Cryptography and Architectures for Computer Security

Exam Code: 095947 (old 090959), A.Y. 2015–2016, Semester: 2

Prof. G. Pelosi

September 28th, 2016 – 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 [**2 pts**]
Alice uses a block cipher `Enc` having 64-bit block size and 64-bit key size. She is worried that the key size is too small to prevent brute force attacks. Therefore, she decides to improve the encryption scheme employing 2 independent 64-bit keys: (k_0, k_1), and encrypting her messages as:

$$c = \texttt{Enc}_{k_0}(m) \oplus k_1, \qquad m, c \in \{0,1\}^{64}$$

- Assume that the adversary gets access to a few plaintext/ciphertext pairs and is indeed able to perform a brute-force attack on the original encryption scheme `Enc` and recover the key with a *known plaintext attack*. Show that he can also break the "improved" scheme and recover the Alice's extended key.

  Solution:
  Considering 2 pairs of plaintexts and ciphertexts (m_1, c_1), (m_2, c_2)
  for each possible value of k_0 compute $k_1 = c_1 \oplus \texttt{Enc}_{k_0}(m_1)$
  and check if $k_1 = c_2 \oplus \texttt{Enc}_{k_0}(m_2)$ holds
  The adversary needs to compute every value of $\texttt{Enc}_{k_0}(m_1)$ and $\texttt{Enc}_{k_0}(m_2)$, thus the cost for obtaining the extended key with 128-bit is only doubled with respect to a bruteforce of the 64-bit key of the original cipher.

Question 2 [**3 pts**]
Consider a file encrypted with a given mode of operation and the problem of decrypting only a single block in "the middle" of the available ciphertext (i.e., we are interested only in the n-th plaintext block, with $n>1$).

- Discuss how this can be done. Of course one can do it by decrypting the whole file, so the answer must describe how to solve the problem considering different mode of operations and point out the most efficient scenario.

  Solution:
  (Sketch)

- ECB. All blocks are independently encrypted, so we just decrypt block n.
- CBC. To decrypt block n, we only need ciphertext blocks $n-1$ and n.
- CFB. To decrypt the block n, we need to access also the $(n-1)$-th block, as the Input Shift Register (ISR) must be set to the value of the $(n-1)$-th block for decrypting the n-th one.
- OFB. To decrypt block n we need to access the ciphertext block n only, as the content of the Output Shift Register (OSR) is obtained encryption n times the `IV`.
- CTR. Similar to OFB; $Enc(\texttt{ctr}_n)$ can be computed by just one encryption, since what is encrypted is $\texttt{IV}||n$, rather than something iteratively defined as in the OFB.

Question 3 [3 pts]
We consider the possibility of using SHA-1 or SHA-2 for authentication as follows.
Bob authenticates a message m for Alice by computing $h(k||m||p)$ where $h(\cdots)$ is the hash function, k is a secret key shared between Alice and Bob, and p is padding.
Prove that this system has the (unwanted) property that the Adversary can authenticate certain messages not sent by Bob.

Solution:
Assume that Bob has authenticated message m by computing the hash value for $k||m||p$ as `h`. Then the Adversary can compute the message digest for $k||m||p||m_0||p_0$ where m_0 is any message and p_0 is the padding needed for the extended message.
He can do this since the hash functions work by splitting the message into blocks $B_1, B_2, \dots B_n$ and iterating: $H_0 = \texttt{IV}$, $H_k = g(B_k, H_{k-1})$ for $k = 1, 2, \dots, n$, where $g()$ is the compression function. Thus the adversary can compute the hash value for the extended message by starting from `h` and iterating over the blocks he added.
Note that the message authenticated in this way is $m||p||m_0$, i.e., it extends message m by m_0 but it also contains the padding p, which might be a nuisance to the Adversary.

