CS 282A/MATH 209A: Foundations of Cryptography	Prof. Rafail Ostrosky

Lecture 4

Lecture date: January 26, 2005 *Scribe: Paul Ray, Mike Welch, Fernando Pereira*

1 Private Key Encryption

Consider a game between three players: Alice, Bob and Eve. In this game, Alice and Bob are communicating over an insecure channel, where Eve can listen to messages, although she cannot modify them. Informally, Bob and Alice win if they are able to communicate without Eve being able to decipher any partial information about the messages. On the other hand, Eve wins if she can discover any information about the messages sent. Notice that Eve does not have to find the whole content of the messages in order to win; for her, it suffices to discover any information hidden in the messages that she did not know when the game started.

In this game, a *plaintext message* m is encrypted by means of a *cipher key*, to produce a *ciphertext* c. Because Eve does not have access to the key used to encrypt the messages, it is said that Bob and Alice communicate by means of a *private key encryption system*. In a private key encryption system, it is necessary that Bob and Alice meet before starting to communicate, so they can agree on a particular key, which should be keep secret, that is, only they must know it. This is demonstrated in Figure 1.

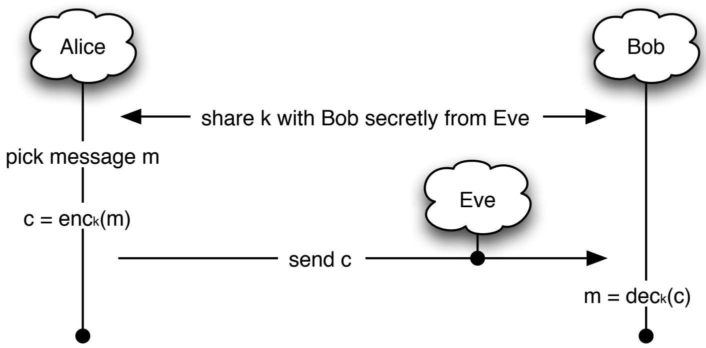

Figure 1: Alice and Bob use a private-key encryption scheme.

Sometimes, in a private key system, an encryption scheme is associated with a message space M and some distribution on M. For example, M may be the set of all strings for a given length. The ciphertext space is the set of all possible plaintext messages $m \in M$

encrypted with each possible key. Note that the ciphertext space does not necessary equal M, that is, for some combinations of m and key k the same ciphertext can be produced. Some ciphertext strings may not correspond to any plaintext. A formal definition of a Private Key cryptosystem follows below:

Definition 1 *We define an encryption scheme as a triple of algorithms (GEN, ENC, DEC):*

- *GEN: Key Generation: a randomized polynomial time algorithm that outputs a key K.*
  $K \leftarrow GEN(1^n, R)$
  where 1^n is a security parameter and R is a polynomial number of coin-flips.
- *ENC: Encryption: a randomized polynomial time algorithm that takes a plaintext message m and key K and generates a ciphertext C.*
  $C \leftarrow ENC_K(m, R)$
- *DEC: Decryption: a polynomial time algorithm that takes a cipertext C and key K and outputs a plaintext message m'.*
  $m' \leftarrow DEC_K(C)$
- *Correctness: a cryptographic system is called correct if $DEC_K(ENC_K(m, R)) = m$, provided that m has been generated by the GEN algorithm. Notice that we can allow negligible error probability: $Pr[DEC_K(ENC_K(m, R)) \neq m] < \epsilon$.*

1.1 Examples of Private Key Systems

Cryptosystems that we used in kindergarden are easy to break. Examples include:

Shift cipher (insecure!) For a shift cipher with $m \in \{A...Z\}^n$. K ← GEN() assigns K a random number in the range {0...25}. C ← $ENC_K(m_1m_2...m_l) = c_1c_2...c_l$ such that $c_i = (m_i + K)$ modulus 26. m' ← $DEC_K(c_1c_2...c_l) = m'_1m'_2...m'_l$ where $m'_i = (c_i - K)$ modulus 26.
Example: If K = 4, m = "HELLO" then C = "LIPPT".
Because there are only 26 different keys, this system is trivial to break.

