CSE5014: Cryptography and Network Security

2024 Spring Semester Summary notes

One-sentence definition of *Cryptography.* “Cryptography is the practice and study of techniques for secure communication in the presence of third parties called adversaries.” – R.L. Rivest

Milestone papers:

[1] C.E. Shannon, Communication theory of secrecy systems, *The Bell system technical journal*, 28(4): 656-715, 1949.

[2] W. Diffie and M.E. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, 22(6): 29-40, 1976.

[3] R.L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, 21(2): 120-126, 1978.

and many others . . .

Three principles of modern cryptography:

1. **Formal definitions**: precise, mathematical model and definition of what security means.
2. **Precise assumptions**: should be clearly stated and unambiguous.
3. **Proofs of security**: make it possible to move away from “design-break-tweak”.

A typical cryptographic definition contains two parts: security guarantee/goal, and threat model. The former means what we want to achieve and/or what we want to prevent the attacker from achieving, and the latter means what (real-world) capabilities the attacher is assumed to have.

Definition 1. *A* private-key encryption *scheme is defined by a message space $\mathcal{M}$ and three algorithms (Gen, Enc, Dec):*

- *Gen (key-generation algorithm): generates the private key k;*
- *Enc (encryption algorithm): takes key k and message $m \in \mathcal{M}$ as input; outputs ciphertext c: $c \leftarrow Enc_k(m)$;*
- *Dec (decryption algorithm): takes key k and ciphertext c as input; outputs m': $m' = Dec_k(c)$.*

Probability review. A *random variable* (r.v.) is a variable that takes on (discrete) values with certain probabilities. For an r.v., a *probability distribution* is the distribution of probabilities that are specified to all possible values of the r.v. such that each probability must be between 0 and 1, and the probabilities must sum to 1. An *event* is a particular occurrence in some experiment, and $\Pr[E]$ denotes the probability that the event E occurs. The *conditional probability* is the probability that one event occurs given that some other event occurred, and the corresponding formula is

$$\Pr[A|B] = \Pr[A \text{ and } B]/\Pr[B].$$

Two r.v.'s X and Y are called *independent* if for all possible values x, y, it holds that

$$\Pr[X = x|Y = y] = \Pr[X = x].$$

By the formula for the conditional probability, an equivalent condition of independence of two r.v.'s is

$$\Pr[X = x \text{ and } Y = y] = \Pr[X = x] \cdot \Pr[Y = y].$$

Given that $E_1, E_2, \ldots, E_n$ are a partition of all possibilities. Then for any event A, it holds that

$$\Pr[A] = \sum_i \Pr[A \text{ and } E_i] = \sum_i \Pr[A|E_i] \cdot \Pr[E_i].$$

This is called the *law of total probability.* For a private-key encryption scheme, the key K can be viewed as the r.v. that ranges over the key space $\mathcal{K}$, and the plaintext M is the r.v. that ranges over the plaintext space $\mathcal{M}$. Note that the two random variables M and K are *independent*, i.e., the message that a party sends does NOT depend on the key used to encrypt the message.

Fix an encryption scheme (Gen, Enc, Dec), and some distribution for M. Consider the following randomized experiment:

1. Choose a message m according to the given distribution
2. Generate a key k using the algorithm Gen
3. Compute $c \leftarrow Enc_k(m)$

Then this defines a distribution on the ciphertext. Let C be an r.v. denoting the value of the ciphertext in this experiment. Thereby, the set of all possible values of the ciphertext constitutes the ciphertext space $\mathcal{C}$

Suppose that $k \in \{0,1\}^n$, $m \in \{0,1\}^\ell$, and $c \in \{0,1\}^L$. With this setting above, how can we define what it means for an encryption scheme

(Gen, Enc, Dec) over the plaintext space $\mathcal{M}$ to be *secure* against a (single) ciphertext-only attack?

Perfect secrecy. The definition of *perfect secrecy* is the following, which means that by observing the ciphertext, the attacker's knowledge about the distribution of M should not be changed.

Definition 2 (Definition 1.5)**.** *An encryption scheme (Gen, Enc, Dec) with the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$ is* perfectly secure *if for every distribution over $\mathcal{M}$, for every message $m \in \mathcal{M}$, and every $c \in \mathcal{C}$ with $\Pr[C = c] > 0$, it holds that*

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

Essentially, perfect secrecy means that the ciphertext c reveals *zero additional information* about the plaintext m. An equivalent version of the definition above is the following.

Definition 3 (Definition 1.5')**.** *For every set $M \subseteq \{0,1\}^\ell$ of plaintexts, and for every strategy used by Eve, if we choose at random $x \in M$ and a random key $k \in \{0,1\}^n$, then the probability that Eve guesses x correctly after seeing $Enc_k(x)$ is* at most $1/|M|$*, i.e.,*

$$\Pr[Eve(Enc_k(x)) = x] \leq 1/|M|.$$

There are another two equivalent versions of the definition of perfect secrecy.

Definition 4 (Definition 1.6)**.** *An encryption scheme (Gen, Enc, Dec) with the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$ is* perfectly secure *if and only if for every two distinct plaintexts $\{x_0, x_1\} \in \mathcal{M}$, and for every strategy used by Eve, if we choose at random $b \in \{0,1\}$ and a random key $k \in \{0,1\}^n$, then the probability that Eve guesses x_b correctly after seeing the ciphertext $c = Enc_k(x_b)$ is at most $1/2$.*

Definition 5 (Definition 1.7)**.** *An encryption scheme (Gen, Enc, Dec) is* perfectly secure *if and only if for every pair of plaintexts $x_0, x_1 \in \mathcal{M}$, it holds that*

$$Enc_{k \leftarrow_{U_n} \mathcal{K}}(x_0) \equiv Enc_{k \leftarrow_{U_n} \mathcal{K}}(x_1).$$

Recall that two probability distributions X, Y over $\{0,1\}^\ell$ are identical, denoted by $X \equiv Y$, if for every $y \in \{0,1\}^\ell$, it holds that $\Pr[X = y] = \Pr[Y = y]$.

The following theorem proves the equivalence of Definition 1.5' and Definition 1.6.

Theorem 1 (Theorem 1.8)**.** *An encryption scheme* (Gen, Enc, Dec) *is* perfectly secure *if and only if for each* $b \in \{0, 1\}$,

$$\Pr[Eve(Enc_k(x_b)) = x_b] \leq 1/2.$$

Proof. "only if" part: this is the special case that $|M| = 2$ in Definition 1.5'.

"if" part: Suppose that the encryption scheme (Gen, Enc, Dec) is NOT perfectly secure, by Definition 1.5', this means that there exists some set M and some strategy for Eve to guess a plaintext chosen from M with probability larger than $1/|M|$. We need prove that there is also some set M' of size 2, and there exists some strategy Eve' for Eve to guess a plaintext chosen from M' with probability larger than $1/2$.

Fix $x_0 = 0^\ell$ and pick x at random from M. Then it holds that for random key k and message $x_1 \in M$,

$$\Pr_{k \leftarrow \{0,1\}^n,\ x \leftarrow M}[Eve(Enc_k(x)) = x] \geq 1/|M|.$$

On the other hand, for very choice of k, $x' = Eve(Enc_k(x_0))$ is a fixed string independent on the choice of x. So if we pick x at random from M, the probability that $x = x'$ is at most $1/|M|$, i.e.,

$$\Pr_{k \leftarrow \{0,1\}^n,\ x \leftarrow M}[Eve(Enc_k(x_0)) = x] \leq 1/|M|.$$

Due to the linearity of expectation, there exists some x_1 satisfying

$$\Pr[Eve(Enc_k(x_1)) = x_1] > \Pr[Eve(Enc_k(x_0)) = x_1]. \text{ (Why?)}$$

We now define a new attacker Eve' as :

$$Eve'(c) = \begin{cases} x_1, & \text{if } Eve(c) = x_1, \\ x_i,\ i \in \{0,1\} \text{ at random}, & \text{otherwise.} \end{cases}$$

Then the probability that $Eve'(Enc_k(x_b)) = x_b$ is larger than $1/2$ (Why?). □

One-time Pad. The XOR operation is defined as the addition mod 2, i.e., $a \oplus b = a + b \bmod 2$. For a message x and a key k with $n = |k| = |x|$, the encryption algorithm $Enc : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is $Enc_k(x) = x \oplus k$, and the decryption algorithm is $Dec_k(y) = y \oplus k$, where $\oplus$ denotes the bitwise XOR operation.

Theorem 2 (Theorem 1.9)**.** *One-time pad is* perfectly secure*.*

Proof. By Definition 1.7, it suffices to prove that for every $x \in \{0,1\}^n$, the distribution $Y_x = Enc_{U_n}(x)$ is uniformly distributed.

Let $y \in \{0,1\}^n$, we need show that

$$\Pr_{k \leftarrow_R \{0,1\}^n}[x \oplus k - y] = 2^{-n}.$$

Since there is a unique single value of $k = x \oplus y$, the probability that the equation is true is 2^{-n}. $\square$

Proof. [Alternative proof] By Definition 1.5, it suffices to prove $\Pr[M = m|C = c] = \Pr[C = c]$ for arbitrary distribution over $\mathcal{M} = \{0,1\}^n$ and arbitrary $m, c \in \{0,1\}^n$.

