6.857 Homework	**Problem Set 1**	# 1-3. Detecting Pad Reuse
Qian Long		
Joseph W Presbrey		
Mikhail Rudoy		February 22, 2014

1-3a)

Define $R(a, b)$ to be the length of the longest bitstring that is a substring of both the bittexts a and b. Define $S(a, b)$ to be the length of the longest bitstring that is a substring of both the bitstrings a and b in the same positions.

First we will prove two lemmas:

Lemma 1. *Suppose we are given two bitstrings a and b of length $l \leq k$ that satisfy $\mathbb{P}(a = s) = \frac{1}{2^l}$ and $\mathbb{P}(b = s) = \frac{1}{2^l}$ for every $s \in \{0,1\}^l$. Then for any asymptotically positive polynomial $q(k)$, we have $\mathbb{P}\Bigg(S(a, b) > \log_2(k) + \log_2 \ln(k)\Bigg) < \frac{1}{q(k)}$ provided k is sufficiently large.*

Proof. The condition on a and b implies that the xor of the two ciphertexts, $a \oplus b$, can be any bitstring of length l with uniform probability. A bit of $a \oplus b$ is 0 if and only if the corresponding bits of a and b are the same. The value $S(a, b)$ is the length of the longest sequence of bits in a such that the corresponding sequence of bits in b is the same. In other words, $S(a, b)$ is the length of the longest run of 0s in $a \oplus b$.

Since $a \oplus b$ is chosen uniformly at random, we can pretend that it was generated by a sequence of l coin flips. We identify 0s in $a \oplus b$ with coins landing heads.

Consider first the case that $l = k$. In this case, the useful fact given in the problem applies. The given fact asserts (in this case) that for any asymptotically positive polynomial $q(k)$,

$$\mathbb{P}\Bigg(\log_2(k) - \log_2 \ln\ln(k) \leq S(a, b) \leq log_2(k) + \log_2 \ln(k)\Bigg) \geq 1 - \frac{1}{q(k)}$$

provided $k \geq K$ (for some K dependent only on q). Then

$$\mathbb{P}\Bigg(\log_2(k) - \log_2 \ln\ln(k) > S(a, b) \text{ or } S(a, b) > log_2(k) + \log_2 \ln(k)\Bigg) < \frac{1}{q(k)}$$

and finally

$$\mathbb{P}\Bigg(S(a, b) > log_2(k) + \log_2 \ln(k)\Bigg) < \frac{1}{q(k)}$$

Thus in the case $l = k$, our desired result is proved.

For every case where $l \neq k$, the number of coin flips is smaller, and so the probability of a long run of heads can only decrease from the $l = k$ case. Thus we have shown that

$$\mathbb{P}\Bigg(S(a, b) > log_2(k) + \log_2 \ln(k)\Bigg) < \frac{1}{q(k)}$$

for all cases, provided k is sufficiently large. □

Lemma 2. *Given that two ciphertexts u and v of length n are properly encrypted, for any asymptotically positive polynomial $q(n)$, we have $\mathbb{P}\Bigg(R(u,v) > \log_2(n) + \log_2 \ln(n)\Bigg) < \frac{1}{q(n)}$ provided n is sufficiently large.*

Proof. Let $u_{[-i]}$ and $v_{[-i]}$ denote, for non-negative i, the substring of u or v which excluding the first i bits. Similarly, define $u_{[i]}$ and $v_{[i]}$ to be, for non-negative i, the substring which excludes the last i bits.

Let s be the longest substring of both u and v. $R(u,v)$ is then the length of s. Suppose s begins at index j_u in u and at index j_v in v, let $i = j_v - j_u$. Consider $u_{[i]}$ and $v_{[-i]}$. In all cases, the first $|i|$ bits are removed from the string in which the index where s begins is larger. The last $|i|$ bits are removed from the other string. The result is that both new strings contain s at the same range of indices, and both strings have the same length. Thus $S(u_{[i]}, v_{[-i]}) \geq R(u,v)$ for some choice of i where $-n \leq i \leq n$. We can conclude that $R(u,v) \leq \max\limits_{-n\leq i\leq n}\Bigg(S(u_{[i]}, v_{[-i]})\Bigg)$.

