Introduction to the basics of cryptography

Monday, January 20, 2025 5:25 AM

5. Pseudorandom objects in practice

Private/Secret/symmetric Key Encryption

Syntax:

1. The *key-generation algorithm* Gen is a probabilistic algorithm that outputs a key k chosen according to some distribution that is determined by the scheme.
2. The *encryption algorithm* Enc takes as input a key k and a plaintext m and outputs a ciphertext c. We denote the encryption of the plaintext m using the key k by $\mathsf{Enc}_k(m)$.
3. The *decryption algorithm* Dec takes as input a key k and a ciphertext c and outputs a plaintext m. We denote the decryption of the ciphertext c using the key k by $\mathsf{Dec}_k(c)$.

$$\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m.$$

secret from everyone else. But maybe they should keep Dec a secret, too? For that matter, perhaps all the algorithms constituting the encryption scheme (i.e., Gen and Enc as well) should be kept secret? (Note that the plaintext space $\mathcal{M}$ is typically assumed to be known, e.g., it may consist of English-language sentences.)

In the late 19th century, Auguste Kerckhoffs gave his opinion on this matter in a paper he published outlining important design principles for military ciphers. One of the most important of these principles (known now simply as Kerckhoffs' principle) was the following:

> *The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.*

being used. Stated differently, Kerckhoffs' principle demands that *security rely solely on the secrecy of the key*. But why?

There are two primary arguments in favor of Kerckhoffs principle. The first is that it is much easier for the parties to maintain secrecy of a short key than to maintain secrecy of an algorithm. It is easier to share aa short (say, 100-bit) string and store this string securely than it is to share and securely store a program that is thousands of times larger. Furthermore, details of an algorithm can be leaked (perhaps by an insider) or learned through reverse engineering; this is unlikely when the secret information takes the form of a randomly-generated string.

from public view. Some of the advantages of "open cryptographic design", where the algorithm specifications are made public, include:

1. Published designs undergo public scrutiny and are therefore likely to be stronger. Many years of experience have demonstrated that it is very difficult to construct good cryptographic schemes. Therefore, our confidence in the security of a scheme is much higher after it has been extensively studied and has withstood many attack attempts.
2. It is better that security flaws are revealed by "ethical hackers" and made public, than having the flaws be known only to malicious parties.
3. If the security of the system relies on the secrecy of the algorithm, then reverse engineering of the code (or leakage by industrial espionage) poses a serious threat to security. This is in contrast to the secret key which is not part of the code, and so is not vulnerable to reverse engineering.
4. Public design enables the establishment of standards.

Attack scenarios. We wrap up our general discussion of encryption with a brief discussion of some basic types of attacks against encryption schemes (these will be helpful in the next section). In order of severity, these are:

- Ciphertext-only attack: This is the most basic type of attack and refers to the scenario where the adversary just observes a ciphertext and attempts to determine the plaintext that was encrypted.
- Known-plaintext attack: Here, the adversary learns one or more pairs of plaintexts/ciphertexts encrypted under the same key. The aim of the adversary is then to determine the plaintext that was encrypted to give some *other* ciphertext (for which it does not know the corresponding plaintext).
- Chosen-plaintext attack: In this attack, the adversary has the ability to obtain the encryption of any plaintext(s) of its choice. It then attempts to determine the plaintext that was encrypted to give some other ciphertext.
- Chosen-ciphertext attack: The final type of attack is one where the adversary is even given the capability to obtain the decryption of any ciphertext(s) of its choice. The adversary's aim, once again, is then to determine the plaintext that was encrypted to give some other ciphertext (whose decryption the adversary is unable to obtain directly).

Basic Principles of Modern Cryptography

(1) Security definition
(2) Construction satisfying tihs definition; if an assumption is needed, it must be minimal and precisely stated; security by reduction to this assumption

Principle (1)

E.g.
How should security of encryptions be defined?

Some wrong answers, and then finally the "right" answer

1. No adversary can find the secret key from a given ciphertext
   E.g. encryption scheme that ignores the secret key but just outputs the message
2. No adversary can find the plaintext that corresponds to the ciphertext
   E.g. What if 90% of the plaintext is revealed and one can show the remaining 10% cannot be learnt?
3. No adversary can find any of the plaintext that corresponds to teh ciphertext
   E.g. Suppose a salary is encrypted. May be impossible to determine the actual salary but could potentially learn whether or not the salary is greater than some fixed amount
4. No adversary can extract any meaningful information about the plaintext from the ciphertext
   Unclear what meaningful means mathematically—
   we want a scheme that works in all situations, and meaning seems to be application specific
5. Final answer—
   No adversary can compute *any* function of the plaintext from the ciphertext

Important: attack model of the adversary
Specifies actions, and limits on abilities

A cryptographic scheme for a given task is secure if no adversary of a specified power can achieve a specified break.