First, by Law of total probability, we have

$$\begin{aligned}
&\Pr[C = c] \\
&= \sum_{m'} \Pr[C = c|M = m'] \cdot \Pr[M = m'] \\
&= \sum_{m'} \Pr[K = m' \oplus c] \cdot \Pr[M = m'] \\
&= \sum_{m'} 2^{-n} \cdot \Pr[M = m'] \\
&= 2^{-n}.
\end{aligned}$$

Then we have

$$\begin{aligned}
&\Pr[M = m|C = c] \\
&= \Pr[C = c|M = m] \cdot \Pr[M = m]/\Pr[C = c] \\
&= \Pr[K = m \oplus c] \cdots \Pr[M = m]/2^{-n} \\
&= 2^{-n} \cdot \Pr[M = m]/2^{-n} \\
&= \Pr[M = m].
\end{aligned}$$

Therefore, one-time pad is perfectly secure. $\square$

Now we have a "clean" definition of security, together with a construction of encryption scheme that is perfectly secure. However, this is not the end of cryptography, since several limitations exist: the key length is as the same as the length of plaintexts; one-time pad leaks information if the same key is used for multiple plaintexts; one-time pad can be easily broken by a known-plaintext attack... The following theorem states the first drawback in terms of key length, i.e., for perfectly secure encryption schemes, the key length cannot be shorter than the length of plaintexts by even one bit.

Theorem 3 (Theorem 1.10)**.** *There is* no *perfectly secure encryption schemes* (Gen, Enc, Dec) *with n-bit plaintexts and $(n-1)$-bit keys.*

Proof. Suppose that (Gen, Enc, Dec) is such an encryption scheme with n-bit plaintexts and $(n-1)$-bit keys. Denote by Y_0 the distribution of $Enc_{U_{n-1}}(0^n)$ and by S_0 the support of Y_0.

Since there are only 2^{n-1} possible keys, we have $|S_0| \leq 2^{n-1}$. Now for every key k the function $Enc_k(\cdot)$ is a one-to-one function and hence the image size of the ciphertexts is $\geq 2^n$. This means that for every k, there must exist an plaintext x such that $Enc_k(x) \notin S_0$. Fix such a key k and the plaintext x, then the distribution $E_{U_{n-1}}(x)$ does not have the same support as Y_0. Therefore, it is not identical to Y_0.

$\square$

Statistical security. To overcome the limitations of perfect secrecy, it is a good idea to examine whether we can relax these assumptions. Namely, we may say that an encryption scheme is ϵ-statistically secure if the probability that Eves guesses correctly which of the two messages was encrypted is at most $1/2 + \epsilon$, with a tiny advantage ϵ.

Definition 6 (Definition 2.1)**.** *Let X and Y be two distributions over $\{0,1\}^n$. The* statistical distance *of X and Y, denoted by $\Delta(X,Y)$ is defined to be*

$$\max_{T \subseteq \{0,1\}^n} |\Pr[X \in T] - \Pr[Y \in T]|.$$

If $\Delta(X,Y) \leq \epsilon$, we denote that $X \equiv_\epsilon Y$.

The following lemma give a systematic way to determine the statistical distance between two distributions.

Lemma 4 (Lemma 2.3)**.**

$$\Delta(X,Y) = \frac{1}{2} \sum_{w \in Supp(X) \cup Supp(Y)} |\Pr[X = w] - \Pr[Y = w]|,$$

where $Supp(X)$ denotes the support of the distribution X.

Proof. For every set $T \subseteq \{0,1\}^n$, define

$$\Delta_T(X,Y) = |\Pr[X \in T] - \Pr[Y \in T]|.$$

Then by Definition 2.1, we have $\Delta(X,Y) = \max_{T\subseteq\{0,1\}^n} \Delta_T(X,Y)$. Note that since $\Pr[X \in T^c] = 1 - \Pr[X \in T]$, we have $\Delta_{T^c}(X,Y) = \Delta_T(X,Y)$. Let $T = \{w : \Pr[X = w] > \text{Y = w}\}$, then we have

$$\begin{aligned}
&\frac{1}{2} \sum_{w\in Supp(X)\cup Supp(Y)} |\Pr[X = w] - \Pr[Y = w]| \\
&= \frac{1}{2}\sum_{w\in T}(\Pr[X = w] - \Pr[Y = w]) + \frac{1}{2}(\Pr[Y = w] - \Pr[X = w]) \\
&= \frac{1}{2}(\Delta_T(X,Y) + \Delta_{T^c}(X,Y)) \\
&= \Delta_T(X,Y) \\
&\leq \Delta(X,Y). \qquad (*)
\end{aligned}$$

On the other hand, let S be the set achieving the maximum of $\Delta_S(X,Y)$, i.e., $\Delta(X,Y) = \Delta_S(X,Y)$. Without loss of generality, assume that $\Pr[X \in S] \geq \Pr[Y \in S]$ (otherwise, take the complement of S). Then we have

$$\begin{aligned}
2\Delta(X,Y) &= \Delta_S(X,Y) + \Delta_{S^c}(X,Y) \\
&= \Pr[X \in S] - \Pr[Y \in S] + \Pr[Y \in S^c] - \Pr[X \in S^c] \\
&= \sum_{w\in S}(\Pr[X = w] - \Pr[Y = w]) + \sum_{w\in S^c}(\Pr[Y = w] - \Pr[X = w]) \\
&\leq \sum_{w\in S}|\Pr[X = w] - \Pr[Y = w]| + \sum_{w\in S^c}|\Pr[Y = w] - \Pr[X = w]| \\
&= \sum_{w}|\Pr[X = w] - \Pr[Y = w]|. \qquad (**)
\end{aligned}$$

Therefore, by (*) and (**), the conclusion is proved. $\square$

Note that it is clear that $0 \leq \Delta(X,Y) \leq 1$ and $\Delta(X,Y) = 0$ if $X = Y$. One may try proving that $0 \leq \Delta(X,Y) \leq \Delta(X,Z) + \Delta(Z,Y)$, and further prove that the statistical distance is a *metric.*

Definition 7 (Definition 2.2 ϵ-statistical security)**.** *An encryption scheme* (Gen, Enc, Dec) *is* ϵ-statically secure *if and only if for every pair of plaintexts* m, m'*, it holds that* $Enc_{U_n}(m) \equiv_\epsilon Enc_{U_n}(m')$*.*

Lemma 5 (Lemma 2.4)**.** *Eve has at most* $1/2 + \epsilon$ *success probability if and only if for every pair of* m_1, m_2*, it holds that*

$$\Delta(Enc_{U_n}(m_1), Enc_{U_n}(m_2)) \leq 2\epsilon.$$

Proof. Suppose that Eve has $1/2 + \epsilon$ success probability with the two messages m_1, m_2. Let $p_{i,j} = \Pr[Eve(Enc_{U_n}(m_i)) = j]$. Then we have

$$\begin{aligned} p_{1,1} + p_{1,2} &= 1 \\ p_{2,1} + p_{2,2} &= 1 \\ (1/2)p_{1,1} + (1/2)p_{2,2} &\leq 1/2 + \epsilon. \end{aligned}$$

The last two together imply that

$$p_{1,1} - p_{2,1} \leq 2\epsilon,$$

which means that if we let T be the set $\{c : \ Eve(c) = 1\}$, then T demonstrates that $\Delta(Enc_{U_n}(m_1), Enc_{U_n}(m_2)) \leq 2\epsilon$.

Similarly, if we have such a set T, we can define an attacker from it that succeeds with probability $1/2 + \epsilon$.

□

Similar to Theorem 1.10, the following result demonstrates that the limitation still exists if we consider ϵ-statistical security instead of perfect security.

Theorem 6 (Theorem 2.5)**.** *Let (Gen, Enc, Dec) be a valid encryption scheme with the encryption algorithm $Enc : \{0,1\}^n \times \{0,1\}^{n+1} \to \{0,1\}^*$. Then there exist plaintexts m_1, m_2 such that $\Delta(Enc_{U_n}(m_1), Enc_{U_n}(m_2)) > 1/2$.*

Proof. **Fact**. For a r.v. Y, if $E[Y] \leq \mu$, then $\Pr[Y \leq \mu] > 0$.

Let $m_1 = 0^{n+1}$, and let $S = Supp(Enc_{U_n}(m_1))$, then $|S| \leq 2^n$.

We choose a random message $m \leftarrow_R \{0,1\}^{n+1}$ and define the following 2^n random variables for every k:

$$T_k(m) = \begin{cases} 1, & \text{if } Enc_k(m) \in S, \\ 0, & \text{otherwise.} \end{cases}$$

Since for every k, the encryption function $Enc_k(\cdot)$ is a one-to-one function, we have $\Pr[T_k = 1] \leq 1/2$. Define $T = \sum_{k \in \{0,1\}^n} T_k$, then

$$E[T] = E[\sum_k T_k] = \sum_k E[T_k] \leq 2^n/2.$$

This means that the probability $\Pr[T \leq 2^n/2] > 0$. In other words, there exists an m such that $\sum_k T_k(m) \leq 2^n/2$. For such an m, at most half of the keys k satisfy $Enc_K(m) \in S$, i.e.,

$$\Pr[Enc_{U_n}(m) \in S] \leq 1/2.$$

Since $\Pr[Enc_{U_n}(0^{n+1}) \in S] = 1$, we then have

$$\Delta(Enc_{U_n}(0^{n+1}), Enc_{U_n}(m)) \geq 1/2.$$

$\square$

Now we consider a standard form of security definition: we formulate an experiment on the basis of Definition 1.6, which will be used similarly afterwards in the definitions of security.

Let $\Pi = (Gen, Enc, Dec)$ be an encryption scheme with message space $\mathcal{M}$, and A an adversary. Define a *randomized* experiment $PrivK_{A,\Pi}$:

1. A outputs $m_0, m_1 \in \mathcal{M}$
2. $k \leftarrow Gen$, $b \leftarrow \{0,1\}$, $c \leftarrow Enc_k(m_b)$
3. $b' \leftarrow A(c)$

Adversary A succeeds if $b = b'$, and we say the experiment *evaluates to* 1 ($PrivK_{A,\Pi} = 1$) in this case.

