Cryptography and Architectures for Computer Security

Exam Code: 095947 (old 090959), A.Y. 2018–2019, Semester: 2

Prof. G. Pelosi

June 25th, 2019 – extra examination session

To have your grade officially recorded in the academic book with the date of today you need to register to this exam session by 7:30pm

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

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

Time: 2h, 15min. 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 [3 pts]
Consider the following composition of the AES-128-CounTeR mode encryption primitive, with $\texttt{IV}_1, \texttt{IV}_2$ sent in clear before the actual ciphertext:

(a) AES-CTR$_{k_1,\texttt{IV}_1}$(AES-CTR$_{k_2,\texttt{IV}_2}$(ptx))

(b) AES-CTR$_{k_1,\texttt{IV}_1}$(AES-CTR$_{k_1,\texttt{IV}_2}$(ptx))

(c) AES-CTR$_{k_1,\texttt{IV}_1}$(AES-CTR$_{k_1,\texttt{IV}_1}$(ptx))

Discuss the resistance against a bruteforce attack of each of these strategies.

Solution:

(a) This strategy is effectively encrypting twice the message under two different keys, as a result its resistance against a plain exhaustive search is 2^{256} calls to the encryption primitive. A meet in the middle approach lowers the computational effort to 2^{129} calls to the encryption primitive, at the cost of 2^{128} cipher blocks in storage.

(b) Since the key employed in the two encryptions is the same, the approach does not offer any significant advantage against a plain exhaustive search for k_1, as the computational effort is only twice the one of a single encryption. On the other hand, a meet-in-the-middle strategy does not offer an advantage as it would entail the same computational effort of a plain exhaustive search, plus significant requirements on the storage side.

(c) This strategy is effectively broken. In fact, the two keystreams generated by the two counter mode encryptions are the same, thus resulting in the plaintext being transmitted.

Question 2 [3 pts]
When using the one-time pad (Vernam's cipher) with the L-bit key $k = \langle 000\ldots0\rangle$, it follows that $Enc_k(m) = k \oplus m = m$ and the message (also encoded with L bits) is effectively sent in the clear! It has therefore been suggested to improve the one-time pad by only encrypting with a key $k \neq \langle 000\ldots0\rangle$ (i.e., k is uniformly chosen at random from the set of non-zero keys having the same bit-length of the plaintext message).
Is this an improvement? In particular, is it still perfectly secret? Prove your answer.
If your answer is positive, explain why the one-time pad is not described in this way.
If your answer is negative, reconcile this fact with the fact that encrypting with $\langle 000\ldots0\rangle$ doesn't change the plaintext.

Solution:
The modified scheme is not perfectly secret. To see this formally, consider the uniform distribution of plaintexts over the message space $\mathcal{M} = \{0,1\}^L$. For any fixed message $m \in \mathcal{M}$, we have

$$Pr[P = m|C = m] = 0 \neq Pr[P = m]$$

This contradicts the definition of *perfect secrecy*. We conclude that in order to obtain perfect secrecy, it must be possible to encrypt using the key $\langle 000\ldots0\rangle$. This may seem counter-intuitive, since this key does not change the plaintext. However, note that an eavesdropper has no way of knowing if the key is $\langle 000\ldots0\rangle$, so the fact that the ciphertext is the same as the plaintext in this case is really of no help to the adversary.

Question 3 [3 pts]

(a) What is a "password salt" and why it is used to store hashed passwords?

(b) What happens if the same salt is used in multiple hashes or the salt has a fixed and relatively short length?

(c) Consider a hash-based account system. Is it reasonable to use the "usernames" as password salts?

Solution:

(a) *What is a password salt and why it is used to store hashed passwords?*
Lookup tables and rainbow tables only work because each password is hashed the exact same way. If two users have the same password, they'll have the same password hashes. These attacks are prevented via randomizing each hash, so that when the same password is hashed twice, the hashes are not the same.

The hashes are randomized through either appending or prepending a random bit-string, called a salt, to the password before hashing. This makes the same password hash into a completely different string every time. To check if a password is correct, the salt is usually stored in the user account database along with the password hash, or as part of the hash string itself.

The salt does not need to be secret. Just by randomizing the hashes, lookup tables, reverse lookup tables, and rainbow tables become ineffective, because an attacker won't know in advance what the salt will be, so they can't pre-compute a lookup table or rainbow table.

(b) *What happen if the same salt is used in multiple hashes or the salt has a fixed and relatively short length?*

A salt re-used several times is ineffective because if two users have the same password, they'll still have the same hash. An attacker can still use a reverse lookup table attack to run a dictionary attack on every hash at the same time. They just have to apply the salt to each password guess before they hash it. If the salt is hard-coded into a popular product, lookup tables and rainbow tables can be built for that salt, to make it easier to crack hashes generated by the product.