Caesar cipher (insecure!) For a substitution cipher, we let K be a random permutation of the numbers {0...25}. Then, C ← $ENC_K(m_1m_2...m_l) = c_1c_2...c_l$ where $c_i = K(m_i)$.
Example: For a limited alphabet $\{A, B, C, D\}$, let K be the mapping $\{A, B, C, D\} \rightarrow \{B, D, A, C\}$. Then for m = "DAD", C = "CBC".
Again, this type of cipher is weak, with a very limited key space. For example, if the only allowed symbols are the lower case characters of the English alphabet, then

there are 26 possible substitutions! However, in order to break this system, it is not necessary to try every possible combination. Attacks based on the frequency of each character are very effective against the Caesar cipher.

On the other hand, we will see that the following cryptosystem is secure:

One-time pad For a one-time pad, we let K $\leftarrow \{0,1\}^n$ with M the set of all binary strings of length l. We let $ENC_K(m)$ = m $\oplus$ K (bitwise). The decryption function $DEC_K(c)$ = c $\oplus$ K (bitwise).
Example: If m = 11001 and K = 01101, C = 11001 $\oplus$ 01101 = 10100.

1.2 Security Criteria

There are several different formal definitions of an information-theoretically *secure cryptographic* system. One is the *Shannon Security Criterion* (Definition 2). According to the Shannon criterion, the probability that a certain message was generated, given the knowledge of a ciphertext, is equal to the *a-priori* probability that any message was produced. That is, the knowledge of the ciphertext does not provides any *a-posteriori* information about the message *m*, except what was already know from *a-priori* probabilities.

Definition 2 *An encryption scheme over message space M is Shannon secure if, for all distributions D over M, for all $m \in M$, and for all ciphers c, the following equation holds:*

$$Pr_D[m|c] = Pr_D[m]$$

The equation above means that the a posteriori *probability that a message m has been sent, given that the cipher c has been sent, is equal to the* a priori *probability that message m has been sent.*

Claim 3 *One-time pad is Shannon secure*

Proof For all the possible distributions D of messages over the message space $M = \{0,1\}^n$, for all keys in the key space $K = \{0,1\}^n$, for all fixed $m \in M$, and for a fixed $c \in \{0,1\}^n$:

$$Pr_{D,k}[m|c] = \frac{Pr_{D,k}[m \wedge c]}{Pr_{D,k}[c]} = \frac{Pr_{D,k}[c|m] \cdot Pr_{D,k}[m]}{Pr_{D,k}[c]}$$

$Pr_{D,k}[c|m] = Pr_{D,k}[m \oplus k = c] = \frac{1}{2^n}$

Notice that, for a given (m, c) pair, there is only one possible value for k such that $k = m \oplus c$. Also, the probability of a given cipher c been produced is given by:

$$Pr_{D,k}[c] = \Sigma_{m \in M} Pr_{D,k}[c|m] \cdot Pr_{D,k}[m] = 2^{-n} \cdot \Sigma_{m \in M} Pr_{D,k}[m] = 2^{-n}$$

Using the latter result in the first equation gives:

$Pr_{D,k}[m|c] = \frac{2^{-n} \cdot Pr_{D,k}[m]}{2^{-n}}$

$Pr_{D,k}[m|c] = Pr_{D,k}[m]$

$Pr_D[m|c] = Pr_D[m]$

■

Unbreakable cryptographic systems can also be characterized by the concept of *Perfect Security* (Definition 4). This definition states that an encryption scheme is perfectly secure if a cyphertext c is an equally likely output for any two messages in the message space.