If we define Π is *perfectly indistinguishable* if for *all* attackers (algorithm) A, it holds that

$$\Pr[PrivK_{A,\Pi} = 1] \leq 1/2.$$

Then we have exactly the definition of perfect security in the form of $PrivK_{A,\Pi}$.
Claim. Π is *perfectly indistinguishable* if and only if Π is *perfectly secure*.

Now that *statistical security* does not allow us to break the impossibility result, namely, the key length should be the same as the length of plaintexts. The idea is that it would be OK if an ecntryption scheme leaked information with *tiny probability* to eavesdroppers with *bounded computational resources*. More precisely, it is allowed that security may fail with tiny probability, and only "efficient" attackers exist. There are two approaches: concrete security and asymptotic security. For concrete (t, ϵ)-indistinguishability, two concrete parameters are involved: security may fail with probability $\leq \epsilon$, and we restrict attention to attackers running in time $\leq t$. However, this concrete setting does *not* lead to a clean theory: this may be sensitive to exact computational model, and Π can be (t, ϵ)-secure for many choices of the parameters t, ϵ.

We focus on asymptotic security, with respect to a *security parameter* n. For now, we view n as the key length, which may be fixed by honest parties at initialization, and assumebly be known by adversary. Then the running time of all parties, and the success probability of the adversary, are both measured by functions in terms of n. For the sake of defining *computational indistinguishability*, it is required that security may fail with probability *negligible* in n, and attackers are assumbed to be running in time (at most) *polynomial* in n.

A function $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ is (at most) *polynomial* if there exists an integer c such that $f(n) < n^c$ for large enough n. A function $f : \mathbb{Z}^+ \to [0, 1]$ is *negligible* if every polynnomial p it holds that $f(n) < 1/p(n)$ for large enough n. A typical example of negligible functions if $f(n) = poly(n) \cdot 2^{-cn}$.

We borrow the concept of "efficient" algorithms from complexity theory as "(probabilistic) polynomial-time (PPT)" algorithms. Two convenient closure properties are useful: $poly * poly = poly$ (poly-many calls to PPT subroutine is still PPT), and $poly * negl = negl$ (poly-many calls to subroutine that fails with negligible probability fails with negligible probability overall).

Computational security. Now we are ready to define *computational indistinguishability* in terms of the experiment $PrivK_{A,\Pi}(n)$. Let $\Pi = (Gen, Enc, Dec)$ be an encryption scheme with message space $\mathcal{M}$ and A an adversary. Define a randomized experiment $PrivK_{A,\Pi}(n)$:

1. $A(1^n)$ outputs $m_0, m_1 \in \{0, 1\}^*$ of equal length
2. $k \leftarrow Gen(1^n)$, $b \leftarrow \{0, 1\}$, $c \leftarrow Enc_k(m_b)$
3. $b' \leftarrow A(c)$

Adversary A succeeds if $b = b'$, and we say the experiment evaluates to 1 ($PrivK_{A,\Pi}(n) = 1$) in this case.

Definition 8 (Definition 3.1)**.** Π *is* computationally indistinguishable *(aka* EAV-secure*) if for all PPT attackers (algorithms) A, there is a negligible function ϵ such that*

$$\Pr[PrivK_{A,\Pi}(n) = 1] \leq 1/2 + \epsilon(n).$$

Example. Consider a scheme where the best attack is brute-force search over the key space (thus the attack algorithm runs in $O(2^n)$), and $Gen(1^n)$ generates a uniform n-bit key. Thus, if A runs in polynomial time $t(n)$, then we have $\Pr[PrivK_{A,\Pi}(n) = 1] \leq 1/2 + O(t(n)/2^n)$. The scheme is EAV-secure: for any polynomial t, the function $t(n)/2^n$ is negligible.

Example. Consider a scheme and a particular attacker A that runs for n^3 minutes and breaks the scheme with probability $2^{40}2^{-n}$. This does not contradict asymptotic security since $2^{40}2^{-n}$ is a negligible function in n. For

$n = 40$, A breaks with probability 1 in approximately 6 weeks; $n = 50$, A breaks with probability 1/1000 in approximately 3 months; $n = 500$, A breaks with probability 2^{-500} in about 200 years. This explains why computational indistinguishability may lead to real-world security with parameters properly chosen.

PRG. To build an encryption scheme, we introduce a new cryptographic primitive, *pseudorandom generator* (PRG). To this end, we define pseudorandomness asymptotically.

Definition 9 (Definition 3.2)**.** *Let D_n be a ditribution over $p(n)$-bit strings. $\{D_n\}$ is* pseudorandom *if for all PPT distinguishers A, there is a negligible function ϵ such that*

$$|\Pr_{x \leftarrow D_n}[A(x) = 1] - \Pr_{x \leftarrow U_{p(n)}}[A(x) = 1]| \leq \epsilon(n),$$

where $U_{p(n)}$ denotes the uniform distributio on $p(n)$-bit strings.

Definition. A *PRG* is an efficient, deterministic algorithm that expands a short, *uniform* seed into a longer, *pseudorandom* output. Let G be a deterministic, poly-time algorithm that expands the length of input, i.e., $|G(x)| = p(|x|) > |x|$. For all efficient distinguishers A, there is a negligible function ϵ such that

$$|\Pr_{x \leftarrow U_n}[A(G(x)) = 1] - \Pr_{y \leftarrow U_{p(n)}}[A(y) = 1]| \leq \epsilon(n),$$

In other words, no efficient A can distinguish whether it is given $G(x)$ for a uniform x or a uniform string y.

Now we have an encryption scheme, i.e., *pseudo one-time pad*, which can be proved to be EAV-secure.

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

Theorem 7 (Theorem 3.3)**.** *If G is a pseudorandom generator (PRG), then the pseudo one-time pad (pseudo-OTP) Π is* EAV-secure.

Proof. We use reduction to prove the result, and the proof idea is: by assuming that there is an efficient attacker A who "breaks" the pseudo-OTP scheme (this means that it is not EAV-secure), we use A as a subroutine to build an efficient D that "breaks" *pseudorandomness* of G. However, by the assumption that G is a PRG, no such a D exists. Thus, no such an A can exist.

Suppose to the contrary that there exists an efficient attacker A such that

$$\Pr[PrivK_{A,\Pi}(n) = 1] > 1/2 + 1/poly(n).$$

(Note that Π is then not EAV-secure.) This means that

$$\Pr[A(Enc_{U_n}(m)) = 1] - \Pr[A(U_{p(n)}) = 1] > 1/poly(n),$$

which further implies that

$$|\Pr[A(G(U_n) \oplus m) = 1] - \Pr[A(U_{p(n)}) = 1]| > 1/poly(n).$$

We define a distinguisher $D : \{0,1\}^{p(n)} \to \{0,1\}$ as $D(y) = A(y \oplus m)$, which means $A(z) = D(z \oplus m)$. Note that D is also efficient. But we have

$$|\Pr[D(G(U_n)) = 1] - \Pr[D(U_{p(n)} \oplus m) = 1]| > 1/poly(n).$$

Since $U_{p(n)} \oplus m \equiv U_{p(n)}$, this contradicts the assumption that G is a PRG.

We may have an alternative form of proof by reduction. The idea is: by assuming that G is a PRG, we fix some arbitrary efficient A attacking the pseudo-OTP scheme, and then use A as a subroutine to build an efficient D attacking G. This thereby relates the distinguishing probability of D to the attacking success probability of A. By assumption of PRG, the distinguishing probability of D must be negligible, and this therefore bounds the success probability of A.

Let $\mu(n) = \Pr[PrivK_{A,\Pi}(n) = 1]$. If the distribution of $y = G(x)$ is pseudorandom, then the view of A is exactly the same as in $PrivK_{A,\Pi}(n)$, i.e.,

$$\Pr_{x \leftarrow U_n}[D(G(x)) = 1] = \mu(n).$$

If the distribution of y is uniform, then A succeeds with probability exactly 1/2, i.e.,

$$\Pr_{y \leftarrow U_{p(n)}}[D(y) = 1] = 1/2.$$

Since G is a PRG, we have

$$|\mu(n) - 1/2| \leq negl(n),$$

and it then follows that

$$\Pr[PrivK_{A,\Pi}(n) = 1] \leq 1/2 + negl(n).$$

Therefore, the encryption scheme Π is EAV-secure.

$\square$

By Theorem 3.3, the pseudo-OTP is EAV-secure, and has a key shorter than the message, but still has the second limitation that key can only be used once. We will keep the security goal the same, but strengthen the threat model, by defining multiple-message indistinguishability.

Multi-message indistinguishability. Let $\Pi = (Gen, Enc, Dec)$ be an encryption scheme and A an adversary. Define a randomized experiment $PrivK^{mult}_{A,\Pi}(n)$:

1. $A(1^n)$ outputs two vectors $(m_{0,1}, \ldots, m_{0,t})$ and $(m_{1,1}, \ldots, m_{1,t})$. It is required that $|m_{0,i}| = |m_{1,i}|$ for all i.
2. $k \leftarrow Gen(1^n)$, $b \leftarrow \{0,1\}$, and for all i, $c_i \leftarrow Enc_k(m_{b,i})$.
3. $b' \leftarrow A(c_1, \ldots, c_t)$.

Adversary A succeeds if $b = b'$, and the experiment evaluates to 1 in this case.