Since u and v are properly encrypted, we know that $\mathbb{P}(u_{[i]} = s) = \mathbb{P}(v_{[-i]} = s) = \frac{1}{2^{n-|i|}}$ for every $s \in \{0,1\}^{n-|i|}$ and for every i with $-n \leq i \leq n$. In addition, $u_{[i]}$ and $v_{[-i]}$ always have length at most n, so we can apply Lemma 1. The lemma tells us that for any asymptotically positive polynomial p, provided n is sufficiently large, $\mathbb{P}\Bigg(S(u_{[i]}, v_{[-i]}) > \log_2(n) + \log_2 \ln(n)\Bigg) < \frac{1}{p(n)}$. Taking $p(n) = (2n+1)q(n)$, we see that provided n is sufficiently large, $\mathbb{P}\Bigg(S(u_{[i]}, v_{[-i]}) > \log_2(n) + \log_2 \ln(n)\Bigg) < \frac{1}{(2n+1)q(n)}$.

Since $R(u,v) \leq \max\limits_{-n\leq i\leq n}\Bigg(S(u_{[i]}, v_{[-i]})\Bigg)$, we know that

$$\mathbb{P}\Bigg(R(u,v) > \log_2(n) + \log_2 \ln(n)\Bigg) \leq \mathbb{P}\Bigg(\max_{-n\leq i\leq n}\Bigg(S(u_{[i]}, v_{[-i]})\Bigg) > \log_2(n) + \log_2 \ln(n)\Bigg)$$

$$= \mathbb{P}\left(\begin{matrix} S(u_{[-n]}, v_{[n]}) > \log_2(n) + \log_2 \ln(n) & \text{or} \\ S(u_{[-(n-1)]}, v_{[(n-1)]}) > \log_2(n) + \log_2 \ln(n) & \text{or} \\ \ldots & \text{or} \\ S(u_{[(n-1)]}, v_{[-(n-1)]}) > \log_2(n) + \log_2 \ln(n) & \text{or} \\ S(u_{[n]}, v_{[-n]}) > \log_2(n) + \log_2 \ln(n) & \end{matrix}\right)$$

$$\leq \mathbb{P}\Bigg(S(u_{[n]}, v_{[-n]}) > \log_2(n) + \log_2 \ln(n)\Bigg) + \ldots + \mathbb{P}\Bigg(S(u_{[-n]}, v_{[n]}) > \log_2(n) + \log_2 \ln(n)\Bigg)$$

$$\leq \frac{1}{(2n+1)q(n)} + \frac{1}{(2n+1)q(n)} + \ldots + \frac{1}{(2n+1)q(n)} = \frac{(2n+1)}{(2n+1)q(n)} = \frac{1}{q(n)}$$

We have shown that $\mathbb{P}\Bigg(R(u,v) > \log_2(n) + \log_2 \ln(n)\Bigg) < \frac{1}{q(n)}$ provided n is sufficiently large, exactly as desired. $\square$

Let $l(n)$ be the number of ciphertexts we are given. Let $r(n)$ be any asymptotically positive polynomial.

What we are interested is the value

$$\mathbb{P}\left(\max_{1\le i<j\le l(n)} R(c_i,c_j) \le \log_2(n)+\log_2\ln(n)\right)$$

and in particular, we wish to show that it is greater than $1-\frac{1}{r(n)}$ for $n \ge n_0$ for some n_0.

Alternatively, since

$$\mathbb{P}\left(\max_{1\le i<j\le l(n)} R(c_i,c_j) > \log_2(n)+\log_2\ln(n)\right) = 1-\mathbb{P}\left(\max_{1\le i<j\le l(n)} R(c_i,c_j) \le \log_2(n)+\log_2\ln(n)\right)$$

we must simply show that $\mathbb{P}\left(\max_{1\le i<j\le l(n)} R(c_i,c_j) > \log_2(n)+\log_2\ln(n)\right) < \frac{1}{r(n)}$ holds for $n \ge n_0$ for some n_0.

In addition, we know that

$$\mathbb{P}\left(\max_{1\le i<j\le l(n)} R(c_i,c_j) > \log_2(n)+\log_2\ln(n)\right) = \mathbb{P}\begin{pmatrix} R(c_1,c_2) > \log_2(n)+\log_2\ln(n) & \text{or}\\ R(c_1,c_3) > \log_2(n)+\log_2\ln(n) & \text{or}\\ R(c_2,c_3) > \log_2(n)+\log_2\ln(n) & \text{or}\\ \dots & \text{or}\\ R(c_{l(n)-1},c_{l(n)}) > \log_2(n)+\log_2\ln(n) & \end{pmatrix}$$

$$\le \sum_{i=1}^{l(n)-1}\sum_{j=i+1}^{l(n)} \mathbb{P}\Big(R(c_i,c_j) > \log_2(n)+\log_2\ln(n)\Big)$$