If the salt is too short, an attacker can build a lookup table for every possible salt.

(c) *Consider a hash-based account system. Is it reasonable to use the usernames as password salts?*

Usernames may be unique to a single service, but they are predictable and often reused for accounts on other services. An attacker can build lookup tables for common usernames and use them to crack username-salted hashes.

To make it impossible for an attacker to create a lookup table for every possible salt, the salt must be reasonably long.

A sensible choice for the size of the salt is to pick a random value as big as the hash message block, as the salt would anyway be padded to fit that size before being input in the hash. Typical values for message block sizes of hashing algorithms are in the 512–1024 bits range and are thus more than sufficient to provide randomization.

Question 4 [6 pts]

Consider the finite field $\mathbb{F}_{2^6}$.

(a) Exhibit the number of irreducible and primitive polynomials, justifying your answer.

(b) State if $f(x)=x^6+x+1\in\mathbb{F}_2[x]$ is a primitive polynomial, justifying your answer.

(c) Given $\beta(x)=x^2+1\in\mathbb{F}_{2^6}\cong\mathbb{F}_2[x]/\langle f(x)\rangle$, compute $\beta(x)^{512^{-1}}$.

Solution:

(a) $N_1(2)+2\cdot N_2(2)+3\cdot N_3(2)+6\cdot N_6(2)=64$.
$N_1(2)=2$, $N_2(2)=\frac{2^2-2}{2}=1$, $N_3(2)=\frac{2^3-2}{3}=2$. $N_6(2)=64-2-2-66=9$.

$M_6(2)=\frac{\varphi(|\mathbb{F}_{2^6}^*|)}{6}=\frac{\varphi(3^2\cdot 7)}{6}=\frac{(3^2-3)\cdot(7-1)}{6}=6$.

(b) Assuming $f(x)$ is primitive, its roots are generators of the multiplicative group $\mathbb{F}_{2^6}^*$. If $\alpha\in\mathbb{F}_{2^6}$ is such that $f(\alpha)=0\Leftrightarrow\alpha^6\equiv\alpha+1$, then $\texttt{ord}(\alpha)$ is not a divisor of $n=|\mathbb{F}_{2^6}^*|=63=3^2\cdot 7$.

$$\begin{cases}\alpha^3\not\equiv 1\\ \alpha^7\equiv\alpha^2+\alpha\not\equiv 1\\ \alpha^9\equiv\alpha^7\cdot\alpha^2\equiv\alpha^4+\alpha^3\not\equiv 1\\ \alpha^{21}\equiv(\alpha^6)^2\cdot\alpha^9\equiv(\alpha^2+1)\cdot(\alpha^4+\alpha^3)\equiv\alpha^5+\alpha^4+\alpha^3+\alpha+1\not\equiv 1\end{cases}$$

$\alpha^{63}\equiv(\alpha^6)^8\cdot\alpha^{15}\equiv(\alpha^8+1)\cdot(\alpha^2+1)\cdot\alpha^3\equiv\cdots\equiv 1\Rightarrow f(x)$ is primitive.

(c) $\beta(x)^{512^{-1}\bmod 63}\equiv\beta(x)^{8^{-1}}\equiv\beta(x)^8\equiv(x^2+1)^8\equiv(x^6)^2\cdot x^4+1\equiv\ldots\equiv x^4+x$.

Question 5 [6 pts]
Consider an elliptic curve $\mathbb{E}(\mathbb{F}_{11}): \ y^2 = x^3 + 3x$, with order $n = |\mathbb{E}(\mathbb{F}_{11})| = 12$. Denote as $(G, +)$, the finite cyclic group employed to implement an Elliptic Curve Diffie-Hellman protocol; where $G = \mathbb{E}(\mathbb{F}_{11}) = \langle \mathtt{P} \rangle$, with $\mathtt{P} = (2, 5)$, while the internal composition law is equal to the usual "chord & tangent" rule.

(a) In case one party has a private DH-key equal to $K_{priv,\mathtt{A}} = 3 \in \mathbb{Z}_n \setminus \{0\}$ and the other party has a public DH-key equal to $K_{pub,\mathtt{B}} = (7, 1)$, show the value of elliptic curve point shared by the two parties: $K_{\mathtt{AB}}$

(b) The group order $n = 12$ is a composite integer. Discuss the security of the protocol with respect to this consideration.