Definition 10 (Definition 3.4)**.** Π *is* multiple-message indistinguishable *if for all PPT attackers A, there is a negligible function ϵ such that*

$$\Pr[PrivK^{mult}_{A,\Pi}(n) = 1] \leq 1/2 + \epsilon(n).$$

Question: Show that the pseudo-OTP scheme is not multiple-message indistinguishable.

CPA-security. Instead of working with multiple-message security, we define a stronger notion of *CPA-security*, i.e., security against chosen-plaintext attacks. This is nowadays the minimal notion of security that an encryption scheme should satisfy. In practice, there are many ways an attacker can influence what gets encrypted, and chosen-plaintext attacks encompass such influences.

Let $\Pi = (Gen, Enc, Dec)$ be an encryption scheme and A an adversary. Define a randomized experiment $PrivKCPA_{A,\Pi}(n)$:

1. $k \leftarrow Gen(1^n)$.
2. $A(1^n)$ interacts with an *encryption oracle* $Enc_k(\cdot)$, and then outputs m_0, m_1 of the same length.
3. $b \leftarrow \{0,1\}$, $c \leftarrow Enc_k(m_b)$, give c to A.

4. A can continue to interact with $Enc_k(\cdot)$.

5. A outputs b'.

Adversary A succeeds if $b = b'$, and the experiment evaluates to 1 in this case.

Definition 11 (Definition 4.1)**.** Π *is* secure against chosen-plaintext attacks *(CPA-secure) if for all PPT attackers A, there is a negligible function ϵ such that*

$$\Pr[PrivKCPA_{A,\Pi}(n) = 1] \leq 1/2 + \epsilon(n).$$

At the first glance, this definition of security seems impossible. Consider the following attacker A: use a chosen-plaintext attack to get $c_0 = Enc_k(m_0)$ and $c_1 = Enc_k(m_1)$. Output m_0, m_1, and get the challenge ciphertext c. If $c = c_0$, output 0, and output 1 if $c = c_1$. Then A succeeds with probability 1?

Note that this attack only works if encryption is deterministic. Thus, to achieve CPA-security, randomized encryption must be used! To build an encryption scheme that is CPA-secure, we introduce another cryptographic primitive called *pseudorandom function* (PRF).

PRF/PRP. Informally, a *pseudorandom function* looks like a random function uniformly picked from the set $Func_n$ of all functions from $\{0,1\}^n$ to $\{0,1\}^n$. By a counting argument, $|Func_n| = 2^{n \cdot 2^n}$. Uniformly choosing a function $f \in Func_n$ is equivalent to choosing $f(x)$ uniformly in $\{0,1\}^n$ for each $x \in \{0,1\}^n$, i.e., filling up the function table with uniform bits for the $n \cdot 2^n$ blanks. It does not make sense to talk about any fixed function being *pseudorandom*. We look instead at keyed functions: by choosing uniformly the key, the keyed function looks like random.

Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient deterministic algorithm. Define $F_k(x) = F(k,x)$, where the first input is called the *key*. Then choosing a uniform $f \in \{0,1\}^n$ is equivalent to choosing the function $F_k : \{0,1\}^n \to \{0,1\}^n$, i.e., the algorithm F defines a distribution over functions in $Func_n$. The number of functions in $\{F_k\}_{k \in \{0,1\}^n}$ is at most 2^n.

Definition 12 (Definition 4.2)**.** *F is a* pseudorandom function *(PRF) if F_k, for uniform $k \in \{0,1\}^n$ is* indistinguishable *from a uniform function $f \in Func_n$. Formally, for all PPT distinguishers D, it holds that*

$$|\Pr_{k \leftarrow \{0,1\}^n}[D^{F_k(\cdot)}(1^n) = 1] - \Pr_{f \leftarrow Func_n}[D^{f(\cdot)}(1^n) = 1]| \leq \epsilon(n).$$

A relevant cryptographic primitive is *pseudorandom permutation* (PRP), which defines a keyed function F_k that is indistinguishable from a random permutation from $f \in Perm_n \subset Func_n$. F is a keyed permutation if F_k is a permutation for every k and F_k^{-1} is efficiently computable, where $F_k^{-1}(F_k(x)) = x$.

Definition 13 (Definition 4.3). *F is a* pseudorandom permutation *(PRP) if F_k, for uniform key $k \in \{0,1\}^n$, is indistinguishable from a uniform permutation $f \in Perm_n$.*

Since for large enough n, a random permutation is indistinguishable from a random function, in practice, PRPs are also good PRFs. In fact, bock ciphers are considered as PRPs. PRFs/PRPs are viewed as a stronger primitive than PRGs, since a PRF/PRP can be viewed as a PRG with random access to exponentially long output: $F_k = F_k(0\dots0)|\dots|F_k(1\dots1)$, which has an output of length $n \cdot 2^n$ bits. On the other hand, a PRF/PRP F immediately implies a PRG G: simply define $G(k) = F_k(0\dots0)|F_k(0\dots1)$, or $G(k) = F_k(\langle 0 \rangle)|F_k(\langle 1 \rangle)|F_k(\langle 2 \rangle)\dots$ as needed, where $\langle i \rangle$ denotes the n-bit encoding of i.

By employing PRFs/PRPs, we are able to build an encryption scheme which can be prove to be CPA-secure. Let F be a length-preserving, keyed function.
$Gen(1^n)$: choose a uniform key $k \in \{0,1\}^n$.
$Enc_k(m)$: for $|m| = |k|$, choose a uniform $r \in \{0,1\}^n$ (called *nonce* or *initialization vector*), and output the ciphertext $c = \langle c_1, c_2 \rangle = \langle r, F_k(r) \oplus m \rangle$.
$Dec_k(c)$: output $c_2 \oplus F_k(c_1)$.

Theorem 8 (Theorem 5.1). *If F is a pseudorandom function (PRF), then this encryption scheme is CPA-secure.*

Proof. Let $\widetilde{\Pi} = (\widetilde{Gen}, \widetilde{Enc}, \widetilde{Dec})$ denote the encryption scheme using $f \in Func_n$, a random function, and $\Pi = (Gen, Enc, Dec)$ denote the encryption scheme using F_k, a PRF. Fix an arbitrary PPT adversary A and let $q(n)$ be the number of queries that $A(1^n)$ makes to its encryption oracle $Enc_k(\cdot)$ (note that $q(n)$ is bounded by some polynomial).

Step 1: prove that

$$|\Pr[PrivKCPA_{A,\Pi}(n) = 1] - \Pr[PrivKCPA_{A,\widetilde{\Pi}}(n) = 1]| \le negl(n). \qquad (*)$$

We prove this by reduction, i.e., we use A to construct a distinguisher D for the function F, whether F is "pseudorandom" (equal to F_k for a random

$k \in \{0,1\}^n$) or "random" (equal to f for a random $f \in Func_n$) and then demonstrate that A succeeds implies that D succeeds.
Distinguisher D:
D is given input 1^n and gets access to an oracle $\mathcal{O} : \{0,1\}^n \to \{0,1\}^n$.

1. Run $A(1^n)$. Whenever A queries its encryption oracle on a message $m \in \{0,1\}^n$, answer this query in the following way:
   (a) choose uniform $r \in \{0,1\}^n$
   (b) query $\mathcal{O}(r)$ and obtain response y
   (c) return the ciphertext $\langle r, y \oplus m \rangle$ to A.
2. When A outputs messages $m_0, m_1 \in \{0,1\}^n$, choose a uniform bit $b \in \{0,1\}$, then do the following:
   (a) choose uniform $r \in \{0,1\}^n$
   (b) query $\mathcal{O}(r)$ and obtain response y
   (c) return the ciphertext $\langle r, y \oplus m \rangle$ to A.
3. Continue answering encryption oracle queries of A as before until A outputs a bit b'. Output 1 if $b = b'$, and otherwise 0.

Note that D is polynomial-time since A is so. Then we have

$$\begin{aligned}\Pr_{k \leftarrow \{0,1\}^n}[D^{F_k(\cdot)}(1^n) = 1] &= \Pr[PrivKCPA_{A,\Pi}(n) = 1] \\ \Pr_{f \leftarrow Func_n}[D^{f(\cdot)}(1^n) = 1] &= \Pr[PrivKCPA_{A,\widetilde{\Pi}}(n) = 1].\end{aligned}$$

By the definition of PRFs, we get the equation $(*)$.
Step 2: we prove that

$$\Pr[PrivKCPA_{A,\widetilde{\Pi}}(n) = 1] \leq 1/2 + q(n)/2^n.$$

Let r^* denote the random string used when generating the challenging ciphertext $\langle r^*, f(r^*) \oplus m_b \rangle$. Denote by *Repeat* the event that r^* is used by the encryption oracle when answering at least one of A's queries.

1. r^* is never used when answering any of A's queries.

   $f(r^*)$ is uniformly distributed and is independent of the rest of the experiment, then we have

$$\Pr[PrivKCPA_{A,\widetilde{\Pi}}(n) = 1] = 1/2 \text{ as one-time pad.}$$

2. r^* is used when answering at least one of A's queries.

