Cryptography and Architectures for Computer Security

Exam Code: 095947 (old 090959), A.Y. 2014–2015, Semester: 2

Prof. G. Pelosi

July 22nd, 2015 – Exam Session

Name:.................................... Surname:..

Student ID:.............................. Signature:..

Time: 2h:30'. Use of textbooks, notes, phones or Internet connected devices is not allowed. Prior to turn in your paper, write your name on any additional sheet and sign it.

Question 1 [3 pts]

Consider a Vigenère Cipher. Each (lower-case) letter of the English alphabet is put into one-to-one correspondence with the integer denoting its position, i.e.: a=0, b=1, ..., z=25. The key $k{=}(k_1, k_2, \ldots, k_m)$, $0{\le}k_i{\le}\ 25$ is composed by $m \geq 2$ letters and is employed to perform a block encryption of the plaintext message $x{=}(x_1, x_2, \ldots, x_m, x_{m+1}, \ldots)$, with $0{\le}x_i{\le}\ 25$, as follows:

$$c{=}(x_1{+}k_1 \bmod 26,\ \ldots,\ x_m{+}k_m \bmod 26,\ x_{m+1}{+}k_1 \bmod 26,\ \ldots\ )$$

- Show how to apply a *Known Plaintext Attack*
- Describe how to execute a *Ciphertext-Only Attack*
- How is it possible to design a *Perfectly Secure Cipher* employing a Vigenère Cipher?

Solution:
see lectures, or book, ...

Question 2 [3 pts]

Let p be a large prime and g be a generator of $(\mathbb{Z}_p^*, \cdot)$. Suppose we are considering the function $h : \mathbb{Z} \mapsto \mathbb{Z}_p^*$ for use as a hash function, where $h(x) = g^x \bmod p$, assuming x to be an arbitrary binary string interpreted as a positive integer number, i.e.: $x{\in}\mathbb{Z}$, $x{\ge}0$.
Note that the *compression property* of typical hash functions is satisfied by the above definition as messages x of arbitrary bit-length are hashed into a fixed size digest.

Assuming the computational difficulty of the discrete logarithm problem in $(\mathbb{Z}_p^*, \cdot)$, which properties of cryptographic hash functions does h satisfy? Discuss if h is a sound cryptographic hash function or not.

Solution:
The *first pre-image* property is guaranteed by the difficulty of the discrete log problem. The *second pre-image* property is not satisfied by the proposed construction as given x and $h(x)$,

it is trivial to compute $x'=x+l(p-1)$, with $l=\{1,2,3,\ldots,\}$ such that $h(x')=h(x)$. The *collision resistance* property is also not satisfied as two messages with the same digest can be found picking them in the set $\{x \text{ s.t. } x=x_0+l(p-1),\ l \geq 1,\ \forall\ x_0 \geq 0\}$ ...

Question 3 [6 pts]
Threefish is a block cipher with a 256-bit (32 bytes) sized block and allows the use of three different key lengths: 256, 512 and 1024 bits. Consider the three following password hashing strategies relying on Threefish:

(a) Split the password p in 9-byte wide blocks $p_0 \ldots p_n$, and generate $n+1$ 23-byte unpredictable random salt blocks s_i, $0 \leq i \leq n$. Store on the disk a sequence of blocks $h_i=\textsc{Threefish}_{p_i||s_i}(p_i||s_i)$ followed by the concatenation of all the s_i, i.e.: $h_0||h_1||\ldots||h_n||s_0||s_1||\ldots||s_n$.

(b) Split the password p in 9 bytes wide blocks $p_0 \ldots p_n$, and generate $n+1$ 23-byte salt blocks s_i obtained as the concatenation of a 1-byte random value for each s_i. Store on the disk a sequence of blocks $h_i=\textsc{Threefish}_{p_i||s_i}(p_i||s_i)$ followed by the concatenation of all the s_i.