Model real world:
done by using Turing Machines to capture the most general mathematical model of capability of adversaries in practice

Principle 2—precise assumptions and reductions

1. *Validation of the assumption:* By their very nature, assumptions are statements that are not proven but are rather conjectured to be true. In order to strengthen this conjecture, it is necessary for the assumption to be studied. The basic understanding is that the more the assumption is looked at without being successfully refuted, the more confident we are that the assumption is true. Furthermore, study of an assumption can provide positive evidence of its validity by showing that it is implied by some other assumption that is also widely believed.

   If the assumption being relied upon is not precisely stated and presented, it cannot be studied and (potentially) refuted. Thus, a pre-condition to raising our confidence in an assumption is having a precise statement of what exactly is assumed.

2. *Comparison of schemes:* Often in cryptography, we may be presented with two schemes that can both be proven to satisfy some definition but each with respect to a different assumption. Assuming both schemes are equally efficient, which scheme should be preferred? If the assumption that one scheme is based on is *weaker* than the assumption the second scheme is based on (i.e., the second assumption implies the first), then the first scheme is to be preferred since it may turn out that the second assumption is false while the first assumption is true. If the assumptions used by the two schemes are incomparable, then the general rule is to prefer the scheme that is based on the better-studied assumption (for the reasons highlighted in the previous paragraphs).

3. *Facilitation of a proof of security:* As we have stated, and will discuss in more depth in principle 3, modern cryptographic constructions are presented together with proofs of security. If the security of the scheme cannot be proven unconditionally and must rely on some assumption, then a mathematical proof that "the construction is secure if the assumption is true" can only be provided if there is a precise statement of what the assumption is.

One observation is that it is always possible to just assume that a construction *itself* is secure. If security is well defined, this is also a precise assumption (and the proof of security for the construction is trivial)! Of course, this is not accepted practice in cryptography (for the most part) for a number of reasons. First of all, as noted above, an assumption that has been tested over the years is preferable to a new assumption that is introduced just to prove a given construction secure. Second, there is a general preference for assumptions that are simpler to state, since such assumptions are easier to study and to refute. So, for example, an assumption of the type that some mathematical problem is hard to solve is simpler to study and work with than an assumption that an encryption schemes satisfies a complex (and possibly unnatural) security definition. When a simple assumption is studied at length and still no refutation is found, we have greater confidence in its being correct. Another advantage of relying on "lower-level" assumptions (rather than just assuming a scheme is secure) is that these low-level assumptions can typically be shared amongst a number of constructions. If a specific instantiation of the assumption turns out to be false, it can be replaced within the higher-level constructions by another instantiation of that assumption.

The reductionist approach. We conclude by noting that most proofs in modern cryptography use what may be called the reductionist approach. Given a theorem of the form

> *Given that Assumption X is true, Construction Y is secure according to the given definition,*

a proof typically shows how to *reduce* the problem given by Assumption X to the problem of breaking Construction Y. More to the point, the proof will typically show (via a constructive argument) how any adversary breaking Construction Y can be used as a sub-routine to violate Assumption X. We will have more to say about this in Section 3.1.3.

Ch 2 Perfectly-secret encryption

Given by three algorithms, Gen Enc Dec

$\mathcal{K}, \mathcal{M}, \mathcal{C}$

by $\mathsf{Enc}_k(m)$. The encryption algorithm may be probabilistic, so that $\mathsf{Enc}_k(m)$ might output a different ciphertext when run multiple times. To emphasize this, we write $c \leftarrow \mathsf{Enc}_k(m)$ to denote the (possibly probabilistic) process by

which message m is encrypted using key k to give ciphertext c. (In case Enc is deterministic, we may emphasize this by writing $c := \mathsf{Enc}_k(m)$.) We let $\mathcal{C}$ denote the set of all possible ciphertexts that can be output by $\mathsf{Enc}_k(m)$, for

DEFINITION 2.1 *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *over a 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* $\Pr[C = c] > 0$*:*

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

LEMMA 2.3 *An encryption scheme* (Gen, Enc, Dec) *over a message space* $\mathcal{M}$ *is perfectly secret if and only if for every probability distribution over* $\mathcal{M}$*, every* $m_0, m_1 \in \mathcal{M}$*, and every* $c \in \mathcal{C}$*:*

$$\Pr[C = c \mid M = m_0] = \Pr[C = c \mid M = m_1].$$

The adversarial indistinguishability experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$**:**

1. *The adversary* $\mathcal{A}$ *outputs a pair of messages* $m_0, m_1 \in \mathcal{M}$*.*
2. *A random key* k *is generated by running* Gen*, and a random bit* $b \leftarrow \{0,1\}$ *is chosen. (These are chosen by some imaginary entity that is running the experiment with* $\mathcal{A}$*.) Then, the ciphertext* $c \leftarrow \mathsf{Enc}_k(m_b)$ *is computed and given to* $\mathcal{A}$*.*
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 is 1 and in this case we say that* $\mathcal{A}$ succeeded.

