# Homework 2

## Due: 7:00PM, Saturday, 11/7/2015

Name: ID:

1. Keyed hash cannot be used for encryption because it takes an arbitrary-length string and maps it to a fixed-length string. If we restrict the input to be $n$ bits and output also $n$ bits so that the input space and the output space are equal, can it be used for encryption? (5 Points)

   **Keyed hash can not be used for encryption even if the size of output is the same as the size of input because it does not guarantee one-to-one mapping. Hash is one-way function and it is computationally infeasible to find a message that has a given message digest. In order words, keyed hash algorithm is noninvertible and decryption cannot practically find out what message corresponds to a given message digest.**

2. Existing message digests are reasonably fast, but here is a much faster function: take your message, divide it into 128-bit chunks, and XOR all the chunks together to get a 128-bit result. Then, perform the standard message digest on the result. Is this a good message digest function? (5 Points)

   **No. It is fairly easy to generate another message with the same 128-bit result.**

3. How to use a hash algorithm for encryption and how to use a secret key cipher for hashing? (10 Points)

   **Slides 13, 15, 16**

4. Most viruses infect your system by implanting themselves into the existing executable files on the disk. Explain how to use a hash algorithm to design a virus detector, which identifies the files that may be infected by viruses (10 Points)

   **A virus detector may generate the file digests by applying a hash algorithm on the files and then stores the file digests securely. Then the virus detector periodically computes the file digests and compares them with the stored version. If a virus changes the content of a file, the new digest will be different from the original digest. In this way, a virus detector can detect the modification of a file by a virus.**

5. Assume a good 128-bit message digest function. Assume there is a particular value, $d$, for the message digest and you want to find a message that has a message digest of $d$. Given that there are many more 2000-bit messages that map to a particular 128-bit message digest than 1000-bit messages, would you theoretically have to test fewer 2000-bit messages to find one that has a message digest of $d$ than if you were to test 1000-bit messages? (5 Points)

   **No. The expected number of messages you need to try is $2^{128}$ in either case.**

6. For RSA key generation, given $p$ = 23, $q$ = 17, and $e$ =5, calculate $d$. Give the process of calculation. (10 points)

   $n = pq = 391$
   $\varphi(n) = (p-1)(q-1) = 22 \times 16 = 352$

   $\frac{352}{5} = 70 \dots 2 \qquad 2 = 352 - 5 \times 70$
   $\frac{5}{2} = 2 \dots 1 \qquad 1 = 5 - 2 \times 2 = 5 - (352 - 5 \times 70) \times 2 = 5 \times 141 - 352 \times 2$

   $d = 141$

7. In RSA, given that the primes $p$ and $q$ are approximately the same size, approximately how big is $\varphi(n)$ compared to $n$? (10 points)

   $$\varphi(n) = (p-1)(q-1) = pq - p - q + 1 \approx n - 2\sqrt{n} = \left(1 - \frac{2}{\sqrt{n}}\right)n$$

8. Suppose Fred sees your RSA signature on $m_1$and on $m_2$ (i.e. he sees $m_1^d$ mod $n$ and $m_2^d$ mod $n$). How does he compute the signature on each of $m_1^j$ mod $n$ (for positive integer $j$), $m_1^{-1}$ mod $n$, $m_1.m_2$ mod $n$, and in general $m_1^j \cdot m_2^k$ mod $n$ (for arbitrary integers $j$ and $k$)? (20 Points)

   $(m_1{}^j)^d$ **mod** $n$ = $(m_1{}^d)^j$ **mod** $n$, ***so to compute your signature on*** $m_1{}^j$ **mod** $n$, ***Fred just raises your signature on*** $m_1$ ***to the jth power,*** **mod** $n$.

   $(m_1{}^{-1})^d$ **mod** $n = (m_1{}^d)^{-1}$ **mod** $n$, ***so to compute your signature on*** $m_1{}^{-1}$ **mod** $n$, ***Fred just computes the inverse*** **mod** $n$ ***of your signature on*** $m_1$.

   $(m_1 \cdot m_2)^d$ **mod** $n = m_1{}^d \cdot m_2{}^d$ **mod** $n$, ***so to compute your signature on*** $m_1 \cdot m_2$ **mod** $n$, ***Fred just multiplies your signature on*** $m_1$ ***by your signature on*** $m_2$, **mod** $n$.

   ***So for the general case of*** $m_1{}^j \cdot m_2{}^k$ **mod** $n$, ***Fred gets your signature on*** $m_1{}^{\operatorname{sgn} j}$ **mod** $n$ ***and raises it to the*** $|j|$***th power,*** **mod** $n$, ***then gets your signature on*** $m_2{}^{\operatorname{sgn} k}$ **mod** $n$ ***and raises it to the*** $|k|$***th power,*** **mod** $n$, ***and finally multiplies the results together,*** **mod** $n$. [ $\operatorname{sgn} x = x/|x|$ ]

9. One solution to Man-in-the-Middle Attack over Diffie-Halleman key exchange is encrypting the Diffie-Hellman value with the other side's public key. Why is this the case? (5 Points)

   **The attacker will not be able to decrypt the Diffie-Hallman values sent to him and so will not be able to compute the shared secrets.**

10. Can you modify the encryption with hash specified in Mixing In the Plaintext so that instead of $b_i$ = MD($K_{AB}|c_{i-1}$) we use $b_i$ = MD($K_{AB}|m_{i-1}$)? How do you decrypt it? Why wouldn't the modified scheme be as secure? (Hint: what would happen if the plaintext consisted of all zeroes?) (20 points)

    $\boldsymbol{m_i = c_i \oplus b_i}$. **So each** $\boldsymbol{m_i}$ **can be determined before it is needed to compute** $\boldsymbol{b_{i+1}}$.

    **This scheme is not secure, because patterns in the plaintext will produce corresponding patterns in the ciphertext. For example, when the plaintext consists of all zeros,** $\boldsymbol{b_i}$ **and** $\boldsymbol{c_i}$ **are the same for all** $\boldsymbol{m_i}$ **when** $\boldsymbol{i \geq 2}$.