Definition 4 *An encryption scheme S over message space M is perfectly secure if, for any distribution D over any set $\{m_1, m_2\} \subset M$ of two messages of same length, and for all cipher texts c we have:*

$$Pr_D[m_1|c] = Pr_D[m_2|c] \tag{1}$$

Given Definitions 2 and 4, which is stronger? That is, which gives the user of a cryptosystem the highest level of security? Although they look different, it is possible to show that they are essentially equivalent definitions. Here we prove that Shannon security is a consequence of perfect security.

Claim 5 *If perfect security holds then Shannon security holds.*

Proof

Given arbitrary messages m_1 and m_2 from the message space M, according to perfect security, we have $Pr_D[c|m_1] = Pr_D[c|m_2]$, where D is any distribution over some arbitrary subset of M with just two elements, m_1 and m_2. Since this equation is true for arbitrary messages m_1 and m_2, it must hold for any pair of messages in M. Assume that the following probabilities range over any distribution of messages from M:

$$Pr[m|c] = \frac{Pr[c|m] \cdot Pr[m]}{Pr[c]}$$

$$= \frac{Pr[c|m] \cdot Pr[m]}{\Sigma_{m' \in M} Pr[c|m'] \cdot Pr[m']}$$

$$= \frac{Pr[c|m] \cdot Pr[m]}{\Sigma_{m' \in M} Pr[c|m] \cdot Pr[m']}$$

$$= \frac{Pr[c|m] \cdot Pr[m]}{Pr[c|m] \cdot \Sigma_{m' \in M} Pr[m']} = Pr[m]$$

Because $Pr[c|m] = Pr[c|m']$ for any message m', it is possible to remove the probability $Pr[c|m']$ from the summation. Also, $\Sigma_{m' \in M} Pr[m'] = 1$, given that a message has been produced. This two remarks justify the last steps of the derivation above.

■

Claim 6 *For any perfectly secure encryption, the number of keys $|K|$ is at least as big as the size of the message space $|M|$.*

Proof Consider an encryption scheme S in which the size of the key space is less than the size of the message space, that is, $|K| < |M|$. Bob and Alice will try to use S to defeat Eve. Given a message $m \in M$, and a key $k \in K$, Bob generates a cipher text $c = ENC_k(m)$, and sends it to Alice. Eve intercepts c and now applies every possible key k' in order to decrypt c. If S is a perfectly secure system, for any message m, and any key k, $Pr_k[m|c] = Pr_k[m]$. However, because $|K| < |M|$, there must be at least one message m such that $Pr_k[m|c] = 0$. Therefore, after knowing c, one of the following situations must hold: $Pr_k[m|c] > Pr_k[m]$, or $Pr_k[m|c] = 0$. In this case, the cryptographic system does not meet the Shannon security criterion, and, consequently, perfect security does not hold.

■

Fact 7 *Shift and Substitution Ciphers do* **not** *satisfy perfect security.*

Proof This is easy to see using Claim 6. ■

Fact 8 *1-Time Pad is perfectly secure.*

2 Indistinguishability

The problem with 1-Time Pad is that the key must be as long as the message, and we cannot seem to do better if we need information-theoretic security. We will see that using so-called pseudo-random generators and weaker "computational security", we can make the key of the private-key encryption scheme much shorter. In the context of this discussion, a pseudo-random generator will be defined as a deterministic polynomial-time algorithm that takes as input an initial sequence of truly random bits, called a *seed*, plus a polynomial Q, and outputs a new, apparently "random looking" sequence of size equal to at least $Q(|seed|)$. A simplified scheme is shown in Figure 2.

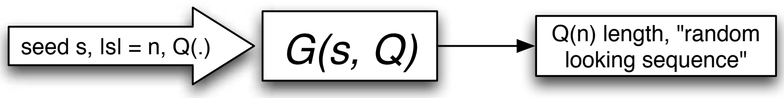

Figure 2: A pseudo-random generator.