   However, since A makes at most $q(n)$ queries to its encryption oracle, and since r^* is chosen uniformly from $\{0,1\}^n$, we have

$$\Pr[\mathit{Repeat}] \leq q(n)/2^n.$$

Therefore, we have

$$\begin{aligned}
&\Pr[\mathit{PrivKCPA}_{A,\widetilde{\Pi}}(n) = 1] \\
&= \Pr[\mathit{PrivKCPA}_{A,\widetilde{\Pi}}(n) = 1 \wedge \mathit{Repeat}] + \Pr[\mathit{PrivKCPA}_{A,\widetilde{\Pi}}(n) = 1 \wedge \neg\mathit{Repeat}] \\
&\leq \Pr[\mathit{Repeat}] + \Pr[\mathit{PrivKCPA}_{A,\widetilde{\Pi}}(n) = 1 | \neg\mathit{Repeat}] \\
&\leq q(n)/2^n + 1/2.
\end{aligned}$$

By combining Steps 1 and 2, we prove that

$$\Pr[\mathit{PrivKCPA}_{A,\Pi}(n) = 1] \leq 1 + q(n)/2^n + \mathit{negl}(n) = 1/2 + \mathit{negl}'(n).$$

□

Another key assumption is that the nonce r should be uniformly chosen. Otherwise, there may exist attacks using this bias in real world. Now that we have a CPA-secure encryption scheme based on PRFs. However, there exist two drawbacks: a 1-block plaintext results in a 2-block ciphertext; this is only defined for encryption of n-bit messages.

It is noted that CPA-security is a stronger notion than security for encryption of multiple messages. Thus, to overcome the first drawback, it is straightforward to encrypt long messages block by block, and the encryption is still CPA-secure. To overcome the second drawback, we can do better by using block-cipher modes of operation: both CTR (Counter) mode and CBC (Cipher Block Chaining) mode may lead to more efficient CPA-secure encryption schemes.

CTR mode. See Figure 1. Choose $ctr \leftarrow \{0,1\}^n$, set $c_0 = ctr$; For $i = 1$ to t, $c_i = m_i \oplus F_k(ctr + i)$; Output $c_0, c_1, \ldots, c_t$. The ciphertext expansion is just 1 block.

Theorem 9 (Theorem 5.2). *If F is a PRF, then CTR mode is CPA-secure.*

One may refer to the proof in the textbook [Page 93, Katz & Lindell].

CBC mode. See Figure 2. Choose $c_0 \leftarrow \{0,1\}^n$ (also called IV); For $i = 1$ to t, $c_i = F_k(m_i \oplus c_{i-1})$; Output $c_0, c_1, \ldots, c_t$. The ciphertext expansion

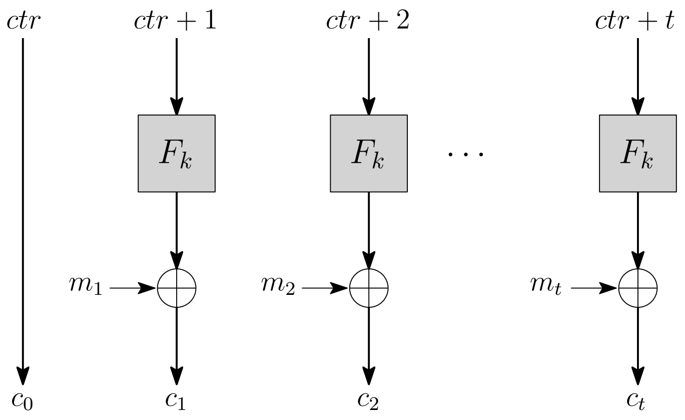

Figure 1: The CTR mode.

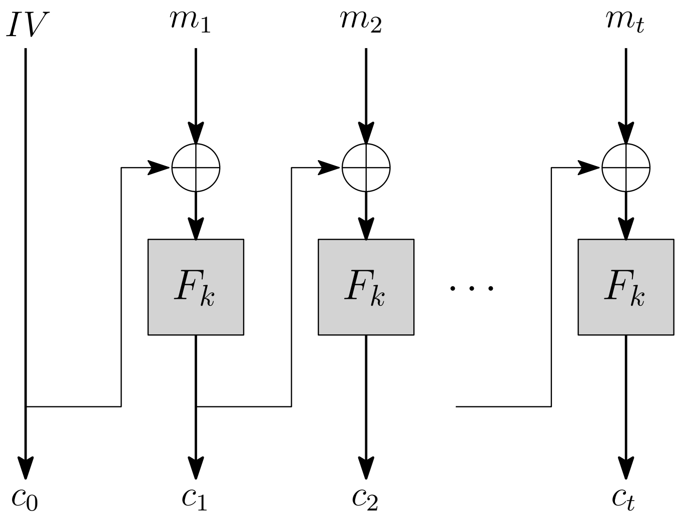

Figure 2: The CBC mode.

is also just 1 block. Decryption of a ciphertext $c_0, c_1, \ldots, c_t$ is done by computing $m_i = F_k^{-1}(c_i) \oplus c_{i-1}$ for $i = 1, \ldots, t$, and this requires F to be *invertible.*

Theorem 10 (Theorem 5.3)**.** *If F is a PRF, then CBC mode is CPA-secure.*

Note that the proof of Theorem 5.3 is much more complicated than that for the CTR mode.

A natural scheme to encrypt long messages is to use block cipher to encrypt block by block independently, that is, $Enc_k(m_1, \ldots, m_t) = F_k(m_1), \ldots, F_k(m_t)$.

This is called the *Electronic Codebook* (ECB) mode. However, note that the ECB mode of encryption is deterministic, and thus is *not* CPA-secure. Further note that the ECB mode of encryption is *not* even EAV-secure, since using ECB mode, it is possible to tell from the ciphertext whether two blocks m_i, m_j are identical.

Stream ciphers. As we defined previously, PRGs are limited in the sense that they have fixed-length output, which is produced in "one shot". In practice, PRGs are based on *stream ciphers*, which can be viewed as producing an infinite stream of pseudorandom bits on demand. Thus, stream ciphers are more flexible and more efficient. Informally, a stream cipher contains a pair of efficient and deterministic algorithms, called $Init$ and $GetBits$. Their functionalities are defined as follows:

- $Init$ takes a seed s_0 (and optional IV) as input and outputs the initial state st_0.
- $GetBits$ takes the current state st as input and outputs a bit y along with the updated state st'.

Informally, a stream cipher is *secure* if the output stream generated from a uniform seed is *pseudorandom*. There are two modes of operation for stream ciphers, i.e., synchronized mode and unsynchronized mode. In the synchronized mode, both the sender and the receiver maintain the same state as synchronized. This makes sense that in a limited-time communication session the messages are transmitted and received in order without being lost. In contrast, in the unsynchronized mode, a random IV is chosen to encrypt next message. For security, with a fixed seed s, the output of the stream cipher when using different IVs should all look uniform and independent.

So far, we have been considering security in the sense that the attacker only behaves passively. What about if the attacker can be active, that is, may interfere the communication channel, and further modify the transmitted ciphertext?

Malleability. A scheme is *malleable* if it is possible to modify a ciphertext and thereby cause a predictable change to the plaintext. All the encryption schemes we have seen so far are malleable. For example, in the one-time pad, by observing a ciphertext $c = m \oplus k$, if we modify the ciphertext by Δ, i.e., $c' = c \oplus \Delta$, it is easy to predict that this leads to the same change Δ to the plaintext after decryption, $m' = m \oplus \Delta$. To model such active attackers, we consider attackers who are able to influence what ciphertexts get decrypted,

in addition to being able to carry out a chosen-plaintext attack. This is the so-called a *chosen-ciphertext attack* (CCA).
CCA-security Let $\Pi = (Gen, Enc, Dec)$ be an encryption scheme and A an adversary. Define a randomized experiment $PrivCCA_{A,\Pi}(n)$:

1. $k \leftarrow Gen(1^n)$.
2. $A(1^n)$ interacts with an *encryption oracle* $Enc_k(\cdot)$, and a *decryption oracle* $Dec_k(\cdot)$, and then outputs m_0, m_1 of the same length.
3. $b \leftarrow \{0,1\}$, $c \leftarrow Enc_k(m_b)$, give c to A.
4. A can continue to interact with $Enc_k(\cdot), Dec_k(\cdot)$, but may not request decryption of c.
5. A outputs b'.

Adversary A succeeds if $b = b'$, and the experiment evaluates to 1 in this case.

Definition 14 (Definition 6.1)**.** Π *is* secure against chosen-ciphertext attacks *(CCA-secure) if for all PPT attackers A, there is a negligible function* ϵ *such that*
$$\Pr[PrivCCA_{A,\Pi}(n) = 1] \le 1/2 + \epsilon(n).$$

By the definition of CCA-security, it is noted that a *malleable* encryption scheme cannot be *CCA-secure*. The reason is obvious: attacker may simply modify c to c', and submit the modified ciphertext c' to the decryption oracle and thereby can determine the original message based on the result since the change to the plaintext is predictable by the definition of malleability. Therefore, *CCA-security* implies *non-malleability*. It is also worth mentioning that the CPA-secure encryption scheme in Theorem 5.1 is not CCA-secure.

In the definition of CCA-security, the attacker can obtain the decrypted message of any ciphertext of its choice except the challenge ciphertext. This indeed happens in practice. There is a realistic chosen-ciphertext attack on a natural and widely used encryption scheme. Moreover, the attack only requires the ability of an attacker to determine whether or not a modified ciphertext decrypts correctly! For details of the so-called *padding-oracle attacks*, one may refer to Chapter 3.7.2 in the textbook [Page 98, Katz & Lindell].

CCA-security are so important that modern encryption schemes are required to be CCA-secure. Note that CCA-security is defined under the threat model that attackers may actively modify the transmitted messages. Therefore, to achieve CCA-security, essentially integrity is required. *Integrity* ensures that a received message is originated from the intended party, and was not modified even if an attacker controls the channel. The tool for achieving integrity is *message authentication codes* (MAC).

