T-79.159 Cryptography and Data Security **Spring 2003**
Tutorial 2
Tue 11.2.2003 12–14 T3 and Wed 12.2.2003 18–20 T1

1. **Let $H(x)$ denote the operation of computing a 160-bit "random" hash from every possible 160-bit input. How many of the 2^{160} possible output values can be expected to be used ?**

   The probability of $H(x) = c$ is 2^{-160} for each value of x and c. Hence the probability of $H(x) = c$ for any given c is is $1 - (1 - 2^{-160})^{2^{160}}$. We see that 2^{160} is a "very big number", and from basic analysis we know that $\lim_{x\to\infty}(1 - \frac{1}{x})^x = e^{-1}$. Hence we can expect about $(1 - e^{-1})2^{160}$ or 63.21% of the output values to occur.

2. **The Davies-Meyer mode works as follows. Use a block cipher $E(K, P)$; K is the key and P is the plaintext. Start with some initial value X_0 and update as $X_{i+1} = E(M_i, X_i) \oplus X_i$. Final value X_n is the hash.**

   We ignore the final padding of message with its length. Is there some way of breaking this construction ? (i.e. of finding collisions)

   Since $E(K, P) = C$ is a block cipher, there is an easily computable inverse function $E^{-1}(K, C) = P$, corresponding to decryption. By utilizing this decryption function, we can find $X_0 = E^{-1}(M, 0)$ by "decrypting a zero block" with some message block M. Now we see that $X_1 = E(M, X_0) \oplus X_0 = X_0$, and therefore message can be made arbitrarily long while still retaining the same message digest. We have found collisions: $X_0 = H(K) = H(K \mid K) = H(K \mid K \mid K) = \cdots$

3. **Devise a method of finding collisions, i.e. a pair $H(x) = H(y)$, where $x \neq y$, so that not too much memory is required by the method.**

   We describe a variant of Floyd's cycle finding algorithm. We define a sequence $x_{i+1} = H(x_i)$. We start from the pair (x_1, x_2) and iteratively compute (x_i, x_{2i}) from the previous pair (x_{i-1}, x_{2i-2}), until $x_m = x_{2m}$ for some m. If the tail of the sequence has length λ and the cycle has length μ, then the first time $x_m = x_{2m}$ is when $m = \mu(1 + \lfloor \lambda/\mu \rfloor)$.

   [picture of cycle and tail]

   After $x_m = x_{2m}$ has been found, it is very likely that $2m = m + \mu$ and thus we have the cycle length μ. Another λ steps is required to find first occurence of $x_i = x_{i+\mu}$, where we start from the beginning and step $i = 0, 1, 2, \cdots$. Then x_{i-1} and $x_{i+\mu-1}$ map to the same output value and constitute a collision.

   It can also happen $\lambda = 0$, which can be easily detected and the algorithm can be started from new start position. However, since $\lambda < m \leq \lambda + \mu$, and because of the Birthday paradox, this algorithm has running time $O(\sqrt{n})$, and requires neglible memory.

4. **Devise a method of producing secure hashes of large messages with a property that when 1 random bit is changed, only $O(\log n)$ effort is required to update the hash – rather than $O(n)$. Here n is the length of message (number of message blocks). We assume the existence of a pseudorandom permutation $H(x)$, which operates in $O(n)$ time.**

   We build a binary hash tree [picture]. The height of the tree is $O(\log n)$, and hence only $O(\log n)$ operations are needed to update the tree.