Question 4 [6 pts]
Consider multiplicative groups: $G_1 = \mathbb{Z}^*_{89}$, $G_2 = \mathbb{Z}^*_{97}$

(a) Show that 7 is a generator for both G_1 and G_2

(b) Solve the discrete logarithm problem $\log^{\mathtt{D}}_7(2)$ in G_1 and G_2

(c) Which discrete logarithm is harder, and why ?

Solution:
(a) for G_1 the proper divisors of the order of the group are: $2, 4, 8, 11, 22, 44$:
$7^2 \equiv_{89} 49$; $7^4 \equiv_{89} 87$; $7^8 \equiv_{89} 4$; $7^{11} \equiv_{89} 37$; $7^{22} \equiv_{89} 34$; $7^{44} \equiv_{89} 88$
therefore 7 is a generator for G_1.
for G_2 the proper divisors of the order of the group are: $2, 3, 4, 6, 8, 12, 16, 24, 32, 48$:
$7^2 \equiv_{97} 49$; $7^3 \equiv_{97} 52$; $7^6 \equiv_{97} 85$; $7^4 \equiv_{97} 73$; $7^8 \equiv_{97} 91$; $7^{12} \equiv_{97} 47$;
$7^{16} \equiv_{97} 36$; $7^{24} \equiv_{97} 75$; $7^{32} \equiv_{97} 35$; $7^{48} \equiv_{97} 96$
therefore 7 is a generator for G_2.

(b) The logarithm exists in both groups as 7 can be considered as a generator in each of them, and 2 is an element of both of them
Applying the BSGS method for the Dlog in G_1:
$g = 7$, $\beta = 2$, $\log_g(\beta) \equiv_{|G_1|} (i \cdot \lceil\sqrt{|G_1|}\rceil + j)$

	0	1	2	3	**4**	5	6	7	**8**	9	10
$g^j = 7^j$	1	7	49	76	87	75	80	26	$\underline{4}$	28	18
$\beta(g^{-\sqrt{\lvert G_1\rvert}})^i = 2 \cdot 5^i$	2	10	50	72	$\underline{4}$						

$|G_1|$=$\varphi(89) = 88 = 2^3 \cdot 11$, $\quad \log_7(2) \equiv_{88} (4 \cdot 10 + 8) \equiv_{88} 48$
for the Dlog in G_2: $g = 7$, $\beta = 2$, $\log_g(\beta) \equiv_{|G_2|} (i \cdot \lceil\sqrt{|G_2|}\rceil + j)$

	0	1	2	3	**4**	5	6	7	8	**9**	10
$g^j = 7^j$	1	7	49	52	$\underline{73}$	26	85	13	91	55	94
$\beta(g^{-\sqrt{\lvert G_2\rvert}})^i = 2 \cdot 32^i$	2	64	11	61	12	93	66	75	72	$\underline{73}$	

$|G_2|$=$\varphi(97) = 96 = 2^5 \cdot 3$, $\quad \log_7(2) \equiv_{96} (9 \cdot 10 + 4) \equiv_{96} 94$

(c) $|G_1| = 88 = 2^3 \cdot 11$, $|G_2| = 96 = 2^5 \cdot 3$.
For a group with order $n = \prod_{i=1}^{s} p_i^{e_i}$, with a B-smooth factorization, the Poligh-Hellmann Dlog extraction strategy has a computational complexity equal to
$\mathcal{O}(s \cdot \max\{e_i\} \cdot (\log_2(n) + \sqrt{B}))$.

For $|G_1|$ the Poligh-Hellmann Dlog requires at most 58.6 bit operations.
For $|G_2|$ the Poligh-Hellmann Dlog requires at most 83.2 bit operations.
Therefore, in G_2 the required discrete log computation is more computationally demanding.

Question 5 [5 pts]
Consider a schoolbook RSA encryption scheme with public and private key $k_{\texttt{pub}}$=$\langle e, n\rangle$, $k_{\texttt{priv}}$=$\langle p, q, \varphi(n), d\rangle$. Given a ciphertext $c \in (\mathbb{Z}_n, +, \cdot)$, a *chaining attack* aims to recover the corresponding plaintext $m \in (\mathbb{Z}_n, +, \cdot)$ without knowing the private key and exploiting the following computations:

$$c^e \bmod n,\quad c^{e^2} \bmod n,\quad \ldots,\quad c^{e^{k-1}},\quad c^{e^k} \equiv c \bmod n$$

Thus the attacker aims to find the smallest integer k which allows him to observe a cycle.