Message authentication code A *message authentication code* (MAC) is composed of three PPT algorithms (Gen, Mac, $Vrfy$):

– Gen: takes as input 1^n and outputs key k (assume that $|k|gen$);

– Mac: takes as input the key k and message $m \in \{0,1\}^*$; outputs *tag* t: $t := Mac_k(m)$;

– $Vrfy$: takes the key k, message m, and tag t as input; outputs 1 ("accept") or 0 ("reject").

For all m and all k output by Gen, it holds that $Vrfy_k(m, Mac_k(m)) = 1$.

To define the security of a message authentication code, we consider the threat model of adaptive chosen-message attack, in which the attacker may induce the sender to authenticate messages of the attacker's choice, and the security goal is "existential unforgeability", i.e., attacker should be unable to forge a valid tag on any message not previously authenticated by the sender.

Formally, fix A and Π, and define a randomized experiment $Forge_{A,\Pi}(n)$:

1. $k \leftarrow Gen(1^n)$.

2. $A(1^n)$ interacts with an *oracle* $Mac_k(\cdot)$; let M be the set of messages submitted to this oracle.

3. A outputs (m, t).

4. A succeeds, and the experiment evaluates to 1, if $Vrfy_k(m,t) = 1$ and $m \notin M$.

Definition 15 (Definition 6.2)**.** Π *is secure if for all PPT attackers A, there is a negligible function ϵ such that*

$$\Pr[Forge_{A,\Pi}(n) = 1] \leq \epsilon(n).$$

An MAC satisfying this definition of security can be used where integrity is needed. *Replay attacks* are often a significant real-world concern and it is application-dependent to avoid replay attacks. We now use PRFs to construct a secure MAC for fix-length messages. Let F be a length-preserving PRF (e.g., block cipher). Construct the following MAC Π:

– *Gen*: choose a uniform key k for F;

– $Mac_k(m)$: output $F_k(m)$;

– $Vrfy_k(m, t)$: output 1 iff $F_k(m) = t$.

Theorem 11 (Theorem 6.3). *Π is a secure MAC.*

Proof. See details on Page 117 in the textbook [Katz & Lindell]. □

Note that although we have a construction of secure MAC, this only works for fixed-length, and short messages, since for example, AES has a only 128-bit block size. To authenticate arbitrary-length messages, a natural idea is to authenticate the message by block with fixed-length. More precisely, to authenticate $m = m_1||\dots||m_\ell$, we have

$$Mac'_k(m_1||\dots||m_\ell) = (t_1||\dots||t_\ell) = Mac_k(m_1)||\dots||Mac_k(m_\ell),$$

and

$$Vrfy'_k(m_1||\dots||m_\ell, t_1||\dots||t_\ell) = 1,$$

iff $Vrfy_k(m_i, t_i) = 1$ for all i. In fact, this is insecure in the sense that a few attacks have to be considered.

- *Block reordering attack.* Suppose that $t_1||t_2$ is a valid tag on the message $m_1||m_2$ with $m_1 \neq m_2$. Then $t_2||t_1$ is a valid tag that attacker may forge on the different message $m_2||m_1$.

- *Truncation attack.* For each block m_i, the corresponding tag is $t_i = Mac'_k(i||m_i)$. This is resistant to block reordering attack, since each block is assigned a counter. But attacker may simply drop blocks from the end of the message and drop the corresponding blocks of the tag as well.

- *Mix-and-match attack.* To avoid truncation attack, the length information of the whole message should be included in the tag. For example, for each block m_i, its corresponding tag is $t_i = Mac'_k(\ell||i||m_i)$. However, mix-and-match attach is possible. Suppose that $t_1||\dots||t_d$ is the tag for the message $m = m_1||\dots||m_d$, and $t'_1||\dots||t'_d$ is the tag for the message $m' = m'_1||\dots||m'_d$, then the tag $t_1||t'_2||t_3||t'_4||\dots$ is the forged tag for the new message $m'' = m_1||m'_2||m_3||m'_4||\dots$.

To construct a secure MAC for arbitrary-length messages, one solution is to use $Mac'_k(m_1||\dots||m_\ell) = r||Mac_k(r||\ell||1||m_1)||\dots||Mac_k(r||\ell||\ell||m_\ell)$, where r is a random identifier. Note that this is not very efficient, and we will introduce two methods of secure MACs for arbitrary-length messages: one is CBC-MAC, and the other is HMAC (will be discussed later).

CBC-MAC. CBC-MAC employs the idea of CBC mode, and only the final value is the output, see Figure 3.

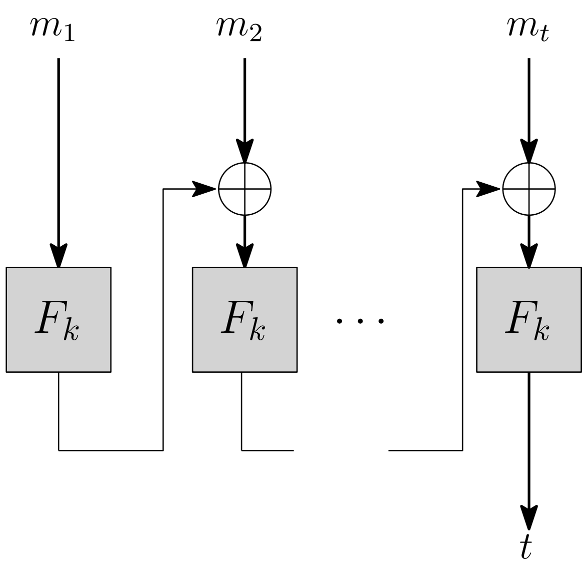

Figure 3: Basic CBC-MAC.

If F is a PRF with block length n, then for any fixed ℓ, basic CBC-MAC is a secure MAC for messages of length $\ell \cdot n$. Note that the sender and the receiver must agree on the length parameter ℓ in advance.

Question: Show that if ℓ is not agreed on in advance, the basic CBC-MAC is not secure.

Here we show a possible way to forge a tag on a new message. For the basic CBC-MAC, the corresponding tag on the message $m_1||m_2$ is $t = c_2 = F_k(c_1 \oplus m_2)$, where $c_1 = F_k(m_1)$, and the corresponding tag on the message $m'_1||m'_2$ is $t' = c'_2 = F_k(c'_1 \oplus m'_2)$, where $c'_1 = F_k(m'_1)$. Then we consider the new message $m_1||m_2||(m'_1 \oplus t)||m'_2$. Therefore, it is easy to verify that t' is the tag on this new message.

There are indeed several ways to handle variable-length messages for generating MACs. One of the simplest ways is to *prepend* the message length before applying the basic CBC-MAC as shown in Figure 4.

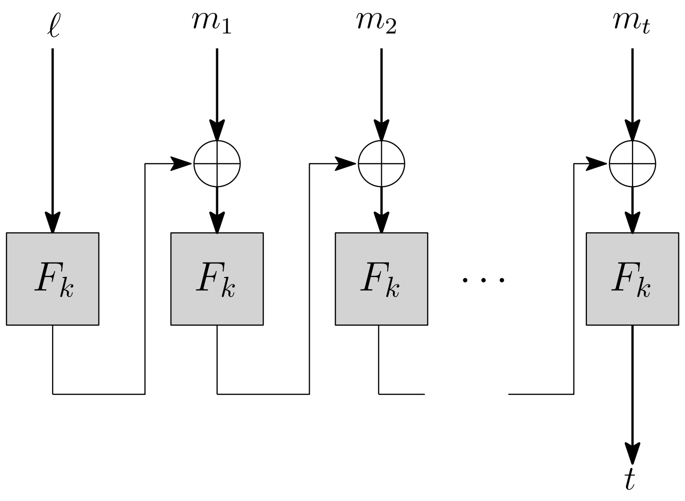

Figure 4: CBC-MAC for variable-length messages.

Question: Show that if the length ℓ is not prepended but appended in the last, the CBC-MAC for variable-length messages is not secure.

There are several ways to forge a tag, and here we give two possibilities.

(i) Obtain the tag t on the 1-block message m, the tag s on the 3-block message $m||\langle 1\rangle||t$, and the tag t' on the 1-block message m', where $m' \neq m$. Then we have

$$\begin{aligned} t &= F_k(F_k(m) \oplus \langle 1\rangle), \\ t' &= F_k(F_k(m') \oplus \langle 1\rangle), \\ s &= F_k(F_k(F_k(F_k(m) \oplus \langle 1\rangle) \oplus t) \oplus \langle 3\rangle) \\ &= F_k(F_k(F_k(F_k(m) \oplus \langle 1\rangle) \oplus F_k(F_k(m) \oplus \langle 1\rangle)) \oplus \langle 3\rangle) \\ &= F_k(F_k(0) \oplus \langle 3\rangle). \end{aligned}$$

We claim that s is a tag on the new 3-block message $m'||\langle 1\rangle||t'$, which can be verified in the following.

$$\begin{aligned} tag &= F_k(F_k(F_k(F_k(m') \oplus \langle 1\rangle) \oplus t') \oplus \langle 3\rangle) \\ &= F_k(F_k(F_k(F_k(m') \oplus \langle 1\rangle) \oplus F_k(F_k(m') \oplus \langle 1\rangle)) \oplus \langle 3\rangle) \\ &= F_k(F_k(0) \oplus \langle 3\rangle) \\ &= s. \end{aligned}$$

(ii) On the 5-block message $m_1 = A||A||A||\langle 3\rangle||B$ and its corresponding tag t_1, the 3-block message $m_2 = A||A||A||\langle 3\rangle$ and its corresponding tag t_2, the 3-block message $m_3 = C||C||C||\langle 3\rangle$ and its corresponding tag t_3, let $E = B \oplus t_2 \oplus t_3$. Then one may verify that t_1 is a valid tag on the new 5-block message $m' = C||C||C||\langle 3\rangle||E$.