(c) Split the password p in 32-byte wide blocks $p_0 \ldots p_n$, obtain a 1024-bit unpredictable random salt s and store on the disk a sequence of blocks h_i, with
$h_0=s$ and $\forall i>0, h_i=\textsc{Threefish}_s(p_{i-1} \oplus h_{i-1})$

Assume you have 16 TiB of disk space available (average access time 1ms), 32 GiB of RAM (average access time 0.1μs) and you are able to compute 2^{32} Threefish encryptions or decryptions per second on a good GPU.

State how much computational effort (i.e., how much time) is required to break the proposed password hashing schemes with the best possible technique, and whether this is practically feasible considering only fully lowercase passwords.

Solution:

(a) This password strategy effectively cuts down the computational effort to perform a bruteforce to the one required to find each p_i exhaustively. The keyspace of a 9 character lowercase password is $\approx 26^9=(2^{4.7})^9=2^{42}$, thus amounting to around 2^{10} seconds in computation time i.e., ≈ 20 minutes, which is feasible. The presence of a 23-byte (184-bit) unpredictable salt however prevents time-to-memory-tradeoffs (TMTOs) to be employed.

(b) This password strategy suffers from a key recovery attack in the same fashion as the previous one. However, considering the fact that the possible values for the salt are limited to 2^8, it is possible to apply a TMTO strategy to reduce the time to break a password after a precomputation effort of 2^{18}s, that is, around 72 hours. Employing a rainbow table strategy, we know that besides the initial computation, the worst time to break a password is given by l encryptions and l rainbow table lookups. Assuming that the chains are stored in such a fashion to allow $O(log(n))$ access, the following efforts are required: for disk stored rainbow tables, 2^{38} chains can be stored, resulting in a chain length of 2^{12}, and thus $\approx 1\mu$s in computation time and $2^{12}*38*1$ms (≈ 15.5s)for the disk lookups. For RAM stored rainbow tables, 2^31 chains are stored, resulting in a 2^{19} chain length yielding a ≈ 122ms computation and a $2^{19}*31*0.1\mu$s lookup time (≈ 1.6s). The best strategy to break this password hashing thus retrieves one password in 1.6 seconds employing RAM housed rainbow tables.

(c) This password hashing strategy is effectively encrypting the password employing THREEFISH in CBC mode, using the salt as both the IV and the key. Since the salt is stored in cleartext together with the hashed password, retrieving the password is just a matter of decrypting it in negligible time.

Question 4 [6 pts]
Show if the irreducible polynomial $m(x){=}x^8{+}x^4{+}x^3{+}x{+}1{\in}\mathbb{F}_2[x]$ specified in the Advanced Encryption Standard (AES) to represent the elements of the finite field $\mathbb{F}_{2^8}$ and employed in the S-BOX construction, is also a primitive polynomial.

Solution:
$|\mathbb{F}^*_{2^8}|{=}255{=}3 \cdot 5 \cdot 17$
Assuming $\alpha{\in}\mathbb{F}^*_{2^8}$ as both a primitive element and root of $m(x)$, $m(x)$ is a primitive polynomial if the following relations hold: $\alpha^8 \equiv \alpha^4 + \alpha^3 + \alpha + 1$

$$\begin{cases} \alpha^3 \not\equiv 1 \\ \alpha^5 \not\equiv 1 \\ \alpha^{15} \not\equiv 1 \\ \alpha^{17} \not\equiv 1 \\ \alpha^{51} \not\equiv 1 \\ \alpha^{85} \not\equiv 1 \end{cases}$$

Note that:
$$\alpha^{15} \equiv \alpha^8 \cdot \alpha^7 \equiv (\alpha^4 + \alpha^3 + \alpha + 1) \cdot \alpha^7 \equiv \alpha^{11} + \alpha^{10} + \alpha^8 + \alpha^7 \equiv$$
$$\equiv (\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3) + (\alpha^6 + \alpha^5 + \alpha^3 + \alpha^2) + (\alpha^4 + \alpha^3 + \alpha + 1) + \alpha^7 \equiv$$
$$\equiv \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$$

$$\alpha^{17} \equiv \alpha^{15} \cdot \alpha^2 \equiv \alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2$$

$$\alpha^{51} \equiv (\alpha^{17})^2 \cdot \alpha^{17} \equiv (\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2)^2 \cdot (\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2) \equiv$$
$$\equiv \cdots \equiv 1$$
Therefore, we can conclude that $m(x)$ is not primitive.