By Lemma 2, $\mathbb{P}\Big(R(c_i,c_j) > \log_2(n)+\log_2\ln(n)\Big) < \frac{1}{q(n)}$ for every i, j, and asymptotically positive polynomial $q(n)$, provided $n \ge n_0$ for some n_0 dependent only on q. Let $q(n) = \frac{1}{2}r(n)l(n)(l(n)-1)$, and let N be the associated value of n_0. Then for $n \ge N$ we see that

$$\sum_{i=1}^{l(n)-1}\sum_{j=i+1}^{l(n)} \mathbb{P}\Big(R(c_i,c_j) > \log_2(n)+\log_2\ln(n)\Big) < \sum_{i=1}^{l(n)-1}\sum_{j=i+1}^{l(n)} \frac{1}{q(n)}$$

$$= \frac{l(n)(l(n)-1)}{2} \times \frac{1}{q(n)} = \frac{l(n)(l(n)-1)}{2q(n)} = \frac{l(n)(l(n)-1)}{2\frac{1}{2}r(n)l(n)(l(n)-1)} = \frac{1}{r(n)}$$

We can conclude that $\mathbb{P}\left(\max_{1\le i<j\le l(n)} R(c_i,c_j) > \log_2(n)+\log_2\ln(n)\right) < \frac{1}{r(n)}$ for $n \ge N$, so

$$\mathbb{P}\left(\max_{1\le i<j\le l(n)} R(c_i,c_j) \le \log_2(n)+\log_2\ln(n)\right) > 1-\frac{1}{r(n)}$$

for $n \ge N$. Since $r(n)$ can be any asymptotically positive polynomial, we see that the length of the longest repeated bitstring $\left(\max_{1\le i<j\le l(n)} R(c_i,c_j)\right)$ is, with high probability, at most $\log_2(n) + \log_2\ln(n)$, as desired.

1-3b)

Since the bound from part a only applies in the asymptotic limit, it is not significant to evaluate it at small inputs and compare it to some other function. We wish instead to compare its asymptotic behavior to that of the average length of the longest run of identical bits in pairs of English passages.

Consider the given graph. Let the function being represented be expressed as $y = f(x)$. Here $x = \ln n$, and y is the average length of the longest run of characters in the same positions of two English passages. We wish to discuss $\hat{y} = \hat{f}(x)$, the average length of the longest run of identical bits. Let l be the length of the longest run of identical characters in some particular two passages, and let $\hat{l}$ be the length of the longest run of identical bits in the same two passages. We note that each character is composed of 8 bits, so, as a result, it is always true that $8l \leq \hat{l}$. Since the two bitstrings can also match in at most the 7 bits preceding and the 7 bits following the matching characters, $\hat{l} \leq 8l + 14$. Since y is the average value of l over some sample and $\hat{y}$ is the average value of $\hat{l}$ over the same sample, we see that $8y \leq \hat{y} \leq 8y + 14$. We can simply say that $\hat{f}(x) = \hat{y} \approx 8y = 8f(x)$, and that $\Theta(\hat{f}(x)) = \Theta(f(x))$. From the graph we see that f is super-linear, so $f(x) = \omega(x)$, and as a result, so is $\hat{f(x)}$.

Let $g(x)$ be the bound found in part a, expressed in terms of x. $g(x) = \log_2(n) + \log_2 \ln(n) = \log_2(e) \times \ln(n) + \log_2 \ln(n) = x \times \log_2(e) + \log_2(x)$. We know that $x \times \log_2(e) = \Theta(x)$ and $\log_2(x) = o(x)$, so $g(x) = \Theta(x) + o(x) = \Theta(x)$.

We have seen that $g(x) = \Theta(x)$, and that $\hat{f}(x) = \omega(x)$, so $\hat{f}(x) = \omega(g(x))$. In other words, for sufficiently large x, the value of $\hat{f}(x)$ will exceed the value of $g(x)$, and pairs of English texts will, on average, have longer common sub-bitstrings in corresponding positions than the longest common substrings (in any position) of pairs of correctly encrypted cryptotexts.

1-3c)

We will describe (and prove the correctness of) an algorithm to find an instance of pad reuse:

We are given $l(n) = poly(n)$ n-bit ciphertexts (c_0 through $c_{l(n)-1}$) with one instance of pad reuse. Let $\Sigma = \{0, 1, 2, ..., l(n) - 1\} \cup \{(a, b) : a \in \{0, 1\}$ and $b \in \{0, 1, 2, ..., n - 1\}\}$. We will construct a string s of length $l(n) \times (n + 1)$ in Σ^* using the following rule: $s = s_0 \| s_1 \| ... \| s_{l(n)-1}$, where each s_i is a string in Σ^{n+1} determined by c_i.