DEFINITION 2.4 (perfect secrecy — alternative definition): *An encryption scheme* (Gen, Enc, Dec) *over a message space* $\mathcal{M}$ *is* perfectly secret *if for every adversary* $\mathcal{A}$ *it holds that*

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

One Time Pad

The one-time pad encryption scheme is defined as follows:

1. Fix an integer $\ell > 0$. Then the message space $\mathcal{M}$, key space $\mathcal{K}$, and ciphertext space $\mathcal{C}$ are all equal to $\{0,1\}^\ell$ (i.e., the set of all binary strings of length ℓ).
2. The key-generation algorithm Gen works by choosing a string 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 $2^{-\ell}$).
3. Encryption Enc works as follows: given a key $k \in \{0,1\}^\ell$ and a message $m \in \{0,1\}^\ell$, output $c := k \oplus m$.
4. Decryption Dec works as follows: given a key $k \in \{0,1\}^\ell$ and a ciphertext $c \in \{0,1\}^\ell$, output $m := k \oplus c$.

THEOREM 2.6 *The one-time pad is a perfectly-secret encryption scheme.*

THEOREM 2.7 *Let* (Gen, Enc, Dec) *be a perfectly-secret encryption scheme over a message space* $\mathcal{M}$*, and let* $\mathcal{K}$ *be the key space as determined by* Gen*. Then* $|\mathcal{K}| \geq |\mathcal{M}|$*.*

Ch 3 Private-Key Encryption and Pseudorandomness

The computational approach incorporates two relaxations of the notion of perfect security:

1. *Security is only preserved against efficient adversaries,* and → PPT
2. *Adversaries can potentially succeed with some very small probability (small enough so that we are not concerned that it will ever really happen).* → smaller thahn inverse-poly

A scheme is secure *if every* PPT *adversary succeeds in breaking the scheme with only negligible probability.*

A technical remark. As we have mentioned, we view the running time of the adversary and the honest parties as a function of n. To be consistent with the standard convention in algorithms and complexity theory, where the running time of an algorithm is measured as a function of the length of its input, we will thus provide the adversary and the honest parties with the security parameter in unary as 1^n (i.e., a string of n 1's) when necessary.

DEFINITION 3.5 *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* $f(n) < \frac{1}{p(n)}$*.*

Example 3.6
The functions $2^{-n}, 2^{-\sqrt{n}}, n^{-\log n}$ are all negligible.

PROPOSITION 3.7 *Let negl_1 and negl_2 be negligible functions.*

1. *The function negl_3 defined by $\mathsf{negl}_3(n) = \mathsf{negl}_1(n) + \mathsf{negl}_2(n)$ is negligible.*
2. *For any positive polynomial p, the function negl_4 defined by $\mathsf{negl}_4(n) = p(n) \cdot \mathsf{negl}_1(n)$ is negligible.*

§ 3.2 A Definition of Computationally-Secure Encryption

DEFINITION 3.8 *A private-key encryption scheme is a tuple of probabilistic polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:*

1. *The key-generation algorithm Gen takes as input the security parameter 1^n and outputs a key k; we write this as $k \leftarrow \mathsf{Gen}(1^n)$ (thus emphasizing the fact that Gen is a randomized algorithm). We will assume without loss of generality that any key k output by $\mathsf{Gen}(1^n)$ satisfies $|k| \geq n$.*
2. *The encryption algorithm Enc takes as input a key k and a plaintext message $m \in \{0,1\}^*$, and outputs a ciphertext c.[5] Since Enc may be randomized, we write this as $c \leftarrow \mathsf{Enc}_k(m)$.*
3. *The decryption algorithm Dec takes as input a key k and a ciphertext c, and outputs a message m. We assume without loss of generality that Dec is deterministic, and so write this as $m := \mathsf{Dec}_k(c)$.*

It is required that for every n, every key k output by $\mathsf{Gen}(1^n)$, and every $m \in \{0,1\}^*$, it holds that $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$.