Question 5 [7 pts]
Consider the cyclic group $(\mathbb{Z}^*_{53}, \cdot)$ with one of its generators being $g{=}2$.

(a) Show the order of the group and exhibit at least one generator for each of its subgroups.

(b) Consider the following two discrete logarithms, x_0, x_1, and state if each of them exists, motivating your answer:
$$x_0 = \log^{\mathsf{D}}_g(101^{101})$$
$$x_1 = \log^{\mathsf{D}}_{g^{20}}(2)$$

(c) Compute the logarithms which you stated to be existing.

(d) Consider a generic cyclic group of the form $\mathrm{G}{=}(\mathbb{Z}^*_{p^k}, \cdot)$, with $p{\geq}2$ a prime integer, and $k{\geq}1$. What is an efficient and secure choice for p and k to properly set up the public parameters of a Diffie-Hellman protocol?

Solution:

(a) $g = 2$
$|(\mathbb{Z}^*_{5^3}, \cdot)| = \varphi(5^3) = 100$
There are 9 subgroups of $(\mathbb{Z}^*_{5^3}, \cdot)$ with order $1, 2, 4, 5, 10, 20, 25, 50, 100$, respectively.
Their generators are:
$g_0 = 1$, this is the generator of the trivial subgroup composed by the neutral element only.
$g_1 = g^{\frac{100}{2}} \equiv_{125} g^{50} \equiv_{125} 124 \equiv_{125} -1$
$g_2 = g^{\frac{100}{4}} \equiv_{125} g^{25} \equiv_{125} 57$
$g_3 = g^{\frac{100}{5}} \equiv_{125} g^{20} \equiv_{125} 76$
$g_4 = g^{\frac{100}{10}} \equiv_{125} g^{10} \equiv_{125} 24$
$g_5 = g^{\frac{100}{20}} \equiv_{125} g^{5} \equiv_{125} 32$
$g_6 = g^{\frac{100}{25}} \equiv_{125} g^{4} \equiv_{125} 16$
$g_7 = g^{\frac{100}{50}} \equiv_{125} g^{2} \equiv_{125} 4$
$g_8 = g^{\frac{100}{100}} \equiv_{125} g \equiv_{125} 2$

(b), (c) The base of the first logarithm is the generator of $(\mathbb{Z}^*_{125}, \cdot)$, therefore it exists for sure. $x_0 \equiv_{100} \log^{\mathsf{D}}_g(101^{101}) \equiv_{100} \log^{\mathsf{D}}_2(101^{101}) \equiv_{100} \log^{\mathsf{D}}_2(101) \equiv_{100}$
$\equiv_{100} \ldots \text{ Baby-Step/Giant-step } \ldots \equiv_{100} 60$

The base of the second logarithm g^{20} is the generator of the subgroup with order equal to 5. The argument of the second logarithm is the generator of the whole group, therefore the logarithm x_1 does not exists. Indeed, if $g = 2$ was in the subgroup generated by g^{20}, then it would be true that $\langle g^{20} \rangle = \langle g \rangle$ (which is clearly impossible).

(d) see lectures ...

Question 6 [3 pts]
Consider a textbook RSA scheme, with $n{=}pq$ being a product of two different primes and the relation between the public exponent $e{\in}\mathbb{Z}^*_{\varphi(n)}$ and the private exponent $d{\in}\mathbb{Z}^*_{\varphi(n)}$ such that $ed \bmod \varphi(n){=}1$.

