Homework 2b

Jacob Doskocil

October 15th, 2018

1. Users A and B use the $Diffie\text{-}Hellman$ key exchange technique with a common prime $q = 71$ and a primitive root $\alpha = 7$.

   (a) If user A has a private key $X_A = 5$, what is A's public key Y_A?
   $Y_A = \alpha^{X_A} \mod q = 7^5 \mod 71 = 51$

   (b) If user B has a private key $X_B = 12$, what is B's public key Y_B?
   $Y_B = \alpha^{X_B} \mod q = 7^{12} \mod 71 = 4$

   (c) What is the shared secret key?
   $K_S = \alpha^{X_A \times X_B} \mod q = Y_A^{X_B} \mod q = Y_B^{X_A} \mod q$
   $K_S = 7^{5\times 12} \mod 71 = 51^{12} \mod 71 = 4^5 \mod 71 = 30$

   (d) In the $Diffie\text{-}Hellman$ protocol, each participant selects a secret number x and sends the other participant $(\alpha^x \mod q)$ for some public number α. What would happen if the participants sent each other $(x^\alpha \mod q)$ instead?
   The key exchange would not work in this situation. The key exchange works because the base is the same for each operation, and because of the fact that
   $a^{b\times c} \mod d \equiv (a^b)^c \mod d \equiv (a^c)^b \mod d$

2. A network resource X is prepared to sign a message by appending the appropriate 64-bit hash code and encrypting that hash code with X's private key as described in class (also in the textbook, Page 330).

   (a) Describe the *Birthday Attack* where an attacker receives a valid signature for his fraudulent message?
   The Birthday Attack works with the attacker generating a large quantity of valid messages and a large quantity of versions of his fraudulent message. Seeing as each fraudulent message is compared to every single valid message, the attacker is able to get a fraudulent message with a valid signature much quicker than brute force.

   (b) How much memory space does attacker need for an M-bit message?
   Each message requires $(M + 64)$ to store. In order to *guarantee* two messages having the same hash, the attacker would need to generate $2^{64} + 1$ messages (assuming an ideal hash function). However the expected value needed is much lower. Given valid message M_{Vi} and fraudulent message M_{Fj}, and

$$x_{vf} = \begin{cases} 1 & \text{if } M_{Vi} = M_{Fj} \\ 0 & \text{else} \end{cases}$$

and the attacker has generated k fraudulent messages and valid messages and the hash has n bits, then
$E[x] = \sum_{i=1}^{k} \sum_{j=1}^{k} E[x_{vf}] = \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{1}{2^n} = \frac{k^2}{2^n}$.
For an expected value of 1 and $n = 64$, we need $k = 4,294,967,296$. This means that the attacker needs to and store k valid messages, and k invalid messages, so the attacker needs $2k(M + 64)$ bits of storage. This comes out to about $1.074 \times (M + 64)$ GB of storage

(c) Assuming that attacker's computer can process 2^{20} hash/second, how long does it take at average to find pair of messages that have the same hash?
The computer needs $2k = 8,589,934,592$ messages on average. If the computer can process 2^{20} hashes in a second, it takes 8192 seconds to find a match. This is 2 hours, 16 minutes, and 32 seconds.

(d) Answer (b) and (c) when 128-bit hash is used instead.
The new messages take $M + 128$ bits per message. The increased number of bits per hash means that more messages are needed in order to find matching messages. The same formula can be used $1 = \frac{k^2}{2^n}$ where $n = 128$. This gives $k = 1.84467 \times 10^{19}$ so we need 3.68934×10^{19} total messages, so we need $4.61167(M + 128)$ Exabytes of storage for all of the messages. In addition, if the attacker can process 2^{20} hashes per second, this will take 3.51844×10^{13} seconds, this is about 1115689 years.

3. Use Trapdoor Oneway Function with following secrets as described in lecture notes to encrypt plaintext P = '0101 0111'. Decrypt the resulting ciphertext to obtain the plaintext P back. Show each step to get full credit.

$$S = \{5, 9, 21, 45, 103, 215, 450, 946\}$$
$$a = 1019,\ p = 1999$$

The set t can easily be calculated using the formula $t_i = a * S_i \mod p$. This gives the set

$$t = \{1097, 1175, 1409, 1877, 1009, 1194, 779, 456\}$$

Since the message being encrypted is '0101 0111' the encrypted message is
$Y = 1175 + 1877 + 1194 + 779 + 456 = 5481$
The inverse of $a \mod p$ is the secret key for the receiver. This is $1019^{-1} \mod 1999 = 1589$ (using the `invert.cpp` program in the repository). When the receiver gets the message, they decrypt it by calculating $Z = a^{-1} * Y \mod p = 1589 * 5481 \mod 1999 = 1665 \mod 1999$. Then the knapsack problem is solved using the original S set. The problem is solved by a greedy algorithm that takes the largest item that is less than the total, removes it from the set, and subtracts it from the total, and then reiterates until the 'knapsack' has been filled (the remaining total is 0). The iteration for this solution is shown below.
$Y = 1665,\ S = \{5, 9, 21, 45, 103, 215, 450, 946\} \rightarrow 1665 - 946 = 719$
$Y = 719,\ S = \{5, 9, 21, 45, 103, 215, 450\} \rightarrow 719 - 450 = 269$
$Y = 269,\ S = \{5, 9, 21, 45, 103, 215\} \rightarrow 269 - 215 = 54$
$Y = 269,\ S = \{5, 9, 21, 45, 103\} \rightarrow 54 - 45 = 9$
$Y = 269,\ S = \{5, 9, 21, 103\} \rightarrow 9 - 9 = 0 \rightarrow$ message decrypted.
This gives the successful decryption of $Z = 9 + 45 + 215 + 450 + 946$. Using the set S, this gives the plaintext P ='0101 0111'.