Here, we keep the experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$ almost exactly the same (except for some technical differences discussed below), but introduce two key modifications in the definition itself:

1. We now consider only adversaries running in *polynomial time*, whereas Definition 2.4 considered even all-powerful adversaries.
2. We now concede that the adversary might determine the encrypted message with probability *negligibly better than* $1/2$.

Defining the "break" for encryption is not trivial, but we have already discussed this issue at length in Section 1.4.1 and the previous chapter. We therefore just recall that the idea behind the definition is that the adversary should be unable to learn *any partial information* about the plaintext from the ciphertext. The definition of *semantic security* directly formalizes exactly this notion, and was the first definition of security for encryption to be proposed. Unfortunately, the definition of semantic security is complex and difficult to work with, and we will not present it in this book. Fortunately, there is an equivalent definition in terms of *indistinguishability* which is somewhat simpler. Since the definitions are equivalent, we can work with the simpler definition of indistinguishability while being convinced that the security guarantees we obtain are those we expect from semantic security.

The adversarial indistinguishability experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$:

1. *The adversary $\mathcal{A}$ is given input 1^n, and outputs a pair of messages m_0, m_1 of the same length.*
2. *A random key k is generated by running $\mathsf{Gen}(1^n)$, and a random bit $b \leftarrow \{0,1\}$ is chosen. The ciphertext $c \leftarrow \mathsf{Enc}_k(m_b)$ is computed and given to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a bit b'.*
4. *The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise. If $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1$, we say that $\mathcal{A}$ succeeded.*

DEFINITION 3.9 *A private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has indistinguishable encryptions in the presence of an eavesdropper if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl such that*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n),$$

where the probability is taken over the random coins used by $\mathcal{A}$, as well as the random coins used in the experiment (for choosing the key, the random bit b, and any random coins used in the encryption process).

DEFINITION 3.13 *A private-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is semantically secure in the presence of an eavesdropper if for every probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a probabilistic polynomial-time algorithm $\mathcal{A}'$ such that for every efficiently-sampleable distribution $X = (X_1, \ldots)$ and all polynomial-time computable functions f and h, there exists a negligible function negl such that*

$$\left|\Pr[\mathcal{A}(1^n, \mathsf{Enc}_k(m), h(m)) = f(m)] - \Pr[\mathcal{A}'(1^n, h(m)) = f(m)]\right| \leq \mathsf{negl}(n),$$

where m is chosen according to distribution X_n, and the probabilities are taken over the choice of m and the key k, and any random coins used by $\mathcal{A}$, $\mathcal{A}'$, and the encryption process.

THEOREM 3.14 *A private-key encryption scheme has indistinguishable encryptions in the presence of an eavesdropper if and only if it is semantically secure in the presence of an eavesdropper.*

Pseudorandom generators. We now proceed to formally define the notion of a pseudorandom generator. Informally, as discussed above, a distribution $\mathcal{D}$

DEFINITION 3.15 *Let $\ell(\cdot)$ be a polynomial and let G be a deterministic polynomial-time algorithm such that upon any input $s \in \{0,1\}^n$, algorithm G outputs a string of length $\ell(n)$. We say that G is a pseudorandom generator if the following two conditions hold:*

1. Expansion: *For every n it holds that $\ell(n) > n$.*
2. Pseudorandomness: *For all probabilistic polynomial-time distinguishers D, there exists a negligible function negl such that:*

$$\left|\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]\right| \leq \mathsf{negl}(n),$$

where r is chosen uniformly at random from $\{0,1\}^{\ell(n)}$, the seed s is chosen uniformly at random from $\{0,1\}^n$, and the probabilities are taken over the random coins used by D and the choice of r and s.

The function $\ell(\cdot)$ is called the expansion factor of G.

this belief is based on the fact that they can be constructed (in a provable sense) under the rather weak assumption that *one-way functions* exist. This will be discussed in greater detail in Chapter 6. For now, suffices it to say that

CPA Security

The CPA indistinguishability experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$:

1. *A random key k is generated by running $\mathsf{Gen}(1^n)$.*
2. *The adversary $\mathcal{A}$ is given input 1^n and oracle access to $\mathsf{Enc}_k(\cdot)$, and outputs a pair of messages m_0, m_1 of the same length.*
3. *A random bit $b \leftarrow \{0,1\}$ is chosen, and then a ciphertext $c \leftarrow \mathsf{Enc}_k(m_b)$ is computed and given to $\mathcal{A}$. We call c the challenge ciphertext.*
4. *The adversary $\mathcal{A}$ continues to have oracle access to $\mathsf{Enc}_k(\cdot)$, and outputs a bit b'.*
5. *The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise. (In case $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1$, we say that $\mathcal{A}$ succeeded.)*

DEFINITION 3.22 *A private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has indistinguishable encryptions under a chosen-plaintext attack (or is CPA-secure) if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl such that*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n),$$

where the probability is taken over the random coins used by $\mathcal{A}$, as well as the random coins used in the experiment.