Solution:

(a) $K_{priv,\mathtt{A}} = 3 \in \mathbb{Z}_n \setminus \{0\}$ and the other party has a public DH-key equal to $K_{pub,\mathtt{B}} = (7, 1)$, show the value of elliptic curve point shared by the two parties: $K_{\mathtt{AB}} = [K_{priv,\mathtt{A}}]K_{pub,\mathtt{B}} = [2](7, 1) + (7, 1) = \ldots = (1, 9) + (7, 1) = (6, 5)$.
$\lambda \equiv_{11} \frac{3 \cdot 7^2 + 3}{2 \cdot 1} \equiv_{11} 7 \cdot 2^{-1} \equiv_{11} 7 \cdot 6 \equiv_{11} 9$.
$x_{[2](7,1)} \equiv_{11} \lambda^2 - 2 \cdot x_{(7,1)} \equiv_{11} 9^2 - 2 \cdot 7 \equiv_{11} 67 \equiv_{11} 1$.
$y_{[2](7,1)} \equiv_{11} \lambda \cdot \left(x_{(7,1)} - x_{[2](7,1)}\right) - y_{(7,1)} \equiv_{11} 9 \cdot (7 - 1) - 1 \equiv_{11} 53 \equiv_{11} 9$.
$[2](7, 1) = (1, 9)$.

$\lambda \equiv_{11} \frac{9-1}{1-7} \equiv_{11} 8 \cdot -6^{-1} \equiv_{11} 8 \cdot (-2) \equiv_{11} 6$.
$x_{[3](7,1)} \equiv_{11} \lambda^2 - x_{[2](7,1)} - x_{(7,1)} \equiv_{11} 6^2 - 1 - 7 \equiv_{11} 28 \equiv_{11} 6$.
$y_{[3](7,1)} \equiv_{11} \lambda \left(x_{(7,1) - x_{[3](7,1)}}\right) - y_{(7,1)} \equiv_{11} 6 \cdot (7 - 6) - 1 \equiv_{11} 5$.
$[3](7, 1) = [2](7, 1) + (7, 1) = (1, 9) + (7, 1) = (6, 5)$.

(b) see lectures ...

Question 6 [8 pts]
Consider an RSA cryptosystem where the public key is $k_{pub} = (n, e) = (221, 5)$

1. Apply the Pollard's ρ factoring method to factorize the public modulus n. Assume the following "random-walking function": $f(x) = x^2 \bmod n$. Show every passage of the computation.
2. Apply the Miller-Rabin test to the smallest non-trivial factor of n through employing the (randomly chosen) bases $a_1 = 2$, $a_2 = 3$. What is the probability of the factor to be "probably prime"?
3. Exhibit the value of the RSA private key, $k_{priv} = (p, q, \varphi(n), d)$, and specify every passage of the computation.
4. Given the ciphertext $c = 101_{\mathtt{decimal}}$ compute the corresponding plaintext applying the CRT formulation of the decryption function.

Solution:
$\ldots$ $n = 13 \cdot 17$
$\varphi(n) = 192 = 2^6 \cdot 3$
$d = e^{-1} \bmod \varphi(n) = 5^{-1} \bmod 192 = \ldots = 77$
$m = ... = 186 \bmod 221$.

Question 7 [6 pts]

(a) 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.

(b) Assume to work into the Montgomery domain: $(\mathbb{Z}_N, +, \times)$, $N = 21$.
Compute the Montgomery multiplication $C{=}A{\times}B \bmod N$, where $A{=}5_{\text{decimal}} = 101_{\texttt{bin}}$ and $B{=}17_{\text{decimal}} = 10001_{\texttt{bin}}$ are Radix-2 encoded values in the Montgomery domain. Show every step of the procedure, performing all the computations in radix-2.

Solution:

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

(c) Radix $b = 2$, $\lceil \log_2 N \rceil = 5$, $R = b^5 = 32$,
$\gcd(R, N) = RR' - NN' = 1 \Leftrightarrow \gcd(32, 21) = 32(2) - 21(3) = 1$,
$R' \stackrel{\text{def}}{=} R^{-1} \bmod N = 11^{-1} \bmod 21 = 2 = (00010)_{\texttt{bin}}$
$N' \stackrel{\text{def}}{=} -N^{-1} \bmod R = -21^{-1} \bmod 32 = 11^{-1} \bmod 32 = 3 = (00011)_{\texttt{bin}}$
$N'_0 = 1$

$B = \langle B_4 B_3 B_2 B_1 B_0 \rangle_b = \langle 10001 \rangle_{\texttt{bin}}$
$A = \langle A_4 A_3 A_2 A_1 A_0 \rangle_b = \langle 00101 \rangle_{\texttt{bin}}$

...

Validation: $C{=}5_{dec} \times 17_{dec} \stackrel{\text{def}}{=} 5 \cdot 17 \cdot R^{-1} \bmod N{=}5 \cdot 17 \cdot 2 \bmod 21{=}170{\equiv_{21}}2$