6.857 Homework April 6, 2014	**Problem Set 1**	# 1-3 - Detecting Pad Reuse

(a) Suppose the length of the longest substring of both c_i, c_j is $L(i,j)$. Then, the probability that the length of the longest repeated substring is larger than $\log n \log\ln n$ is,

$$P[\exists i \neq j \text{ s.t. } L(i,j) > \log n \log\ln n] \leq \sum_{i \neq j} P[L(i,j) > \log n \log\ln n] \tag{1}$$

Suppose c_i, c_j has a common substring s with length l , s starts at $c_i[k]$ and $c_j[m]$ and assume $m \geq k$ without loss of generality. Then the substring of c_i, $c_i[k] - c_i[k+l-1]$, is the same as the substring of c_j, $c_j[m] - c_j[m+l-1]$. This is equivalent to $c_i[k : k+l-1] \bigoplus c_j[m, m+l-1]$ is all 0, and in consequence, $c_i \bigoplus (c_j \gg (k-m))$ has l consecutive 0's where $c_j \gg (k-m)$ means shifting c_j to left by $(k-m)$ bits. Thus, that c_i, c_j share a substring with length more than l is equivalent to that there exists t such that $c_i \bigoplus (c_j \gg t)$ or $(c_i \gg t) \bigoplus c_j$ contains l consecutive 0's. Since c_i and c_j are encrypted by independent pads, $c_i \bigoplus (c_j \gg t)$ and $(c_i \gg t) \bigoplus c_j$ are two uniformly randomly chosen bit strings. Thus, according to the result of longest run of heads, with high probability, the longset run of 0's in $c_i \bigoplus (c_j \gg t)$ and $(c_i \gg t) \bigoplus c_j$ is less than $\log n + \log\ln n$. That is to say,

$$\text{for all } \alpha, \text{ there exists some constant } c_\alpha, \text{ such that,}$$

$$P[\text{the longset run of 0's in } c_i \bigoplus (c_j \gg t) > \log n + \log\ln n] < \frac{c_\alpha}{n^\alpha}$$

and

$$P[\text{the longset run of 0's in } (c_i \gg t) \bigoplus c_j > \log n + \log\ln n] < \frac{c_\alpha}{n^\alpha}$$

Thus,

$$\begin{aligned}
&P[L(i,j) > \log n + \log\ln n] \\
&\leq P[\exists t \text{ such that the longset run of 0's in } (c_i \gg t) \bigoplus c_j > \log n + \log\ln n] \\
&+ P[\exists t \text{ such that the longset run of 0's in } (c_i \gg t) \bigoplus c_j > \log n + \log\ln n] \\
&\leq \sum_t P[\text{the longset run of 0's in } c_i \bigoplus (c_j \gg t) > \log n + \log\ln n] \\
&+ \sum_t P[\text{the longset run of 0's in } (c_i \gg t) \bigoplus c_j > \log n + \log\ln n] \\
&\leq 2n \cdot \frac{c_\alpha}{n^\alpha} \\
&= \frac{2c_\alpha}{n^{\alpha-1}}
\end{aligned}$$

Pluging the above inequality into (1), we get,

$$P[\exists i \neq j \text{ s.t. } L(i,j) > \log n \log\ln n] \leq \frac{l(l-1)}{2} \cdot \frac{2c_\alpha}{n^{\alpha-1}} = \frac{l(l-1)c_\alpha}{n^{\alpha-1}}$$

Since it is true for all α and l is poly(n), for any β, there exists c_β such that

$$P[\exists i \neq j \text{ s.t. } L(i,j) > \log n \log\ln n] \leq \frac{c_\beta}{n^\beta}$$

Therefore, with high probability, the longest repeated substring has length less than $\log n + \log\ln n$.

(b) If we interpret the result in bits, each English character is 8 bits, so the length of common substring will be 8 times the previous data, while $\log n$ will be essentially the same. We can pick some data point to analyze, $(5, 1)$ will become $(8, 8)$, $(10, 2.5)$ will become $(13, 20)$, $(15, 5)$ will become $(18, 40)$, $(20, 8)$ will become $(23, 64)$. We can easily see that as n gets bigger, the ratio of the length of longest run and $\log n$ is increasing, and will become bigger than 2.2 when n gets larger. That is to say, the length of identical strings in English text will be much longer than random cyphertext. This might because English text is not randomly chosen and there is some inference between words and sentences.

(c) Concatenate all strings together while adding a special character \$ to separate them. For example, if we have 1011 and 0011, the concatenated string will be 1011\$0011. Denote the concatenated string by S, We claim that it we can find the longest repeated string in S, we can solve the problem.

According to the assumption, each pair of plaintexts does share a long common run of identical characters, and any pair of ciphertexts with independently chosen pads does not. Thus, given poly(n) n-bit ciphertexts with total length N and one instance of pad reuse, the two ciphertexts using the same pad will have a common run of identical bits with length more than $2.2 \log n$ (according to the result of 2). Thus, the longest repeated substring S should be longer than $2.2 \log n$. Suppose we find the longest repeated substring S and the two identical substring is a, b.

If a and b is not overlapped, they should both contain \$ or both not. If they do contain \$, suppose a is divided to a_1, a_2 by \$ and b is divided to b_1, b_2 by \$. Then, a_1, b_1 must be identical and a_2, b_2 must be identical. At least one group (suppose a_1, b_1) has length larger than $1.1 \log n$, so with high probability, the two ciphertexts containing a_1, b_1 share the same key. If both of them do not contain \$, for the same reason, the two ciphertexts a and b should share the same key.

If a and b is ovelapped, they should not contain \$, otherwise \$ will appear twice within n bits. So a, b are in the same ciphertext. We claim that this will not happen with high probability. Suppose the ciphertext is s and the distance between a, b is k. Then, s and $s \gg k$ share an identical substring at identical locations, which indicates, $s \bigoplus (s \gg k)$ contains a long run of 0's. Suppose s is the cyphertext of x encrypted by a key p, then $s = x \bigoplus p$. Thus,

$$s \bigoplus (s \gg k) = (x \bigoplus p) \bigoplus ((x \bigoplus p) \gg k) = x \bigoplus (x \gg k) \bigoplus p \bigoplus (p \gg k)$$

Suppose $p = p_1 \dots p_n$ and $m = p \bigoplus (p \gg k) = m_1 \dots m_n$, then

$$m_i = p_i \bigoplus p_{i-k}$$

We can see that $P[m_i = 0 | p_1, \dots, p_i - 1] = 0.5$ for all i since p_i is independent of all p_j, $j < i$. Thus, m is uniformly distributed on all bit strings with length n. According to part a), with high probability, the longest run of 0's in $(x \bigoplus (x \gg k)) \bigoplus m$ should be shorter than $\log n + \log \ln n$, which indicates that a and b is overlapped and longer than $2.2 \log n$ happens with very low probability.

Therefore, we have shown how to transfer the original problem into finding the longest repeated substring in S. Finding the longest repeated substring in S is easy using suffix tree. We just need to find the internal node which has longest path from the root, which can be done just by traversing the suffix tree. Building the suffix tree for S takes $\widetilde{O}(N)$ time, traversing it also takes $O(N)$ time. Thus, the whole algorithm takes $\widetilde{O}(N)$ time.

6.857 Network and Computer Security

Spring 2014