consider not all possible adversaries, but only *computationally feasible* adversaries, that is, “real world” adversaries that must perform their calculations on real computers using a reasonable amount of time and memory.

Definition of a computational cipher

- A computational cipher $\xi = (E; D)$ is a pair of efficient algorithms, E and D. The encryption algorithm E takes as input a key *k*, along with a message *m*, and produces as output a ciphertext *c*.
- The decryption algorithm D takes as input a key *k*, a ciphertext *c*, and outputs a message *m*.
- Keys lie in some finite key space K, messages lie in a finite message space M, and ciphertexts lie in some finite ciphertext space C.

Definition of a computational cipher – con’d

- Just as for a Shannon cipher, we say that ξ is defined over (K; M; C).
- We will allow the encryption function E to be a probabilistic algorithm $c \leftarrow \text{Enc}(k, m)$
- From now on, whenever we refer to a cipher, we shall mean a computational cipher, as defined above. Moreover, if the encryption algorithm happens to be deterministic, then we may call the cipher a deterministic cipher.

Definition of semantic security

To motivate the definition of semantic security, consider a deterministic cipher $\xi = (E, D)$, defined over (K, M, C).
For all predicates φ on the ciphertext space, and all messages m_0, m_1, we have

$$\Pr[\varphi(E(k, m_0))] = \Pr[\varphi(E(k, m_1))]$$

Where k is a random variable uniformly distributed over the key space K. Instead of insisting that these probabilities are equal, we shall only require that they are very close; that is,

$$\Pr[\varphi(E(k, m_0))] - \Pr[\varphi(E(k, m_1))] \leq \varepsilon$$

for some very small, or *negligible*, value of ε.

Example of semantic security

- Suppose it were the case that using the best possible algorithms for generating m_0, and m_1, and for testing some predicate φ, and using (say) 10,000 computers in parallel for 10 years to perform these calculations, it holds for $\varepsilon = 2^{-100}$.
- While not perfectly secure, we might be willing to say that the cipher is *secure for all practical purposes*.

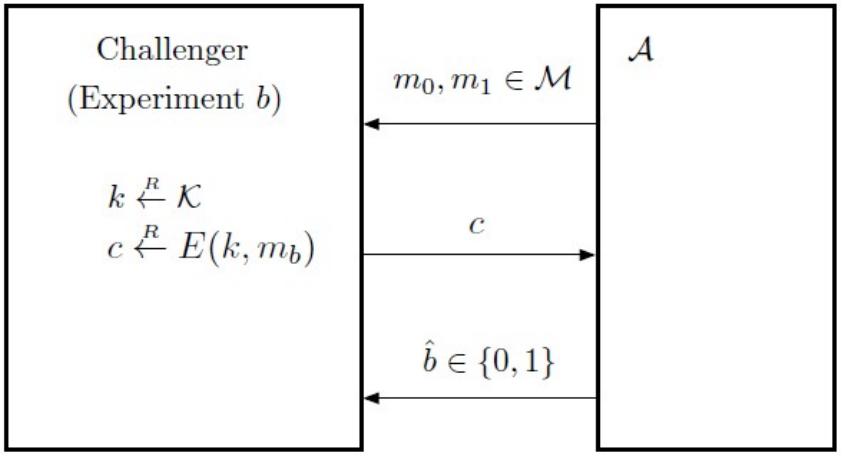
NYU

Attack Game

Attack Game 2.1 (semantic security). For a given cipher $\mathcal{E} = (E, D)$, defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, and for a given adversary $\mathcal{A}$, we define two experiments, Experiment 0 and Experiment 1. For $b = 0, 1$, we define

Experiment b**:**

- The adversary computes $m_0, m_1 \in \mathcal{M}$, of the same length, and sends them to the challenger.
- The challenger computes $k \xleftarrow{R} \mathcal{K}$, $c \xleftarrow{R} E(k, m_b)$, and sends c to the adversary.
- The adversary outputs a bit $\hat{b} \in \{0, 1\}$.



Semantically secure

For $b = 0, 1$, let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. We define $\mathcal{A}$'s **semantic security advantage** with respect to $\mathcal{E}$ as

$$\mathrm{SSadv}[\mathcal{A}, \mathcal{E}] := \Big|\Pr[W_0] - \Pr[W_1]\Big|. \quad \square$$

Note that in the above game, the events W_0 and W_1 are defined with respect to the probability space determined by the random choice of k, the random choices made (if any) by the encryption algorithm, and the random choices made (if any) by the adversary. The value $\mathrm{SSadv}[\mathcal{A}, \mathcal{E}]$ is a number between 0 and 1.

Definition 2.2 (semantic security). *A cipher $\mathcal{E}$ is **semantically secure** if for all efficient adversaries $\mathcal{A}$, the value $\mathrm{SSadv}[\mathcal{A}, \mathcal{E}]$ is negligible.*

An *efficient adversary* is one that runs in a "reasonable" amount time.