# Lecture 2

## 1 One Time Pad

The One Time Pad sounds like a great solution, but it has its share of problems:

- There is no integrity protection. Suppose that Alice wants to send a message $M$ to Bob using key $K$. Let $C = M \oplus K$, and let $C' = C$ with one bit flipped. If Bob gets $C'$, he can decrypt it (with one bit flipped), but may not be able to tell that the message was modified by a third party. (Later in the class, we will see how this can be fixed using message authentication).
- The one time pad can only send fixed-length messages. Note that sending a smaller message by truncating the key to fit a smaller message leaks information about the size of the message, so instead short messages should be padded to the key length.
- Each key can only be used once.
- The key length has to be just as long as the message; this leads to many problems. You must have all the keys in advance, for all the messages you will send in the future; storing all these keys requires a lot of space. You must also generate them securely, transfer them securely, and store them securely.

Can Shannon secrecy be achieved with a shorter key? As it turns out, it cannot.

Theorem 1
*Let $E$ be an encryption algorithm, $E : Keys \times M \rightarrow C$. If $E$ satisfies perfect secrecy, then $|Keys| \geq |M|$ That is, the key space must be at least as large as the message space.*

Proof: Fix a cipher text $c \in C$. Construct $D = \{D_k(c) : k \in \text{Keys}\}$. Note that $|D| \leq |\text{Keys}|$. Furthermore $\forall m \in M, \exists k \in \text{Keys}$ s.t. $D_k(c) = m$. So $\forall m \in M, Pr[E_k(m) = c] > 0$. $|\{D_k(c) : k \in \text{Keys}\}| = |M|$. □

This applies to all encryption algorithms, even if they are randomized. Shannon showed something even stronger that this. He showed that the entropy in the key needs to be at least as much as the entropy in the message.

## 2 Other Definitions of Security

Since the information theoretic definition has not worked well for us, lets try an algorithmic definition. The following are a few possible definitions; an encryption algorithm is secure if:

1. $\Pr[E_k(m_0) = c] = \Pr[E_k(m_1) = c]$ for all $m_0, m_1 \in M, c \in C$

2. $\Pr[E_k(m_0) \in S] = \Pr[E_k(m_1) \in S]$ for all $m_0, m_1 \in M, S \subseteq C$. Though this seems like a stronger and stricter definition than the previous one, they are in fact equivalent.

3. $|\Pr[E_k(m_0) \in S] - \Pr[E_k(m_1) \in S]| \le \epsilon$ for all $m_1, m_0, S$. Informally, this says that an encryption algorithm is secure if the probabilities of sending two messages to $S$ are very close to each other, though not necessarily equal. This is called *statistically secure*, or *$\epsilon$-statistically secure*. Note that def. 2 implies def. 3 (setting $\epsilon = 0$), but the reverse is not true.

4. $|\Pr[A(E_k(m_0)) = 1] - \Pr[A(E_k(m_1)) = 1]| \le \epsilon$ for all algorithms $A$, for all $m_0, m_1 \in M$. This definition is equivalent to def. 3, because there exists a bijection between $S$ and $A$: given $S$, we map it to the algorithm that sends members of $S$ to 1, and sends everything else to 0; given an algorithm $A$, we define the corresponding set $S$ to be the inputs for which $A$ outputs 1, and 0 for the rest of the inputs.

5. $\Pr[A(E_k(m_0)) = 1] - \Pr[A(E_k(m_1)) = 1] \le \epsilon$ for all algorithms $A$, for all $m_0, m_1 \in M$, with the restriction that the worst cast running time is T, and the algorithm will always terminate and output a bit. This definition is the same as 4, but requires not that it holds for all algorithms, just that it holds for all efficient algorithms. Definitions $1 - 4$ required long keys, this is the first definition in the list that escapes that constraint. This is known as *computational security.*

# 3 Distinguish-ability

1. $\Pr[X \xleftarrow{\$} D; X = x] = \Pr[X' \xleftarrow{\$} D'; X' = x]$ for all $x$. This is known as *perfect-indistinguishability.*

2. $|\Pr[X \xleftarrow{\$} D; A(X) = 1] - \Pr[X' \xleftarrow{\$} D'; A(X') = 1]| \le \epsilon$ for all algorithms $A$. This is known as *statistically $\epsilon$-indistinguishable*, and the left hand side is the *advantage* of $A$, Adv$A$. You can think of this as a game: you are given at random, one of two black boxes. Each black box takes no inputs, and has only one output. One blackbox is programmed to output $x \xleftarrow{\$} D$, the other is programmed to output $x \xleftarrow{\$} D'$. The point of the game is to figure out which box you were given. If you guess blindly, you will guess each box 50% of the time, giving you an advantage of 0. The advantage is a measure of how much better you can do that that. Note that there is no bound on the running time in this formulation.