CONSTRUCTION 3.16

Let G be a pseudorandom generator with expansion factor ℓ. Define a private-key encryption scheme for messages of length ℓ as follows:

- Gen: on input 1^n, choose $k \leftarrow \{0,1\}^n$ uniformly at random and output it as the key.
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^{\ell(n)}$, output the ciphertext
$$c := G(k) \oplus m.$$
- Dec: on input a key $k \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{\ell(n)}$, output the plaintext message
$$m := G(k) \oplus c.$$

THEOREM 3.17 *If G is a pseudorandom generator, then Construction 3.16 is a private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper.*

DEFINITION 3.24 *Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient, length-preserving, keyed function. We say F is a pseudorandom function if for all probabilistic polynomial-time distinguishers D, there exists a negligible*

At first sight, it may appear that Definition 3.22 is impossible to achieve. In particular, consider an adversary that outputs (m_0, m_1) and then receives $c \leftarrow \mathsf{Enc}_k(m_b)$. Since the adversary has oracle access to $\mathsf{Enc}_k(\cdot)$, it can request that this oracle encrypt the messages m_0 and m_1 and $\mathcal{A}$ can thus obtain $c_0 \leftarrow \mathsf{Enc}_k(m_0)$ and $c_1 \leftarrow \mathsf{Enc}_k(m_1)$. It can then compare c_0 and c_1 to its "challenge ciphertext" c; if $c = c_0$ then it knows that $b = 0$, and if $c = c_1$ then it knows that $b = 1$. Why doesn't this strategy allow $\mathcal{A}$ to determine b with probability 1?

The answer is that indeed, as with security under multiple encryptions, no *deterministic* encryption scheme can be secure against chosen-plaintext attacks. Rather, any CPA-secure encryption scheme must be probabilistic.

CCA

Consider the following experiment for any private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, adversary $\mathcal{A}$, and value n for the security parameter.

The CCA indistinguishability experiment $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$**:**

1. *A random key k is generated by running $\mathsf{Gen}(1^n)$.*
2. *The adversary $\mathcal{A}$ is given input 1^n and oracle access to $\mathsf{Enc}_k(\cdot)$ and $\mathsf{Dec}_k(\cdot)$. It outputs a pair of messages m_0, m_1 of the same length.*
3. *A random bit $b \leftarrow \{0,1\}$ is chosen, and then a ciphertext $c \leftarrow \mathsf{Enc}_k(m_b)$ is computed and given to $\mathcal{A}$. We call c the* challenge ciphertext.
4. *The adversary $\mathcal{A}$ continues to have oracle access to $\mathsf{Enc}_k(\cdot)$ and $\mathsf{Dec}_k(\cdot)$, but is not allowed to query the latter on the challenge ciphertext itself. Eventually, $\mathcal{A}$ outputs a bit b'.*
5. *The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.*

DEFINITION 3.31 *A private-key encryption scheme Π has* indistinguishable encryptions under a chosen-ciphertext attack *(or is* CCA-secure*) if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function* negl *such that:*

$$\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n),$$

where the probability is taken over all random coins used in the experiment.

DEFINITION 3.24 *Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient, length-preserving, keyed function. We say F is a* pseudorandom function *if for all probabilistic polynomial-time distinguishers D, there exists a negligible function* negl *such that:*

$$\left|\Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f_n(\cdot)}(1^n) = 1]\right| \leq \mathsf{negl}(n),$$

where $k \leftarrow \{0,1\}^n$ is chosen uniformly at random and f_n is chosen uniformly at random from the set of functions mapping n-bit strings to n-bit strings.

CONSTRUCTION 3.25

Let F be a pseudorandom function. Define a private-key encryption scheme for messages of length n as follows:

- Gen: on input 1^n, choose $k \leftarrow \{0,1\}^n$ uniformly at random and output it as the key.
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, choose $r \leftarrow \{0,1\}^n$ uniformly at random and output the ciphertext

$$c := \langle r,\ F_k(r) \oplus m \rangle.$$

- Dec: on input a key $k \in \{0,1\}^n$ and a ciphertext $c = \langle r, s \rangle$, output the plaintext message

$$m := F_k(r) \oplus s.$$

THEOREM 3.26 *If F is a pseudorandom function, then Construction 3.25 is a fixed-length private-key encryption scheme with length parameter $\ell(n) = n$ that has indistinguishable encryptions under a chosen-plaintext attack.*