Before moving to the formal definition, it is important to give an intuitive idea about what does it mean for a sequence of symbols to be "apparently random". A distribution is called pseudo-random if it is not possible to build an algorithm of polynomial complexity that can distinguish that distribution from a truly random distribution of strings of the same length. It is possible to establish an analogy between this concept and the idea underlying the *Turing Test*. This test, proposed by Alan Turing, aims to determine if a computer could be considered intelligent. According to the test, a computer is considered intelligent if it is impossible to distinguish it from a true human being in an interview conducted on-line. In our case, the test aims to distinguish between a truly random and a pseudo-random sequence of bits. Such sequences, according to Definition 9, are said to be from a *Sampleable Distribution*.

Definition 9 *A Sampleable Distribution – A sampleable distribution is a probabilistic polynomial time algorithm* $S(1^n, r)$ *that takes a finite sequence of random bits* r *and* 1^n*, and produces strings of length* n*, such that, for some distribution* X_n*:*

$$Pr_r[S(1^n, r) = \alpha] = Pr_{X_n}[X_n = \alpha] \tag{2}$$

The test T, used to determine if two distributions are the same or not, is called a *statistical test*. According to definition 10, given two sampleable distributions X_n and Y_n, it is said that they are indistinguishable with respect to the probabilistic polynomial test T if T cannot determine whether a sample comes from X_n or Y_n. Figure 4 represents a statistical test.

Definition 10 *The random distributions X_n and Y_n are said to pass the probabilistic polynomial test T if $\forall c, \exists N$ such that $\forall n > N$:*

$$|Pr_{w,X_n}[T_w(X_n) = 1] - Pr_{w,Y_n}[T_w(Y_n) = 1]| < \frac{1}{n^c} \tag{3}$$

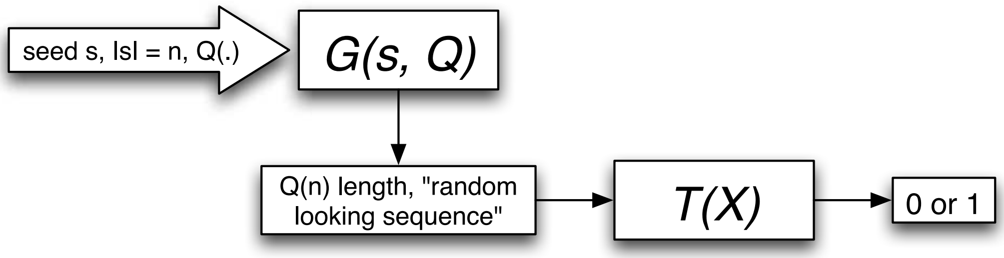

Figure 3: Experiment 1: statistical test receiving pseudo-random string.

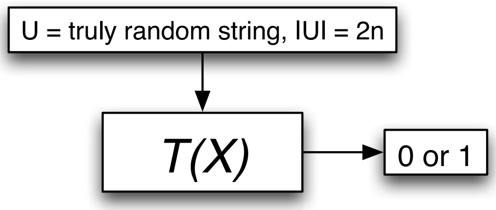

Figure 4: Experiment 2: statistical test receiving truly random sequence.

According to Definition 11, two distributions are statistically close if the difference in probability that a sample string α comes from one distribution versus the other is negligible.

Definition 11 *We say that two distributions X_n and Y_n are statistically close if $\forall c, \exists N$, such that $\forall n > N$:*

$$\sum_{\alpha \in \{0,1\}^n} |Pr[X_n = \alpha] - Pr[Y_n = \alpha]| < \frac{1}{n^c} \tag{4}$$

If it is not possible for any polynomial time algorithm to distinguish two distributions, it is said that these distributions are computationally close. Definition 12 formally states this fact.

