NWI-IMC061 – Applied Cryptography
Final Exam, Academic Year 2022 – 2023
Tuesday, June 20, 2023, 8.30 – 11.30

Remarks:

- This exam consists of **6 questions**, printed on **6 pages**, front and back. Each (sub-)question indicates how many points it is worth. You can score a maximum of **100 points**.
- Each question is independent; the questions can be solved in the order of your choice.
- The exam is open book. You are not allowed to use any digital device, except for a calculator.
- Write the answers to **each question** on a new page.
- Write clearly, and **always** explain your approach/answer.
- Put on **each page (including this one!)**: your **name** and your **student number**.
- Hand in **both the exam (this sheet) and your solutions** when you are finished.

---

1. **(18 points; General Symmetric Cryptography)** This question covers various topics of symmetric cryptography.
   (a) **(3pt)** Given that one can use public-key cryptography to do key agreement, encryption, and signing, there is no need for symmetric cryptography anymore: yes or no? Include argumentation.
   (b) **(3pt)** Suppose you want to communicate with a friend of yours. Assume that you agreed on a key K. You prefer that *no outsider can read or manipulate* the sensitive data. For your implementation, you can choose between using OFB on top of AES, GCM on top of AES, or CBC-MAC on top of AES (only one of these three options). Which is your preferred choice? Include argumentation.
   (c) **(3pt)** If $\mathcal{H}$ is a collision resistant cryptographic hash function, then $\mathcal{H}(K\|M) = T$ is a secure MAC function: yes (argue why) or no (describe counterexample)?
   (d) **(3pt)** Using an extendable output function $\mathcal{XOF}$, key derivation functions are much simpler to construct than HKDF and PBKDF2: yes or no (argue why)?
   (e) **(3pt)** Name one advantage and one disadvantage of Wegman-Carter MAC over Protected Hash MAC.
   (f) **(3pt)** The sponge construction can be instantiated with the Keccak-f[400] permutation to obtain a hash function that generically achieves 256-bit collision resistance: yes or no? Include argumentation.

(The exam continues on the next page!)

2. **(20 points; MAC and Authenticated Encryption)** A well-known authenticated encryption mode not discussed in the lectures is CCM, which stands for Counter mode with CBC-MAC. It is defined on top of AES-128 instantiated with secret key K, which we denote as E_K. For $a \in \mathbb{N}$ and $X \in \{0, \ldots, 2^a - 1\}$, $\langle X \rangle_a$ denotes the encoding of X as an a-bit string. CCM now operates as follows. (Hint: make a drawing of the scheme!)

   - On input of a 64-bit nonce $N \in \{0,1\}^{64}$, arbitrary-length associated data $A \in \{0,1\}^*$, and arbitrary-length message $M \in \{0,1\}^*$, it defines a string[1]

     $$B = N \parallel \langle |A| \rangle_{64} \parallel M \parallel \langle |A| \rangle_{64} \parallel A \parallel 0^*$$

     to be of length exactly a multiple of 128 bits. This string is then split into 128-bit blocks $B_1, \ldots, B_\ell$.
   - Plain CBC-MAC is evaluated on these ℓ blocks:

     ```
     $X_0 \leftarrow 0^{128}$
     for $i = 1, \ldots, \ell$ do
         $X_i \leftarrow E_K(X_{i-1} \oplus B_i)$
     end for
     return $X_\ell$
     ```
   - Counter mode is evaluated with nonce N to generate a $(128 + |M|)$-bit keystream S:

     ```
     $S \leftarrow$ empty string
     for $i = 0, \ldots, \lceil |M|/128 \rceil$ do
         $S \leftarrow S \parallel E_K(N \| \langle i \rangle_{64})$
     end for
     $S \leftarrow \text{left}_{128+|M|}(S)$
     return $S$
     ```
   - This keystream S is added to $X_\ell \| M$ to obtain 128-bit tag T and $|M|$-bit ciphertext C: $T \| C \leftarrow S \oplus X_\ell \| M$.

   (a) **(3pt)** What is the size of the key K?

   (b) **(4pt)** Is this mode an Encrypt-and-MAC, Encrypt-then-MAC, or MAC-then-Encrypt type construction? Explain your answer.

   (c) **(6pt)** Assume you only have access to the encryption functionality of CCM, so not to its decryption functionality. Each new encryption is for a unique nonce. Describe an attack that breaks the security of CCM in approximately 2^{64} 128-bit message blocks. Describe the security model, the attack, and an informal description as to why your attack succeeds.

   (d) **(4pt)** CCM requires the nonce to be unique for each new evaluation. Suppose an implementer decides to generate nonces at random. What are the implications for the security of CCM? Explain your answer.

   (e) **(3pt)** The plain CBC-MAC construction is known to be vulnerable to the length extension attack. Explain why the length extension attack is not a problem for the evaluation of CBC-MAC within CCM.

(The exam continues on the next page!)

[1]Here, it is implicitly assumed that A and M are of size at most $2^{64} - 1$ bits.

3. **(12 points; Symmetric Cryptography in Leaky Settings)** Let p be a 512-bit permutation. Denote the sponge hash function construction operating on permutation p and with parameters $(b, c, r) = (512, 256, 256)$ as $\mathsf{Sponge}(M, t)$, where M is the input and t the required digest length.

   (a) **(3pt)** What level of collision, preimage, and second preimage resistance does this construction offer generically, assuming that we impose $t \geq 128$?

