# CS276 Scribe Notes

Emil Ong <emilong@cs.berkeley.edu>

January 26, 2004

## 1 Recap from last class

**Definition.** $E$ satisifies Shannon secrecy with respect to the random variable $M$ if

$$\Pr_{M,K}[M = m | E_K(M) = c] = \Pr_M[M = m] \qquad \forall m \in \mathcal{M}, c \in \mathcal{C}$$

**Definition.** $E$ satisifies perfect secrecy with respect to the random variable $M$ if

$$\Pr_K[E_K(m_0) = c] = \Pr_K[E_K(m_1) = c] \qquad \forall m_0, m_1 \in \mathcal{M}, c \in \mathcal{C}$$

**Example.** The one-time-pad encryption scheme (OTP) is $E_k(x) = k \oplus x$, $E : \{0,1\}^l \times \{0,1\}^l \to \{0,1\}^l$.

**Theorem.** OTP has perfect secrecy.

## 2 This class

**Theorem.** $E$ has perfect secrecy $\iff$ ($E$ has Shannon secrecy with respect to random variable $M$ for all random variables $M$).

*Proof.* $\Rightarrow$ Fix any $m \in \mathcal{M}, c \in \mathcal{C}$.

**Lemma.** If $E$ has perfect secrecy, then

$$\Pr_{M,K}[E_K(M) = c] = \Pr_K[E_K(m) = c] \qquad \forall m \in \mathcal{M}, c \in \mathcal{C}, \text{random variables } M$$

*Proof.*

$$\begin{aligned}
\Pr_{M,K}[E_K(M) = c] &= \sum_{m' \in \mathcal{M}} \Pr_K[E_K(m') = c] \cdot \Pr_M[M = m'] && (1)\\
&= \Pr_K[E_K(m) = c] \cdot \sum_{m' \in \mathcal{M}} \Pr_M[M = m'] && \text{by perfect secrecy} \quad (2)\\
&= \Pr_K[E_K(m) = c] && (3)
\end{aligned}$$

□

By Bayes' rule,

$$\begin{aligned}
\Pr[M = m | E_K(M) = c] &= \frac{\Pr[E_K(M) = c | M = m] \cdot \Pr[M = m]}{\Pr[E_K(M) = c]} && (4)\\
&= \frac{\Pr[E_K(m) = c] \cdot \Pr[M = m]}{\Pr[E_K(M) = c]} && (5)\\
&= \Pr[M = m] && \text{by the lemma} \quad (6)
\end{aligned}$$

$\Leftarrow$ According to 5 and 6,

$$\frac{\Pr[E_K(m) = c] \cdot \Pr[M = m]}{\Pr[E_K(M) = c]} = \Pr[M = m] \tag{7}$$

$$\Pr[E_K(m) = c] = \Pr[E_K(M) = c] \tag{8}$$

□

## 2.1 The trouble with perfect secrecy

We can conclude that OTP thus has Shannon and perfect secrecy, so it's wonderful. There are however some drawbacks:

1. You must exchange keys before sending the message.
2. The key must be as long as the message.
3. You can only use the key once.

**Theorem.** If $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ has perfect secrecy, then $|\mathcal{K}| \geq |\mathcal{M}|$.

*Proof.* Fix $c \in \mathcal{C}$. Then $|\{D_k(c) : k \in \mathcal{K}\}| \leq |\mathcal{K}|$. In other words, the number of messages to which we can decrypt $c$ with any key $k$ must be less than or equal to the total number of keys in our key space. Perfect secrecy implies that $|\{D_k(c) : k \in \mathcal{K}\}| = |\mathcal{M}|$. Thus, $|\mathcal{K}| \geq |\mathcal{M}|$. □

We would like to see if there are some relaxed forms of perfect secrecy that would preserve privacy, but also be practical. We can restate perfect secrecy as

$$\begin{aligned} \text{Perfect secrecy} &\iff E_K(m_0) \text{ has the exact same distribution as } E_K(m_1) \\ &\iff \forall S \subseteq \mathcal{C}, \Pr[E_K(m_0) \in S] = \Pr[E_K(m_1) \in S] \end{aligned}$$

Using this final form, we can construct *statistical secrecy*:

$$\forall S \subseteq \mathcal{C}, |\Pr[E_K(m_0) \in S] - \Pr[E_K(m_1) \in S]| \leq \epsilon$$

This formulation is looser than perfect privacy, but is still hard to use. Statistical secrecy is fortunately equivalent to saying that for all algorithms $A$ of any running time,

We can thus describe *computation secrecy* by saying that for all *efficient* algorithms $A$,

$$|\Pr[A(E_K(m_0)) = 1] - \Pr[A(E_K(m_1)) = 1]| \leq \epsilon$$

With this definition, we can discuss the criteria for creating a "good" pseudo-random number generator.

## 2.2 Pseudo-Random number Generators (PRG's/PRNG's)

**Definition.** $G : \{0,1\}^k \to \{0,1\}^n$ is a pseudo-random number generator if

1. $n > k$
2. $G$ is efficiently computable

The reason that we might want a PRG is to try to emulate a OTP. Specifically, we would like to use a small input of size $k$ to create a larger output of size $n$. The output of this PRG would serve as our one-time-pad, but we would have described it in only $k$ bits. Ideally, an output string from out OTP should be as difficult to guess as an $n$-bit string chosen uniformly at random.

