Cryptography and Architectures for Computer Security

Exam Code: 095947 (old 090959), A.Y. 2016–2017, Semester: 2

Prof. G. Pelosi

July 25th, 2017 – Exam Session

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

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

Time: 2h:30'. Use of textbooks, notes, or Internet connected devices (including smartphones) is not allowed. The usage of simple calculators is allowed. Prior to turn in your paper, write your name on any additional sheet and sign it.

Question 1 [4 pts]

(a) Consider a common PKI infrastructure based on X.509 certificates. You have received a certificate for `www.site-A.com` signed by rootCA-A. Recently, rootCA-B has been compromised, but the news did not reach you.
The validity of your certificate is affected by the news in any way?
Does it change anything if you still have to retrieve the certificate for `www.site-A.com`, and you do not know yet who is signing it?

(b) What is the (approximate) total size of a 100KiB non-compressed email message encrypted for ten recipients in the OpenPGP format?

(c) Consider the case of Alice willing to send an email to Bob encrypting it with OpenPGP. Alice fetches the Bob's certificate from a public-key server. She knows that Dave signed Bob's certificate but she doesn't find Dave's signature on the certificate she retrieved.
How can she be sure that the certificate is authentic?

Solution:
see lectures...

Question 2 [4 pts]

Consider the case of a password hashing scheme employing SHA-256 as its compression function.

(a) Assuming that the passwords are composed of 8 random printable ASCII characters (there are 128 of them), compute the chain length and size of the rainbow table you would use, assuming that you have 64 GiB of RAM available.

(b) Compute the size of the salt, assuming it is composed of fully random bits, to prevent a Time-To-Memory-Tradeoff (TMTO) from being performed, assuming that computing 2^{128} SHA-256 hashes is unfeasible.

(c) Consider the following two password hashing strategies for an 8 printable ASCII characters password `p` an 8 B salt `s`. The notation `p[i:j]` indicates the portion of the password starting at byte `i` and ending at byte `j`, and the same goes for the salt. The || symbol denotes bitwise concatenation.

Strategy A: SHA-256(`p[0:2]||s[0:2]||p[3:5]||s[3:5]||p[6:7]||s[6:7]`)

Strategy B: SHA-256(`p[0:2]||s[0:2]`)||SHA-256(`p[3:5]||s[3:5]`)||SHA-256(`p[6:7]||s[6:7]`)

Justify which one of the choices is better, computing the effort to perform an exhaustive search for the password value given a hash in terms of the required number of SHA-256 hash computations.

Solution:

(a) Given that a SHA-256 digest takes 32 B and two of them should be memorized for each chain, it is possible to memorize 2^{30} chains in the available RAM. Since the entire password space is $2^{8\cdot 7} = 2^{56}$ passwords wide, each chain should be $\frac{2^{56}}{2^{30}} = 2^{26}$ hashes long.

(b) Since a TMTO needs to compute the hash of the entire password space, including the salt, at least once, and there are 2^{56} passwords, a 72b (9B) long salt is sufficient to prevent the computation.

(c) Strategy A requires the attacker to perform an exhaustive search over a possible input space of size $2^{8\cdot 7} \cdot 2^{8\cdot 8} = 2^{120}$, which is not feasible, while the second strategy allows the attacker to search for the inputs to the three hash function calls independently. As a consequence, the attacker may tabulate the results of computing SHA-256 on the concatenation of any possible three ASCII character string, and a three random byte sequence at a cost of $2^{21}2^{24} = 2^{45}$ computations, retrieving the input to the first two SHA-256 calls. Finally, he will search for the correct value of the string composed by two ASCII characters concatenated with two random bytes with an effort of $2^{14}2^{16} = 2^{30}$. The total search effort for Strategy B is thus $2^{45} + 2^{30} < 2^{46}$ which is well within feasibility, thus the best strategy is Strategy A.

Question 3 [6 pts]

The *Pohlig-Hellman Exponentiation Cipher* is a symmetric key cipher.
Its definition assumes that a prime p is publicly known, and that a plaintext message m and its corresponding ciphertext c belong to $\mathbb{Z}_p$.
The secret key of the cipher is $k=(k_{\texttt{enc}}, k_{\texttt{dec}})$, where $k_{\texttt{enc}}=e\in\mathbb{Z}^*_{\varphi(p)}$ is employed for encryption, while $k_{\texttt{dec}}=d\in\mathbb{Z}^*_{\varphi(p)}$ is employed for decryption, with $e\cdot d\equiv_{\varphi(p)}1$.
A message $m\in\mathbb{Z}_p$ is encrypted as: $c\leftarrow \textsc{Enc}_{k_{\texttt{enc}}}(m)=m^e \bmod p$, while the corresponding ciphertext is decrypted as: $m\leftarrow \textsc{Dec}_{k_{\texttt{dec}}}(c)=c^d \bmod p$.