(a) Show the correctness of the identity

$$m^{ed} \equiv_n m \qquad \text{with} \ \ m \in \mathbb{Z}_n$$

which states that we obtain the same message after encryption and decryption (or viceversa)

(b) Consider an RSA public modulus computed as the square of a prime number, i.e.: $p{=}q$ and $n{=}p^2$. Show a simple example for the fact that, in this case, the condition $ed \bmod \varphi(n){=}1$ does not imply $m^{ed} \equiv_n m, \ \forall \ m{\in}\mathbb{Z}_n$.

Solution:
see lectures ...
if $n = p^2$, and $m \notin \mathbb{Z}^*_n$, this means that $m{=}u \cdot p$ for a proper choice of the integer u.
Therefore, $m^{ed} \equiv_n m^{l\varphi(n)+1} \equiv_n m^{lp(p-1)+1}$, for any integer l.
If $lp(p-1)+1 \geq 2$ then $m^{lp(p-1)+1} \equiv_n (up)^{lp(p-1)+1} \equiv_n u^2 \cdot p^2 \cdot (up)^{lp(p-1)-1} \equiv_n 0 \neq m$

Question 7 [6 pts]
Many cryptosystems such as RSA and Diffie–Hellman key exchange are based on arithmetic operations modulo a large integer number.

(a) Explain why the Montgomery multiplication method is the preferred choice for implementing the aforementioned cryptographic schemes.

(b) Describe the Montgomery multiplication technique, specifying the notions of Montgomery transformation, Montgomery reduction and the basic idea for applying it to multi-precision integer representations.

(c) Assume to work into the Montgomery domain: $(\mathbb{Z}_N, +, \times)$, $N = 21$.
Compute the Montgomery multiplication C=A×B mod N, where A=16_{decimal} and B=18_{decimal} are binary encoded values in the Montgomery domain. Show every step of the procedure.

Solution:

(a), (b) See lectures ...

(c) $\lceil \log_2 N \rceil = 5,\ R = 2^5 = 32,$
$\gcd(R, N) = RR' - NN' = 1 \Leftrightarrow \gcd(32, 21) = 32(2) - 21(3) = 1,$
$R' \overset{\text{def}}{=} R^{-1} \bmod N = 2 \bmod 21 = 2,$
$N' \overset{\text{def}}{=} N^{-1} \bmod R = 3 \bmod 32 = 3;\ N'_0 = 1$

$B = \langle B_4B_3B_2B_1B_0\rangle = \langle 10010\rangle_2$
$A = \langle A_4A_3A_2A_1A_0\rangle = \langle 10000\rangle_2$

$$\begin{array}{r l}
\mathbf{00000} & + \\
00000 & \quad A_0B = 0 \cdot \langle 10010\rangle_2 \\
\hline
00000 & + \\
00000 & \quad t\,N = (N'_0 x_0)\,N = 0 \\
\hline
00000 & \text{perform a right-shift of 1 bit} \\
 & \\
\vdots & + \\
\vdots & \quad A_iB = 0 \cdot \langle 10010\rangle_2,\ \mathbf{i = 1, 2, 3} \\
\hline
\vdots & + \\
\vdots & \quad t\,N = (N'_0 x_0)\,N = 0,\ \mathbf{i = 1, 2, 3} \\
\hline
00000 & \text{perform a right-shift of 1 bit} \\
 & \\
00000 & + \\
10010 & \quad A_4B = 1 \cdot \langle 10010\rangle_2 \\
\hline
10010 & + \\
00000 & \quad t\,N = (n'_0 x_0)\,N = 0 \\
\hline
10010 & \text{perform a right-shift of 1 bit} \\
 & \\
\mathbf{01001} &
\end{array}$$

$C = \langle 01001\rangle_2 = 9 < N \Rightarrow C = 16 \times 18 = 9 \in (\mathbb{Z}_N, +, \times)$

Validation:

C=$16 \times 18 \overset{\text{def}}{=} 16 \cdot 18 \cdot R^{-1} \bmod N$=$16 \cdot 18 \cdot 2 \bmod 21$=$-2 \bmod 15 \equiv_{21} 9$