Let $(c_i)_j$ be the jth bit of c_i and let $(s_i)_j$ be the jth character of s_i. We define s_i as follows: $(s_i)_j = ((c_i)_j, j)$ for $0 \leq j \leq n - 1$ and $(s_i)_n = i$.

The next step is to generate a suffix tree for s. Using the suffix tree, the longest repeated substring can be found.

Let t be a repeated substring of s. t occurs twice in s, so every character in t occurs at least twice in s. The characters in the set $\{0, 1, 2, ..., l(n) - 1\}$ each occur exactly once in s, so t does not contain any of these characters.

We have seen that t consists only of characters in $\{(a, b) : a \in \{0, 1\}$ and $b \in \{0, 1, 2, ..., n - 1\}\}$. Since the only contiguous runs of these characters are the regions from the first to the nth character of the strings s_i, we know that t must be entirely contained in the first n characters of s_i for some i between 0 and $l(n) - 1$.

Note that for each i, each of the first n characters of s_i is different. This is because these n characters are ordered pairs where the second term is the position of the character in s_i; no two characters are in the same position, so no two characters have the same second term, and so no two characters are equal. Thus, since no character of s_i is repeated, t occurs as a substring of s_i at most once. As a result, t is a substring of at least two of the strings s_i.

Consider any character in t. The character is an ordered pair whose second coordinate is the index of the character in whichever s_i it is a substring of. The other coordinate is the bit of c_i in the same position. Thus, since t is a substring of at least two of the strings s_i, it corresponds with an identical run of bits in identical positions in 2 ciphertexts. In fact, the length of t is the length of the run of bits.

We have shown that every repeated substring of s corresponds with (and shares the length of) an identical run of bits in identical positions in two of the ciphertexts. It is easy to see that every run of bits that occurs in the same positions of two ciphertexts will result in an identical run of characters in the two s_i which correspond with those ciphertexts. As a result, the longest repeated substring of s, call it T, which can be found using the suffix tree, will correspond with the longest identical run of bits in identical positions of two ciphertexts.

The previous parts show that with high probability, two ciphertexts contain a long common run of identical bits in identical positions if and only if those two ciphertexts use the same pad. The problem guarantees that one instance of pad reuse is included, so the two ciphertexts which share a pad are expected to have the longest identical run of bits in identical positions of every pair of ciphertexts. If the two ciphertexts sharing a pad are c_{i_1} and c_{i_2}, this tells us that T is a substring of s_{i_1} and of s_{i_2} and of no other s_i.

At this point in the algorithm, we simply need to know which of the s_i contains T as a substring.

The characters of T specify which index of s_i they should be found at, so we can simply go through the correct positions of every s_i to determine the two indices i_1 and i_2 for which T is a substring of s_{i_1} and of s_{i_2}. Once we find those indices, we can conclude that the two ciphertexts sharing a pad are c_{i_1} and c_{i_2}.

We wish to show that the above algorithm runs in $O(N \log N)$ time where $N = l(n)n$ is the length of the ciphertexts combined.

The first thing the algorithm does is compute s. s is a string of length $l(n) \times (n+1) < 2nl(n) = 2N$. Each character is an element of Σ, which each take up approximately $\log l(n) < \log N$ bits. Thus s takes up $2N \log N = O(N \log N)$ space. The computation of s itself is simple, and requires only keeping track of two indices and using some simple logic to determine which character of Σ to include next. The dominating time cost of computing s comes from needing to write it all down, and this takes $O(N \log N)$ time.

The next action is to create a suffix tree for s and find T, the longest repeated substring of s. We were given that this takes $O(|s| \log |s|)$ time. Since $N < |s| = l(n) \times (n+1) < 2N$, and since $|s| = \Theta(N)$, this part of the algorithm is also $O(N \log N)$.

After that, we can look up the range of indices of s_{i_1} and s_{i_2} which comprise T. This information is encoded in the characters of T. Looking through the different s_i, we can find the two correct indices by comparing T with the proper region of s_i for each i. This requires at most one comparison (which can take $O(\log N)$ time) per character of s, so again, this requires $O(N \log N)$ time.

The indices found are the indices of the ciphertexts which share a pad. As you see, the entire algorithm runs in $O(N \log N)$ time, as desired.