(a) Explain how to recover the secret key given one plaintext-ciphertext pair.

(b) Prove the correctness of the cipher for every value of p and m, and state the criteria you should employ to choose the value of p.

(c) Consider the choice $p=10223\cdot 2^{31172165}+1$. Is the corresponding instance of the Pohlig-Hellman Exponentiation Cipher secure? Justify your answer.

Solution:

(a) Given $m, c \in \mathbb{Z}_p$, if $m = c = 0$, the secret key cannot be recovered; otherwise, we need to solve a dlog in the cyclic group $\mathbb{Z}_p^*$ as $c \equiv_p m^e$. Being $\langle m, c \rangle$ a valid plaintext-ciphertext pair, the dlog always exists. Knowing the factorization of $\varphi(p) = p-1$, the Poligh-Hellman attack could be applied to reduce the computational complexity of the problem.

(b) $\text{DEC}_{k_{\text{dec}}}(\text{ENC}_{k_{\text{enc}}}(m)) = m \; \forall \; m \in \mathbb{Z}_p \Leftrightarrow (m^e)^d \equiv_p m, \; \forall \; m \in \mathbb{Z}_p$
$\text{ENC}_{k_{\text{enc}}}(\text{DEC}_{k_{\text{dec}}}(c)) = c \; \forall \; c \in \mathbb{Z}_p \Leftrightarrow (c^d)^e \equiv_p c, \; \forall \; c \in \mathbb{Z}_p$
Therefore, we can limit ourselves to prove that $(m^e)^d \equiv_p 1 \; \forall \; m \in \mathbb{Z}_p$.
Considering that $(m^e)^d \equiv_p m^{1+t\cdot\varphi(p)} \equiv_p m \cdot \left(m^{\varphi(p)}\right)^t$, for a proper integer value t; it is easy to see that if $m = c = 0$ the previous chain of equivalences hold, moreover being $\gcd(m, p)=1 \; \forall m \neq 0$ the chain of equivalences also hold for every other element of $\mathbb{Z}_p$.

The value of p should be chosen such that the factorization of $p-1$ admits a large prime factor, ideally $p=2p_1+1$ would be a good choice, with p_1 a large prime.

(c) The order of the cyclic group $\mathbb{Z}_p$ is $p-1 = 10223 \cdot 2^{31172165}$, therefore the application of the Poligh-Hellman attack greatly simplify the extraction of dlogs, as its computational complexity is dominated by the time required to solve the DLP in the cyclic subgroups of prime order, i.e.: $(\mathbb{Z}_2, \cdot)$ and $(\mathbb{Z}_{10223}, \cdot)$

Question 4 [6 pts]
Consider the cyclic group $G=(\mathbb{Z}_{19}^*, \cdot)$.

(a) List all subgroups of G, exhibiting at least one generator for each of them.

(b) Compute the following discrete logarithm $x \equiv \log_3^{\mathtt{D}}(-1)$ applying the Pohlig-Hellman method.

Solution:

(a) $n = |G| = 18 = 1 \cdot 2 \cdot 3^2$, divisors of the order are: 1, 2, 3, 6, 9, 18
$g = 2$, is a generator of G as: $2^2 \equiv 4, 2^3 \equiv 8, 2^6 \equiv 7, 2^9 \equiv 18$
$H_0 = \langle g^{\frac{18}{1}} \rangle = \langle 1 \rangle = \{1\}, \; |H_0|=1$
$H_1 = \langle g^{\frac{18}{2}} \rangle = \langle -1 \rangle = \{-1, 1\}, \; |H_1|=2$
$H_2 = \langle g^{\frac{18}{3}} \rangle = \langle 7 \rangle = \{7, 11, 1\}, \; |H_2|=3$, there are $\varphi(3)=2$ generators, i.e.: 7, $7^2 \equiv 11$
$H_3 = \langle g^{\frac{18}{6}} \rangle = \langle 8 \rangle = \langle 12 \rangle = \{8, 7, -1, 11, 12, 1\}, \; |H_3|=6$
$H_4 = \langle g^{\frac{18}{9}} \rangle = \langle 4 \rangle = \{4, 16, 7, 9, 17, 11, 6, 5, 1\}, \; |H_4|=9$, there are $\varphi(9)=6$ generators:
4^1, $4^2 \equiv 16$, $4^4 \equiv 9$, $4^5 \equiv 17$, $4^7 \equiv 6$, $4^8 \equiv 5$
$H_5 = \langle g^{\frac{18}{18}} \rangle = G = \langle 2 \rangle = \langle 2^5 \equiv 13 \rangle = \langle 2^7 \equiv 14 \rangle = \langle 2^{11} \equiv 15 \rangle = \langle 2^{13} \equiv 3 \rangle = \langle 2^{17} \equiv 10 \rangle, \; |H_5| = |G| = 18$