3. Distributions $D$ and $D'$ are *computationally indistinguishable*, specifically, $(t, \epsilon) -$ indistinguishable if no matter what strategy you use, if you take no longer than time $t$, then you have advantage at most $\epsilon$. That is Adv$A \le \epsilon$ for all algorithms $A$ with running time $\le t$.

# 4 Examples

## 4.1

Define the following subroutine:

```
unsigned int state; // initialized with a random 32 bit number

rand() {
    state = 1103515245 * state + 12345;
    return state;
}
```

To produce a random bit, output `rand()` (mod 2). More generally, let $x_0$ be chosen uniformly at random from $\{0,1\}^{32}$. Let $\alpha, \beta$ be two numbers, then define

$$\begin{aligned} x_{i+1} &= \alpha x_i + \beta \pmod{2^{32}} \\ y_i &= x_i \pmod{2} \end{aligned}$$

where $y_i$ is the $i$th output of our pseudo random number generator.

Define $Y = (\mathrm{lsb}(y_1), \mathrm{lsb}(y_2), \ldots, \mathrm{lsb}(y_{128}))$, where $\mathrm{lsb}(y)$ denotes the least significant bit of $y$. Is $Y$ indistinguishable from $U \xleftarrow{\$} \{0,1\}^{128}$? Because the seed $x_0$ can only take on $2^{32}$ values, and $Y$ is a deterministic function of that seed, $Y$ can only have $2^{32}$ unique values, which is much less that $2^{128}$, so there is a very large advantage here. In fact, the specific case above with $\alpha = 1103515245$, $\beta = 12345$, $Y$ has only two possible values, $010101\ldots$ or $101010\ldots$. This is because $x_i$ is either even or odd. If it is even, then $\alpha x_i$ is even and $\alpha x_i + \beta$ is odd. If $x_i$ is odd, then $\alpha x_i$ is odd, and $\alpha x_i + \beta$ is even.

## 4.2

Let $n = pq$, for large prime numbers $p, q$. Construct

$$\begin{aligned} X, Y &\xleftarrow{\$} \{0, 1, \ldots, n-1\}. \\ Z &= Y^3 \bmod n \end{aligned}$$

Then $X$ is indistinguishable from $Z$.

## 4.3

If $X$ is indistinguishable from $Z$, then $\mathrm{lsb}(X)$ is indistinguishable from $\mathrm{lsb}(Z)$, where lsb() is the least significant bit.

Theorem 2
*Suppose $X, Y$ are $(t, \epsilon)$-indistinguishable, then lsb$(X)$, lsb$(Y)$ are $(t-1, \epsilon)$-indistinguishable.*

PROOF: Proof by reduction through the contra-positive. Suppose an algorithm $A$ (with run-time $t$) distinguishes $\mathrm{lsb}(X)$ from $\mathrm{lsb}(Y)$ and Adv $A > \epsilon$. Define a strategy $B$ by $B(w) = A(\mathrm{lsb}(w))$. Then

$$\begin{aligned} \mathrm{Adv}B &= |\Pr[B(X) = 1] - \Pr[B(Y) = 1]| \\ &= |\Pr[A(\mathrm{lsb}(X)) = 1] - \Pr[A(\mathrm{lsb}(Y)) = 1]| \\ &= \mathrm{Adv}A \\ &> \epsilon \end{aligned}$$

□

The running time of $B$ is just the running time of $A$ plus the run time required to calculate lsb(). More generally, you can replace the function lsb() in the above proof by any deterministic function, but then you have to make sure to account for the change in run-time in stating how indistinguishable the new distributions are.

THEOREM 3
*Suppose $X, Y$ are $(t, \epsilon')$-indistinguishable, and $Y, Z$ are $(t, \epsilon')$-indistinguishable. Then $X, Z$ are $(t, \epsilon + \epsilon')$-indistinguishable.*
PROOF:

$$\begin{aligned} \mathrm{Adv}^{\{X,Z\}}A &= |\Pr[A(X) = 1] - \Pr[A(Z) = 1]| \\ &= |\Pr[A(X) = 1] - \Pr[A(Y) = 1] + \Pr[A(Y) = 1] - \Pr[A(Z) = 1]| \\ &\leq |\Pr[A(X) = 1] - \Pr[A(Y) = 1]| + |\Pr[A(Y) = 1] - \Pr[A(Z) = 1]| \\ &= \mathrm{Adv}^{\{X,Z\}}A + \mathrm{Adv}^{\{Y,Z\}}A \end{aligned}$$

□