**Example.** A bad PRG: Linear congruential. Start with the seeds $a, b$. The $i^{\text{th}}$ output of the generator is given by $x_i = (a \cdot x_{i-1} + b) \bmod 2^{32}$. We only need 3 $x_i$'s to get a system of two linear equations with two unknowns (the unknowns being $a$ and $b$). Thus, given a small portion of the output of this PRG, we are able to determine the complete output.

Back to the ideal case... suppose $u_m$ is the uniform distribution on $m$-bits. Then we would like the distributions $G(u_k)$ and $u_n$ to be computationally indistinguishable (i.e. difficult to distinguish given a reasonable amount of computational time).

**Definition.** Let $D, D'$ be two distributions on the same set. Then the *advantage* of algorithm $A$ is

$$\mathrm{Adv}(A) := |\Pr_{x \leftarrow D}[A(x) = 1] - \Pr_{x' \leftarrow D'}[A(x') = 1]|$$

**Definition.** (Concrete security) $D$ and $D'$ are $(t, \epsilon)$-*indistinguishable* if $\mathrm{Adv}(A) \leq \epsilon$ for all algorithms $A$ with running time less than or equal to $t$.[1]

**Definition.** (Asymptotic version of concrete security) Two families of distributions $\{D_n\}, \{D'_n\}$ are *(asymptotically) computationally indistinguishable* if for all polynomials $p$, $\exists \epsilon : \mathbf{N} \to \mathbf{R}$, for all polynomials $q$ such that:

1. $D_n, D'_n$ are $(p(n), \epsilon(n))$-indistinguishable $\forall n$
2. $\epsilon(n) = \mathrm{O}(\frac{1}{q(n)})$

Theorists tend to like the asymptotic version better than first version.

**Definition.** $G : X \to Y$ is a $(t, \epsilon)$-PRG if $G(u_x)$ and $u_y$ are $(t, \epsilon)$-indistinguishable.

No proven $(t, \epsilon)$-PRG's exist, but experience has shown that some PRG's are at least pretty hard to crack.

### 2.2.1 Some quick PRG theorems

**Theorem.** Let $G : \{0,1\}^k \to \{0,1\}^n$ be a $(t, \epsilon)$-PRG. Define $\mathrm{first}_m : \{0,1\}^n \to \{0,1\}^m$ as $\mathrm{first}_m(x_{1..n}) = x_{1..m}$. Let $H : \{0,1\}^k \to \{0,1\}^m$ be defined as $H(x) = \mathrm{first}_m(G(x))$. Then $H$ is a $(t - \mathrm{O}(m), \epsilon)$-PRG.

*Proof.* Suppose $B$ is an adversary that $(t - \mathrm{O}(m), \epsilon)$-breaks $H$ (i.e. runs in $t - \mathrm{O}(m)$ and has $\mathrm{Adv}(B) > \epsilon$). Define $A$ by $A(y) = B(\mathrm{first}_m(y))$. Then $A$ $(t, \epsilon)$-breaks $G$:

1. $A$ runs in time $\leq t$ (because $\mathrm{first}_m(y)$ takes $\mathrm{O}(m)$ time to run).
2. $$\begin{aligned}\mathrm{Adv}(A) &= |\Pr_{y \leftarrow G(u_k)}[A(y) = 1] - \Pr_{y' \leftarrow u_n}[A(y') = 1]| \\ &= |\Pr_{x \leftarrow u_k}[A(G(x)) = 1] - \Pr_{y' \leftarrow u_n}[A(y') = 1]| \\ &= |\Pr_{x \leftarrow u_k}[B(\mathrm{first}_m(G(x))) = 1] - \Pr_{y' \leftarrow u_n}[B(\mathrm{first}_m(y')) = 1]| \\ &= |\Pr_{x \leftarrow u_k}[B(H(x)) = 1] - \Pr_{z' \leftarrow u_m}[B(z') = 1]| \\ &= |\Pr_{z \leftarrow H(u_k)}[B(z) = 1] - \Pr_{z' \leftarrow u_m}[B(z') = 1]| \\ &= \mathrm{Adv}(B) \\ &> \epsilon\end{aligned}$$

$A$'s existence contradicts the assumption that $G$ is a $(t, \epsilon)$-PRG. □

**Theorem.** Define $H : X \times X \to Y \times Y$ by $H(x, x') = (G(x), G(x'))$. Let $G : X \to Y$ be a $(t, \epsilon)$-PRG that runs in $t_G$. Then $H$ is a $(t - t_G, 2\epsilon)$-PRG.

*Proof.* Will be completed in the next class by using the triangle inequality to show that since $(G(u_x), G(u'_x))$ and $(G(u_x), u'_y))$ are close and $(G(u_x), u'_y))$ and $(u_y, u'_y)$ are close, then $(G(u_x), G(u'_x))$ and $(u_y, u'_y)$ are close. □

[1] Note that we are not specifying a model of computation. Thus, we'll use asymptotic running times.