(b) The logarithm exists as its base 3, is a generator of the group G, thus the argument of the logarithm $-1 \equiv 18$ also belongs to G. Indeed, there must exist an integer $0 \leq x < \varphi(19)$, such that $3^x \equiv_{19} -1$.
$n = |\mathbb{Z}_{19}^*| = \varphi(19) = 18 = 2 \cdot 3^2$; let us denote the prime factors as $p_1=2$, $p_2=3$.

$$x \equiv_{\varphi(19)} \log_3^{\mathtt{D}}(-1) \quad \Leftrightarrow \quad \begin{cases} x_1 \equiv x \bmod 2 \\ x_2 \equiv x \bmod 3^2 \end{cases} \quad \Rightarrow$$

$$x \equiv_{18} x_1 \cdot 9 \cdot (9^{-1} \bmod 2) + x_2 \cdot 2 \cdot (2^{-1} \bmod 9) \Rightarrow x \equiv_{18} 9 \cdot x_1 + 10 \cdot x_2$$

Denoting as g=3 the base of the logarithm, β=$-1 \equiv_{19} 18$ the argument of the logarithm, the Pohlig-Hellman algorithm compute $(x_1 = x \bmod p_1^1)$, by considering the 2-radix expansion of $x_1 = x \bmod 2$, i.e.: x_1=l_0, $l_0 \in \{0,1\}$ and proceeding as follows:
$\eta = g^{\frac{n}{p_1}} \equiv_{19} 3^9 \equiv_{19} -1$,
$\gamma_0 = 1$,
$\delta_0 \equiv (\beta\gamma_0^{-1})^{\frac{n}{p_1}} \equiv_{19} (-1)^9 \equiv_{19} -1$ **(...accidentally, this is exactly what we were looking for)**
Knowing that $\delta_0 = (g^x\gamma_0^{-1})^{\frac{n}{p_1}} \equiv g^{x_1\frac{n}{p_1}} \Rightarrow \delta_0 \equiv (g^{\frac{n}{p_1}})^{l_0} \equiv_{19} \eta^{l_0}$, we have that:
$-1 \equiv_{19} (-1)^{l_0}$, therefore: $l_0 \equiv_2 1$, and

$\mathbf{x_1 = 1}$.

To compute the value $(x_2 = x \bmod p_2^2)$, with p_2=3
let the 3-radix expansion of $x_2 = x \bmod 3^2$ be x_2=l_0+l_1·3 and denote as
g=3 the base of the logarithm, and β=$-1 \equiv_{19} 18$ the argument of the logarithm.

The computation of l_0 proceeds as follows:
$\eta = g^{\frac{n}{p_2}} \equiv_{19} 3^6 \equiv_{19} 7$
$\gamma_0 = 1$,
$\delta_0 \equiv (\beta\gamma_0^{-1})^{\frac{n}{p_2}} \equiv_{19} (-1)^6 \equiv_{18} 1$.
Knowing that $\delta_0 = (g^x\gamma_0^{-1})^{\frac{n}{p_2}} \equiv g^{x_2\frac{n}{p_2}} \Rightarrow \delta_0 \equiv (g^{\frac{n}{p_2}})^{l_0} \equiv_{19} \eta^{l_0}$, we have that:
$1 \equiv_{19} (7)^{l_0}$, therefore: $l_0 \equiv_3 18 \equiv_3 0$.