Definition 12 *[Yao] X_n and Y_n are polynomial-time indistinguishable if, and only if, $\forall c$, and $\forall A \in PPT$, $\exists n$ such that $\forall n > N$:*

$$|Pr[A(X_n) = 1] - Pr[A(Y_n) = 1]| < \frac{1}{n^c} \tag{5}$$

2.1 Extended Statistical Test

One could argue that, if it was possible to feed the testing algorithm T with several, instead of just one, samples from the random distributions, perhaps it would be easier to distinguish a pseudo-random sequence from a truly random distribution. According to Definition 13, a test that receives more than one sample is called an *extended statistical test.* An extended statistical test T' is a machine that takes a polynomial number of inputs and ouputs a single bit in $\{0, 1\}$. Given two distributions X_n and Y_n, T' is given a polynomial number of samples from either X_n or Y_n (not both). Then, if T' determines that the samples come from X_n it outputs bit b, otherwise it outputs bit $\overline{b}$

Definition 13 *Extended Statistical Test – The sampliable distributions X_n, and Y_n pass extend statistical test T' if $\forall c_1, c_2, \exists N$ such that $\forall n > N$:*

$$[|Pr(T'(X_n^1, \ldots X_n^{c_2}) = 1) - Pr(T'(Y_n^1, \ldots Y_n^{c_2}) = 1)|] < \frac{1}{n^{c_1}} \tag{6}$$

Claim 14 *If X_n and Y_n are sampleable distributions which can be distinguished by a (uniform) extended statistical test T', then there exist a single sample (uniform) statistical test T which distinguishes X_n and Y_n.*

Proof Let k =poly(n) and $\epsilon(n) = 1/k$. We assume that there exists T' and show how to construct T. Assuming that there exists T' means, w.l.o.g. that

$$Pr_{X_n}(T'(X_1, X_2, X_3, ..., X_{poly}) = 1) - Pr_{Y_n}(T'(Y_1, Y_2, Y_3, ..., Y_{poly}) = 1) > \epsilon(n)$$

Consider "hybrids" P_j, for $0 \le j \le k$, where in P_j the first j samples come from Y_n and the remaining samples come from X_n:

$$\begin{aligned}
P_0 &= x_1\, x_2\, x_3\, x_4 \ldots x_k \\
P_1 &= y_1\, x_2\, x_3\, x_4 \ldots x_k \\
P_2 &= y_1\, y_2\, x_3\, x_4 \ldots x_k \\
P_3 &= y_1\, y_2\, y_3\, x_4 \ldots x_k
\end{aligned}$$

$$\ldots$$

$$P_k \ = \ y_1 \, y_2 \, y_3 \, y_4 \ldots y_k$$

We know that $P_0 - P_k > \epsilon(n)$, and therefore, $\exists j$ such that $P_j - P_{j+1} > \epsilon(n)/k$ (which is another 1/poly fraction!) Consider a distribution:

$$P(\boxed{\text{z}}) \ = \ y_1 \, y_2 \, y_3 \ldots y_j \, \boxed{\text{z}} \, x_{j+2} \ldots x_k$$

Notice that if z is a sample from Y_n then $P(z) = P_j$ and if z is a sample from X_n then $P(z) = P_{j+1}$. Hence, if we are given z on which we have to guess which distribution it came from, if we put z in the box above, and somehow fix other locations we could distinguish on a single sample z. Two questions remain: (1) how do we find the correct $j+1$ position, and (2), how do we fix other values. The answers differ in uniform and non-uniform case:

<u>non-uniform case</u> (i.e. both T' and T are circuits): Since T is a circuit, we can non-uniformly find the correct $j+1$ value and find values to other variables which maximizes distinguishing probability.

<u>uniform case</u> : Since X_n and Y_n are sampleable, we can fix values different from j to be samples from X_n and Y_n and by guessing correctly j (we guess the position of j correctly with probability 1/poly). The distinguishing probability could be further improved. By experimenting with the distinguisher, we get (again using sampleability of X_n and Y_n!) to check if our choice of j and samples of other positions are good. ■

3 Pseudo-Random Generators

A Pseudo-Random Generator must be able to generate a sequence of symbols which cannot be distinguished, in polynomial time, from a truly random distribution, even though such a sequence is deterministically generated, given a short truly random seed. Definition 15 formally defines the properties of a pseudo-random generator.