(a) Assuming that the above mentioned k exists, show how to derive the plaintext.

(b) Explain why such a value $k \in \{1, \ldots, n-1\}$ always exists. Hint: Recall that RSA is an encryption algorithm and therefore bijective, i.e., $m_1^e \not\equiv_n m_2^e$, $\forall m_1, m_2 \in (\mathbb{Z}_n, +, \cdot)$

Solution:

(a) $c^{e^k} \equiv_n c \Rightarrow e \cdot e^{k-1} \equiv_{\varphi(n)} 1 \Rightarrow d \equiv_{\varphi(n)} e^{k-1}$, $m \equiv_n c^d$

(b) the outcome of the encryption function with a fixed public key can be seen as a permutation of the plaintext space ...

Question 6 [14 pts]

(a) Describe the Fermat primality test and the Miller-Rabin primality test.

(b) Apply the Pollard's ρ method to factor the RSA public modulus $n = p \cdot q = 1147$

(c) Let e=$49_{\texttt{dec}}$ be the public exponent of an RSA public-key k_{pub}=$\langle e, n\rangle$. Knowing the factorization of the modulus n, compute the value of the corresponding RSA private-key k_{priv}=$(p, q, \varphi(n), d)$. Show every step of the computation.

(d) Decrypt the message c=$10_{\texttt{dec}}$$\in\mathbb{Z}_n$ (provided without any padding scheme) through applying the CRT. Describe each step of the procedure.

(e) Describe the Optimized Asymmetric Encryption Padding (OAEP) scheme and the reasons to employ it in a non school-book implementation of the RSA cryptosystem.

(f) Assume to work into the Montgomery domain: $(\widetilde{\mathbb{Z}}_p, +, \times)$, p=23

- Given A=$5_{\texttt{dec}}$, B=$7_{\texttt{dec}}$, show the corresponding values in the Montgomery domain (i.e., $\widetilde{A}$, $\widetilde{B}$) justifying your answer.
- Compute the Montgomery multiplication $\widetilde{C}$=$\widetilde{A}$$\times$$\widetilde{B}$ mod p, assuming a binary encoding of the operands.
- Explain the reasons to employ a Montgomery-based arithmetic for the efficient implementation of asymmetric cryptosystems.

Solution:
(Sketch)

(a) see lectures ...

(b) $p = 31$, $q = 37$

(c) $\varphi(n) = 30 \cdot 36 = 2^3 \cdot 3^3 \cdot 5 = 1080$
$\varphi(\varphi(n)) = 2^5 \cdot 3^2 = 288$
$d = 49^{-1} \bmod \varphi(n) \equiv_{\varphi(n)} 49^{287} \equiv_{\varphi(n)} \dots \equiv_{\varphi(n)} 529$

(d) see lectures ...

(e) see lectures ...

(f) $t = \lceil \log_2 p \rceil = 5$, $R = 2^t = 2^5$... $R' \equiv_p R^{-1} \equiv_p 18$, $N' \equiv_R (-N)^{-1} \equiv_{32} 25$
$\widetilde{A}$=MMul(A, R^2) $\equiv_p A \cdot R \equiv_p 22$
$\widetilde{B}$=MMul(B, R^2) $\equiv_p B \cdot R \equiv_p 17$
$\widetilde{C}$=MMul($\widetilde{A}$,$\widetilde{B}$)= ... (the final result of the multiplication computed considering the operands in binary) ... $\widetilde{A} \cdot \widetilde{B} \cdot R^{-1} \bmod p \equiv_p 22 \cdot 17 \cdot 18 \equiv_{23} 16$
see lectures ...