Authenticated encryption. Now we have cryptographic primitives for *secrecy* and *integrity*, respectively. What if we want to achieve both? There are three natural generic constructions:

– Encrypt and authenticate (E&A): compute $c = Enc_{k_1}(m)$ and $t = Mac_{k_2}(m)$ and send (c, t) (SSH style);

– Authenticate and then encrypt (AtE): compute $t = Mac_{k_2}(m)$ and then $Enc_{k_1}(m||t)$ (SSL style);

– Encrypt and then authenticate (EtA): Compute $c = Enc_{k_1}(m)$ and $t = Mac_{k_2}(c)$ and send (c, t) (IPSec style).

Note that the two keys k_1, k_2 are independent keys for encryption and authentication, respectively. Our goal is to obtain an authenticated encryption scheme by combining a CPA-secure encryption scheme and a secure MAC. The results we discuss here mainly come from the following classical paper by Hugo Krawczyk.

[4] H. Krawczyk, The order of encryption and authentication for protecting communications (or: how secure is SSL?), *Advances in Cryptology (CRYPTO)*, pp. 310-331, 2001.

We first consider E&A: after receiving (c, t), the receiver first decrypts $m = Dec_{k_1}(c)$, and then verifies whether $Vrfy_{k_2}(m, t) = 1$. There may be two problems for this E&A construction: on the one hand, the tag t may leak information about m since nothing in the definition of a secure MAC implies that it hides information about m. As a trivial example, consider a secure MAC where the first bit of the tag is always equal to the first bit of the message. Thus, the E&A combination may not even be *EAV-secure*; on the other hand, if the MAC is deterministic (can still be secure), then the tag leaks whether the same message is encrypted twice. Hence, the E&A combination may not be *CPA-secure*.

We now consdier AtE: after receiving $c = Enc_{k_1}(m||t)$, the receiver first decrypts $m||t = Dec_{k_1}(c)$, and then verifies whether $Vrfy_{k_2}(m, t) = 1$. In this case, padding-oracle attack still works, and the AtE combination may not be *CCA-secure*. Now we give an example to explain that AtE is not secure in general. The idea is that we combine a CPA-secure encryption scheme with a secure MAC in the form of AtE, by proving it is malleable, we show that it is even not CPA-secure. This means the secure MAC just makes things worse.

We use the encryption scheme (Gen, Enc, Dec) in Theorem 5.1 that is CPA-secure to build a new encryption scheme (Gen', Enc', Dec'). For an n-bit plaintext m, first apply an encoding of m into a $2n$-bit string m' by representing each bit m_i, for $i = 1, \ldots, n$, in m with two bits in m': if the bit $m_i = 0$, then $(m'_{2i-1}, m'_{2i}) = (0, 0)$; if the bit $m_i = 1$, then $(m'_{2i-1}, m'_{2i}) = (0, 1)$ or $(1, 0)$. The encryption Enc is then applied to m'. For decrypting $c = Enc'_k(m')$, one first applies the decryption Dec to obtain m', which is then decoded into m by mapping $(0, 0) \mapsto 0$, and $(0, 1)$ or $(1, 0) \mapsto 1$. In particular, if m' contains a pair $(m'_{2i-1}, m'_{2i}) = (1, 1)$, the decoding outputs the invalidity sign $\perp$.

We consider an *active attack*. When an attacker Eve sees a transmitted ciphertext $c = Enc'_k(m)$, she can learn the first bit m_1 of m as follows: she intercepts c, flips the first two bits (c_1, c_2) of c, and sends the modified ciphertext c' to its destination. If she can obtain the information of whether the decryption outputs a valid or invalid plaintext, then Eve learns the first bit of m. This is so since the modified c' is valid if and only if $m_1 = 1$. Note that in the AtE combination, the MAC is applied to the data before encoding and encryption. If the original bit of M is 1, the change in ciphertext will result in the same decrypted plaintext and then the MAC verification will succeed.

The application of an encoding to a plaintext before encryption is commonly used for padding and other purposes. Moreover, endcoding of this type can be motivated by stronger security requirements, for example, to prevent an attacker from learning the exact length of transmitted messages or other traffic analysis information. We also emphasize that if one claims the secuity of AtE, one needs to analyze the combination as a wholeor use stronger and specific properties of the encryption function. Furthermore, this does not mean that the SSL protocol is not secure, but does mean that it is not generically secure.

The remainder case is the EtA combination: after receiving (c, t), where $t = Mac_{k_2}(c)$ and $c = Enc_{k_1}(m)$, the receiver first verifies whether $Vrfy_{k_2}(c, t) = 1$, and then decrypts $m = Dec_{k_1}(c)$. If the encryption scheme is *CPA-secure* and the MAC is *secure*, then this is an *authenticated encryption scheme*. In fact, it achieves something even stronger: given ciphertexts corresponding to chosen plaintexts $m_1, \ldots, m_k$, it is infeasible for an attacker to generate any new, valid ciphertext. Hence, encrypt-then-authenticate (EtA) with independent keys is the recommended generic approach for constructing authenticated encryption.

Authenticated encryption are useful for parties who wish to communicate securely over a period of time over which the parties are willing to maintain state. Common attacks for secure sessions include *replay attack*, *re-ordering attack*, *reflection attack* and others. These attacks can be prevented using counters/sequence numbers and identifiers.

Hash function. A *hash function* is a deterministic function that maps arbitrary-length inputs to a short, fixed-length output (sometimes called *digest*). Hash functions can be keyed or unkeyed, where the key can be part of the input. For simplicity, we restrict attention to unkeyed hash functions. Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a hash function. A *collision* is a pair of distinct inputs x, x' such that $H(x) = H(x')$.

Definition 16 (Definition 7.1)**.** *H is collision-resistant if it is infeasible for a PPT adversary to find a collision in H.*

Besides the definition of *collision resistance*, there are two related definitions.

Definition 17. *H is second-preimage resistance if given a uniform $x \in \{0,1\}^*$, it is infeasible for a PPT adversary to find an x' with $x' \neq x$ such that $H(x) = H(x')$.*

Definition 18. *H is preimage resistant if given a uniform $y \in \{0,1\}^\ell$, it is infeasible for a PPT adversary to find an x such that $H(x) = y$.*

By the definitions above, we are able to verify that

- Collision resistance implies second-preimage resistance: if given a uniform $x \in \{0,1\}^*$, an adversary can find $x' \neq x$ such that $H(x) = H(x')$, then clearly a colliding pair $x \neq x'$ can be found.
- Second-preimage resistance implies preimage resistance: if it is possible for a given y to find an x such that $H(x) = y$, then one could also take a given input x', compute $y = H(x')$, and then obtain and x with $H(x) = y$. With high probability it holds that $x' \neq x$, i.e., a second preimage has been found.

What is the best generic *collision attack* on a hash function $H : \{0,1\}^* \to \{0,1\}^\ell$? Note that collisions are guaranteed to exist: simply compute $H(x_1)$, $\ldots, H(x_{2^\ell+1})$, for $2^\ell + 1$ distinct values, we are guaranteed to find a collision. A better way to find collisions is the so-called *birthday attack*.

We consider a generic question: By computing $H(x_1), \ldots, H(x_k)$, what is the probability that a collision occurs? Or how many people are needed to have a 50% chance that there are two people who share the same birthday?

Let A denote the event that at least two people in the room have the same birthday, and B denote the event that no two people in the room have the same birthday. Then A and B are complement of each other, i.e., $\Pr[A] = 1 - \Pr[B]$. We have

$$\begin{aligned}\Pr[B] &= \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdot \cdots \cdot \left(1 - \frac{n-1}{365}\right) \\ &= \prod_{i=1}^{n-1} \left(1 - \frac{i}{365}\right),\end{aligned}$$

and

$$\Pr[A] = 1 - \prod_{i=1}^{n-1} \left(1 - \frac{i}{365}\right).$$

We use a generic notation of this probability $p(n; H) := 1 - \prod_{i=1}^{n-1} \left(1 - \frac{i}{H}\right)$, where H denotes the size of the image set of the hash function, and n denotes the number of values computed on the hash function. By the Taylor expansion $e^x = 1 + x + \frac{x^2}{2!} + \cdots$, and for $|x| \ll 1$, $e^x \approx 1 + x$, we have $e^{-i/H} \approx 1 - \frac{i}{H}$. Then the probability can be approximated as

$$p(n; H) \approx 1 - e^{-n(n-1)/2H} \approx 1 - e^{-n^2/2H}.$$

Let $n(p; H)$ denote the smallest number of values we have to choose, such that the probability for finding a collision is at least p. By inverting the expression above, we have

$$n(p; H) \approx \sqrt{2H \ln \frac{1}{1-p}}.$$

This equation well explains the reason why 23 people suffice to have the probability of a collision $\geq 50\%$, and $O(2^{\ell/2})$ hash-function evaluations are needed to have the probability of a collision $\geq 50\%$. In other words, $\ell = 2n$-bit output length is needed to get security against attackers running in time 2^n.

Merkle-Damgård construction. Practical construction of cryptographic hash functions are based on a basic compression function $h : \{0,1\}^{2n} \rightarrow \{0,1\}^n$. The function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ is defined as

$$H(m_1, \ldots, m_t) = h(h(h(m_1, IV), m_2), \ldots, m_t),$$

when the message $m = m_1||\dots||m_t$ is composed of t blocks. This is known as the Merkle-Damgård construction.