Definition 15 A deterministic algorithm $G(\cdot, \cdot)$ is pseudo-random generator if:

- 1. $G(x, Q)$ runs in time polynomial in $|x|, Q(|x|)$ where Q is a polynomial.
- 2. $G(x, Q)$ outputs strings of length $Q(|x|)$ for all x.
- 3. For every polynomial $Q()$, the induced distribution $\{G(x, Q)\}$ is indistinguishable from $\{U_{Q(|x|)}\}$, where $U_{Q(|x|)}$ is a uniform distribution on strings of length $Q(|x|)$.

3.1 Construction

We will now show how we can construct a pseudo-random generator from a one-way permutation. Remember from the previous lecture that for a one-way permutation $f(x)$, it is possible to obtain a hard-core bit b, which is as difficult to predict as inverting f.

Consider the following steps:

1. Pick a one-way permutation function f, random seed X_0, and a random bit string P, $|X_0| = |P| = n$
2. For i from 1 to $Q(|x|)$ do:
   Calculate $X_i = f(X_{i-1})$ and output the hardcore bit $< P * X_{i-1} >$
3. Output $X_{Q(|x|)}$ and P
4. Return the sequence of random bits in reverse order.

Remark In the construction above, the bits are returned in "reversed" order. However, since we will show that this is pseudo-random, the order (left–to-right or right-to-left) is unimportant: if it is pseudo-random one way, it must be pseudo-random the other way. We have chosen this particular order of the outputs to make the proof easier.

A graphical representation of the algorithm is shown in Figure 5.

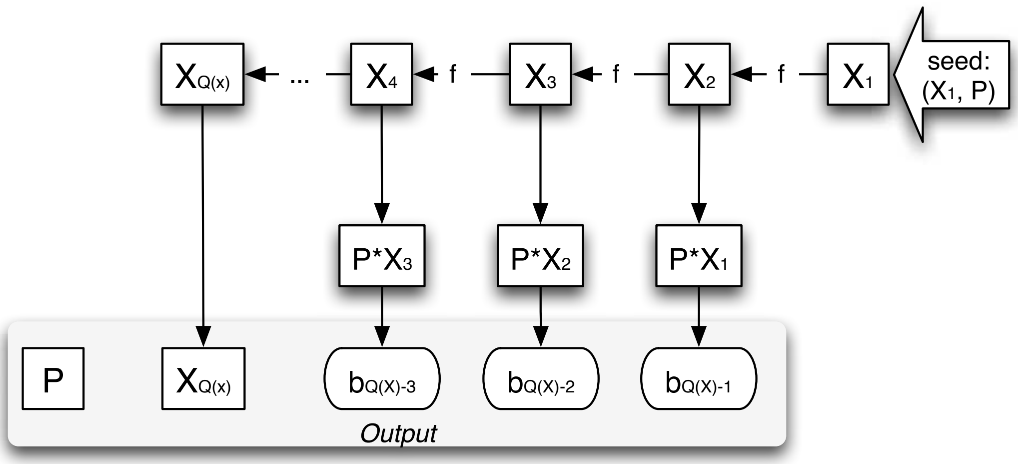

Figure 5: A pseudo-random generator.

The proof that the output of the above algorithm is pseudo-random is shown in two steps: First, we will show that the pseudo-random sequence is unpredictable as defined by Blum and Micali. Then we show that every unpredictable sequence is pseudo-random (which was shown by Yao).

Informally, a pseudo-random generator G is *unpredictable* if it is passes the next-bit-test, defined as follows: given $G_1, ..., G_k$, it is hard to predict G_{k+1}, for any k. That is, given the first k bits of the output, it is hard to predict the next bit with probability better than $\frac{1}{2}$

Definition 16 *X_n passes* the next bit test *(is unpredictable) if $\forall c$ and $\forall A \in PPT$, there $\exists N$ such that $\forall n > N$ and $\forall i, (0 \leq i \leq n)$:*

$$Pr_{X_n,\ coins\ of\ A}\ [A(\ \textit{first } i \textit{ bits } b_1, b_2, \ldots, b_i \textit{ of } x \in X_n\ ) = b_{i+1}] < \frac{1}{2} + \frac{1}{n^c} \tag{7}$$

Claim 17 If f is a strong one-way permutation, then G as defined in the previous section is unpredictable.