The computation of l_1 proceeds as follows:
$\gamma_1 = \gamma_0 \cdot g^{l_0 p_2^0} = 1 \cdot 3^0 = 1$,
$\delta_1 \equiv (\beta\gamma_1^{-1})^{\frac{n}{p_2^2}} \equiv_{19} (-1)^2 \equiv_{19} 1$.
$\delta_1 = (g^x\gamma_1^{-1})^{\frac{n}{p_2^2}} \equiv (g^{x_2-l_0})^{\frac{n}{p_2^2}} \Rightarrow \delta_1 \equiv_{19} (g^{\frac{n}{p_2}})^{l_1} \equiv_{19} \eta^{l_1}$, we have that:
$1 \equiv_{19} (7)^{l_1}$, therefore: $l_1 \equiv_3 18 \equiv_3 0$.

$\mathbf{x_2 = l_0 + l_1 \cdot 3 = 0}$
Therefore, the solution of the discrete logarithm:
$3^x \equiv_{19} (-1)$ is $\mathbf{x \equiv_{18} 9 \cdot x_1 + 10 \cdot x_2 \equiv_{18} 9}$.

Question 5 [15 pts]

(a) Describe the Pollard's $P-1$ factorization method.

(b) Consider the RSA modulus $n = p \cdot q = 253 = 11 \cdot 23$

- Given the public exponent e=$7 \in \mathbb{Z}^*_{\varphi(n)}$, show the value of the RSA private key, k_{priv}=$(p, q, \varphi(n), d)$ and specify every step of the computation.
- Apply a Square & Multiply strategy to sign the message m=25_{decimal}, employing a radix-4 encoding of the exponent.
- Apply the CRT to sign the message m=25, showing every step of the computation.

(c) Compare the asymptotic computational complexity of performing the RSA signature with the CRT to the one of computing the same signature employing a plain S&M strategy with an exponent encoded in a radix R=2^w.

(d) Explain the Montgomery strategy for performing modular multiplications, highlighting its advantages in implementing RSA or dlog-based primitives.

(e) Assume to work into the Montgomery domain: $(\widetilde{\mathbb{Z}}_p, +, \times)$, p=31, and compute the Montgomery multiplication $\widetilde{C}$=MonPro$(\widetilde{A}, \widetilde{B})$, where $\widetilde{A}$=16_{dec} and $\widetilde{B}$=15_{dec}, assuming a binary encoding of the operands.

Solution:

(a) see lectures...

(b) $n = 253$, $\varphi(n) = 220$, $\varphi(\varphi(n)) = 80$, $e = 7$, $d = 63$.
$s \equiv_n m^d \equiv_{253} 25^{63_{\texttt{dec}}}_{\texttt{dec}} \equiv_{253} 25^{333_4}_{\texttt{dec}} \equiv_{253}$
$\equiv_{253} ((25^3)^4 \cdot 25^3)^4 \cdot 25^3 \equiv_{253} (192^4 \cdot 192)^4 \cdot 192 \equiv_{253} (163 \cdot 192)^4 \cdot 192 \equiv_{253} 78 \cdot 192 \equiv_{253}$
$\equiv_{253} 49$
$s \equiv_n m^d$ with CRT, see lectures ...

(c) assuming $n \approx \varphi(n)$ and denoting as $t = \lceil \log_2 n \rceil$ the number of bits employed to encode the secret exponent and the plain message, and that $\lceil \log_2 p \rceil \approx \lceil \log_2 q \rceil \approx \frac{t}{2}$, the computation of $s = m^d \bmod n$ with the CRT has a computational complexity of $\mathcal{O}(\frac{1}{4} \cdot \frac{3}{2}(t-1)\texttt{mul}_t)$, where $\texttt{mul}_t$ is the cost of performing a single multiplication between two operands encoded with t bits each.
Writing down the pseudo-code of the second exponentiation strategy ... it is possible to observe that the w-ary exponentiation algorithm (as a generalization of the square-&-multiply) requires approximately ... $t + 1$ squarings and $2^w + \lfloor \frac{t}{w} \rfloor$ multiplications between t-bit operands; plus a look up table for storing the values of all possible powers of the base m^{2^1}, m^{2^2}, ..., $m^{2^w - 1}$.
Therefore, considering the costs of squarings and multiplications as equal, the w-ary exponentiation will have a computational cost of approximately $\mathcal{O}\left((t + 1 + \lfloor \frac{t}{w} \rfloor + 2^w)\texttt{mul}_t\right)$ including the execution time needed for computing the LUT and considering the LUT accesses with negligible cost.
...

(d) see lectures ...

(e) ... $\widetilde{C} = 23_{\texttt{decimal}}$