Theorem 12 (Theorem 8.1)**.** *(Merkle-Damgård construction preserves collision resistance) Let H be constructed from h above. Then given two messages $m \neq m' \in \{0,1\}^{tn}$ such that $H(m) = H(m')$, we can efficiently find two messages $x \neq x' \in \{0,1\}^{2n}$ such that $h(x) = h(x')$.*

Proof. We show that a collision for H yields a collision in h. Suppose that

$$\begin{aligned} H(m_1, m_2, \dots, m_t) &= h(h(h(m_1, IV), m_2), \dots, m_t) \\ H(m'_1, m'_2, \dots, m'_t) &= h(h(h(m'_1, IV), m'_2), \dots, m'_t). \end{aligned}$$

If $H(m_1, m_2, \dots, m_t) = H(m'_1, m'_2, \dots, m'_t)$, then there are two cases:

- Case 1: $h(\dots), m_t = h(\dots), m'_t$, then $m_t = m'_t$ and $h(h(\dots), \dots, m_{t-1}) = h(h(\dots), \dots, m'_{t-1})$. We continue to go backwards on h until we find a collision for h.
- Case 2: $h(\dots), m_t = h(\dots), m'_t$. THen we have dound a collision for h, i.e., $x = h(\dots)||m_t$ and $x' = h(\dots)||m'_t$.

□

Hash-and-MAC. Recall that we may use CBC-MAC to have a secure MAC for arbitrary-length messages. Now we introduce another secure MAC for arbitrary-legnth messages using hash functions, i.e., hash-and-MAC. For an arbitrary-lengt message m, the sender uses a hash function H to compute $h = H(m)$, and then use a secure MAC to generate a tag on h, $t = Mac_k(h)$. At the recevier end, the recevier first computes $h = H(m)$, and then verifies whether $Vrfy_k(h, t) = 1$.

Theorem 13 (Theorem 8.2)**.** *If the MAC is secure for fixed-length messages and H is collision-resistant, then the construction is a secure MAC for arbitrary-length messages.*

Proof. We give the sketch of the proof here, and one may check more details on Pages 159-161 in the textbook [Katz & Lindell].

Say the sender authenticates $m_1, m_2, \dots$ and let $h_i = H(m_i)$. An attacker outputs a forgery (m, t) where $m \neq m_i$ for all i. Then there are two cases that may happen:

– $H(m) = H(m_i)$ for some i, then this leads to a collision for the hash funciton H;

- $H(m) \neq H(m_i)$ for all i, then this leads to a forgery in the underlying MAC for fixed-length MAC.

$\square$

In practice, popular hash funcitons include SHA-2, and SHA-3, where "SHA" stands for Secure Hash Algorithms. Attacks on hash functions and design of new hash algorithms for specific scenarios are active research directions in symmetric cryptogrpahy. Hash function are ubiquitous in practical applications. For example, they can be used as one-way functions, and can be modeled as a "random oracle", and can also be used as "fingerprinting" for file integrity. We consider the problem of outsourcing files to an untrusted server. The task is to outsource n files $x_1, \ldots, x_n$ to a server, and ensure the integrity of all these files. There are several ways to complete this task:

- Client computes $h_i = H(x_i)$ for all i, and stores $x_1, \ldots, x_n$ on the server. On request i, server sends the file x_i to client, and client verifies whether $H(x_i) = h_i$. In this case, client storage is $O(n)$ for the n hash values and the communication is $O(|x|)$.
- Client computes $h = H(x_1, \ldots, x_n)$, and stores $x_1, \ldots, x_n$ on the server. On request i, server sends all the n files to client, and client verifies whether $H(x_1, \ldots, x_n) = h$ by the n files from the server. In this case, client storage is $O(1)$ and the communication is $O(n \cdot |x|)$.
- Client computes $h = H(H(x_1), \ldots, H(x_n))$, and stores $x_1, \ldots, x_n$ on the server. On request i, server sends the file x_i together with $h_1, \ldots, h_n$ to client, and client verifies whether $H(h_1, H(x_i), \ldots, h_n) = h$. In this case, client storage is $O(1)$ and the communication is $O(n + |x|)$.
- Using a *Merkle tree* as shown in Figure 5, client only stores the hash value of the root of the Merkel tree. For example, on request 3, server sends x_3, along with x_4, $h_{1,2} = H(x_1, x_2)$, and $h_{5,6,7,8} = H(H(x_5, x_6), H(x_7, x_8))$ to client. Client computes $h'_{1,2,3,4} = H(h_{1,2}, H(x_3, x_4))$ and $h'_{1\ldots4,5\ldots8} = H(h'_{1\ldots4}, h'_{5\ldots8})$, and then verifies whether $h_{1\ldots4,5\ldots8} = h'_{1\ldots4,5\ldots8}$. In this case, client storage is $O(1)$ and the communication is $O(\log n + |x|)$.

Hash functions are widely used in *key derivation.* As we recall, in encryption schemes, the key generation algorithm *Gen* generates keys uniformly. How to derive a key from shared high-entropy information, e.g., biometric data? Let X be a distribution. Given shared information x sample from distribution X, if we use a collision resistant hash function H and derive shared

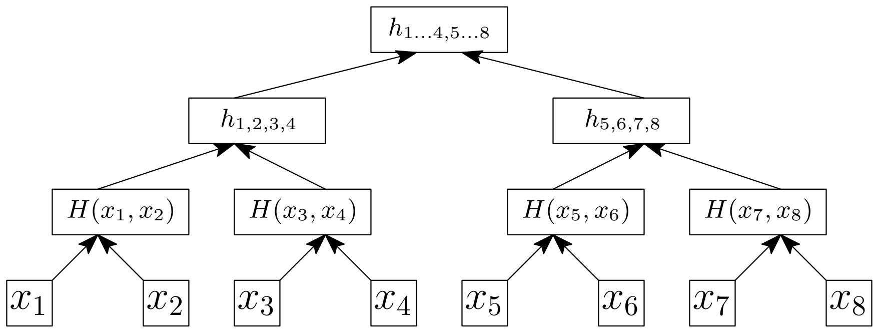

Figure 5: Merkle tree.

key $k = H(x)$, then we are able to claim that k is a "good" cryptographic key.

Practical constructions of symmetric-key primitives. Read Chapter 6 in the textbook [Katz & Lindell] for details about the practical constructions and standards of symmetric-key primitives, including stream ciphers, block ciphers, and hash functions.

Number theory and cryptographic hardness assumptions. A *group* is a set G and a binary operation $\circ$ defined on G such that:

- (*Closure*) For all $g, h \in G$, $g \circ h$ is in G.
- (*Identity*) There is a unique element $e \in G$ such that $g \circ e = g$ for all $g \in G$.
- (*Inverse*) Every element $g \in G$ has an inverse $h \in G$ such that $g \circ h = e$.
- (*Associativity*) For all $f, g, h \in G$, the associativity law holds: $f \circ (g \circ h) = (f \circ g) \circ h$.

A group is called *abelian* if the following property further holds.

- (*Commutativity*) For all $g, h \in G$, $g \circ h = h \circ g$.

The *order* of a finite group G is the number of elements in G.
Example Determine whether the following are groups or not.
$\mathbb{Z}$ under addition
$\mathbb{Z}\backslash\{0\}$ under multiplication
$\mathbb{Q}$ under addition
$\mathbb{Q}\backslash\{0\}$ under multiplication

$\mathbb{R}$ under addition
$\mathbb{R}\backslash\{0\}$ under multiplication
$\{0,1\}^*$ under concantenation
$\{0,1\}^n$ under bitwise XOR
2×2 real matrices under addition
2×2 invertible, real matrices under multiplication

The group operation can be written additively or multiplicatively, i.e, instead of $g \circ h$, we write $g + h$ or $g \cdot h$. Note that this does not mean that the group operation is the (integer) addition or multiplication. The identity element is usually denoted by 0 or 1, respectively. The inverse of an element g is denoted by $-g$ or g^{-1}, respectively. The group exponentiation is denoted by $m \cdot a$ or a^m, respectively.

Consider $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$ under addition modulo N. It is easily checked that $\mathbb{Z}_N$ is indeed a group. What if we consider multiplication modulo N? In general, $\mathbb{Z}_N$ is not a group under multiplication modulo N, even if we exclude 0 element. Instead we consider the set $\mathbb{Z}_N^* = \{x : 0 < x < N \text{ and } \gcd(x, N) = 1\}$, and one may check that $\mathbb{Z}_N^*$ is also a group with the order $\phi(N)$. If $N = p$ is a prime, then $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ is actually a prime *field*.

Let $s_n = \langle 1, 2, \ldots, n\rangle$ denote a sequence of integers 1 through n. Denote by P_n the set of all permutations of the sequence s_n. For example, $s_3 = \langle 1, 2, 3\rangle$, then

$$P_3 = \{\langle 1,2,3\rangle, \langle 1,3,2\rangle, \langle 2,1,3\rangle, \langle 2,3,1\rangle, \langle 3,1,2\rangle, \langle 3,2,1\rangle\}.$$

Define a binary operation $\circ$ on the elements in P_n: for $\rho, \pi \in P_n$, $\pi \circ \rho$ denotes a *re-permutation* of the elements of ρ according to the elements of π. For $\pi = \langle 3,2,1\rangle$ and $\rho = \langle 1,3,2\rangle$, we can calculate that $\pi \circ \rho = \langle 2,3,1\rangle$ and $\rho \circ \pi = \langle 3,1,2\rangle$. One may check that all the properties of a group are satisfied and therefore $(P_n, \circ)$ is a permutation group. Note that $(P_n, \circ)$ is not abelian.