Proof The proof is by contradiction. We will show that if G does not pass the next bit test, then we can invert a one-way permutation on a random input. The construction is as follows: We are given an adversary $A \in PPT$ which for some prefix $b_1 ... b_i$ of output from a pseudo-random generator, can compute b_{i+1} with probability $> 1/2 + \epsilon(n) = 1/2 + 1/\text{poly}$. We wish to invert f, that is, find $f^{-1}(y)$. We know that if we can predict a hard-core bit of $f^{-1}(y)$ and p (for a random p), then we can find the inverse of y (with 1/poly probability). We make y the $(n-i)^{th}$ X_i value of the generator. We can then compute (by applying f to y i times, and computing hard-core bits) the first i bits of the generator in the straightforward fashion. Finally, we feed it to our "next-bit" predictor A. Notice that the next bit is exactly the hard-core bit of y. ■

Remark Notice that we are dealing with f, which is a one-way permutation. Hence, a uniform distribution of the inputs implies a uniform distribution of the outputs, and in the above experiment, the fact that we start with y and compute the first i bits of the output, has the same (uniform) distribution over the seeds.

Claim 18 Ensemble X_n is pseudo-random if and only if X_n is unpredictable.

Proof Proof in one direction is trivial: a next bit test is a type of statistical test. Hence if it passes all polynomial-time statistical tests it passes the next-bit test as well.

In the opposite direction, we must show that passing the next bit test implies passing all polynomial-time statistical tests. We will show a proof by contradiction. That is, if $\exists$ distinguisher D, then $\exists$ a next bit predictor. A distinguisher means that for distributions

p = X_n and q = Y_n, $|p - q| > \frac{1}{n^c}$.
By the hybrid argument, there $\exists$i such that $|P_i - P_{i+1}| > \frac{\epsilon}{l}$. Then, we can construct an $(i+1)^{st}$ bit predictor as follows:
Give an adversary A an input string $B_1...B_i\hat{b}Y$, where $\hat{b}$ is a random bit. If the adversary outputs 1 on the input, we output $\hat{b}$, otherwise we output $\bar{\hat{b}}$.

$$\begin{aligned}\Pr\left[A(B_1...B_i) = b_{i+1}\right] &= \Pr\left[\hat{b} = b_{i+1}\right] * \Pr\left[A(B_1...B_i b_{i+1} Y) = 1\right] \\ &+ \Pr\left[b_{i+1} = \bar{\hat{b}}\right] * \Pr\left[A(B_1...B_i \overline{b_{i+1}} Y) = 0\right] \\ &= \tfrac{1}{2} * P_{i+1} + \tfrac{1}{2} * q\end{aligned}$$

So, what is q? To figure it out, let's expand P_i

$$\begin{aligned}P_i = \Pr\left[A(B_1...B_i Y)\right] &= \Pr\left[A(B_1...B_i b_{i+1} Y) = 1\right] * \tfrac{1}{2} \\ &+ \Pr\left[A(B_1...B_i \overline{b_{i+1}} Y) = 1\right] * \tfrac{1}{2} \\ &= P_{i+1} * \tfrac{1}{2} + (1 - q) * \tfrac{1}{2}\end{aligned}$$

or, $q = 1 + P_{i+1} - 2 * P_i$

Substituting for q above, we get:
$= \frac{1}{2} * P_{i+1} + \frac{1}{2} * q = \frac{1}{2} * P_{i+1} + \frac{1}{2}[1 + P_{i+1} - 2P_i] = \frac{1}{2}[P_{i+1} - P_i] > \frac{1}{2} + \frac{\epsilon}{l}$ ■