   In the fifth lecture of the symmetric cryptography part, we touched upon leaky environments. We consider the outer keyed sponge construction in a leaky setting. Let $k = t = 128$. On input of a k-bit key K and arbitrary-length plaintext P, it computes a t-bit tag as $\mathsf{OKS}(K, M) = \mathsf{Sponge}(K \| M, t)$. Assume that the attacker has magically applied a certain trick to the implementation such that for each evaluation $p : X \mapsto Y$ it learns some information. In detail, denoting the bits of X as $X = x_0 x_1 \cdots x_{511}$ (here, the leftmost bit is the outer-most bit of the state, or in other words, absorption and squeezing is done over bits $0, 1, \ldots, 255$ of the state), it learns bit x_i of X, where

$$i = x_{128} + 2x_{129} + \cdots + 2^8 x_{136} \in \{0, \ldots, 511\} .$$

   (b) **(5pt)** Describe an attack on OKS that recovers the key after 128 evaluations of the construction.

   (c) **(4pt)** Consider the exact same construction but now with a $(k = 256)$-bit key. Explain why the attack in (b) does not work anymore, not even in 256 evaluations. It is still possible to somehow recover the key. What is the approximate number of evaluations you need to recover the key: is it of the order 2^{10}, 2^{128}, or 2^{256}? Explain your answer. (Note, this question is *independent* of question (a).)

(The exam continues on the next page!)

4. **(17 points; Parity Oracle Attack on RSA)** We start with some warm-up questions.

   (a) **(1pt)** In lecture 4 and 5 as well as in assignment 3 we treated vulnerabilities of textbook RSA and of variations of RSA that used padding schemes. What is the main property of textbook RSA that was used in these attacks?

   (b) **(2pt)** Describe in your own words what an active oracle attack is.

   (c) **(2pt)** What kind of oracle attacks have most often been the core of attacks against SSL/TLS?

   In this question we will treat an attack on textbook RSA that is possible when a specific oracle exists, that we will call a *parity oracle.*

   Let $N = pq$ be a 2048-bit RSA modulus that we will use in a textbook RSA encryption scheme, where p and q are appropriately chosen safe primes of size around 1024 bits. Furthermore, let e be the public exponent and E denote the encryption function, defined as $E(M) = M^e \pmod{N}$.

   We define a *parity oracle* $\mathcal{O}$ that given an encryption of a message $C = E(M)$, where $0 \leqslant M < N$, returns the parity of M. In other words: $\mathcal{O}(E(M)) = M \pmod{2}$.[2]

   Suppose you want to use the oracle to find out an unknown message M given the public key (e, N) and the ciphertext $C = E(M)$.

   (d) **(1pt)** If a is an integer such that $0 \leqslant a < N/2$, what is the parity of $2a \pmod{N}$? Also, if $N/2 < a < N$, what is the parity of $2a \pmod{N}$?

   (e) **(2pt)** If you query the oracle with input $E(2) \cdot E(M) \pmod{N}$, what will you get as an answer, the parity of which number? What can you learn from this answer about the subdomain that M belongs to?

   (f) **(2pt)** Consider a generalization: split the domain into four intervals (subdomains) and consider the parity of $4a \pmod{N}$. What can you say about the parity of $4a \pmod{N}$ for each of the cases: $0 \leqslant a < N/4$, $N/4 < a < N/2$, $N/2 < a < 3/4N$, $3N/4 < a < N$?

   (g) **(3pt)** What can you learn about the subdomain that M belongs to, if you query the oracle with input $E(4) \cdot E(M) \pmod{N}$? More generally, hat can you learn about the subdomain that M belongs to, if you query the oracle with input $E(2^i) \cdot E(M) \pmod{N}$ for some integer $i \leqslant \log_2 N$?

   (h) **(4pt)** The answers to questions (a)-(g) can be used to learn the entire message M. Present a detailed write-up of the interaction with the oracle and what you learn with each subsequent query. How many oracle queries do you need in case of a 2048-bit modulus N?

(The exam continues on the next page!)

[2]Note that the existence of such an oracle is not so strange: for example a, web shop might always require orders of even numbers and a server-side validation might return errors if one enters an odd number in the order field. This is basically an implementation of the oracle. Nowadays, we implement this more smartly, but one could think of a more involved similar oracle that is harder to exploit, but whose presence is also harder to detect.

5. **(15 points; Variants of Schnorr's Identification Scheme and Digital Signatures)**

Recall Schnorr's Zero-Knowledge protocol discussed in lecture 6, and its more efficient single round variant, Schnorr's identification protocol.

Let p be a prime, take q a divisor of $p-1$ and take $g_1, g_2 \in \mathbb{Z}_p^*$ of order q. Suppose the prover $\mathcal{P}$ gets as input a secret $w \in \mathbb{Z}_q$ and computes the public key $(h_1, h_2) = (g_1^w \pmod p, g_2^w \pmod p)$. The bit length of q is $\log_2 q = 128$ and of p is $\log_2 p = 256$. Consider the following protocol:

$$
\begin{array}{lcl}
\mathcal{P}(w, (h_1, h_2)) & & \mathcal{V}((h_1, h_2)) \\
\hline
r \xleftarrow{\$} \mathbb{Z}_q, & & \\
a_1 \leftarrow g_1^r \pmod p,\ a_2 \leftarrow g_2^r \pmod p & & \\
\mathsf{com} = (a_1, a_2) & \xrightarrow{\mathsf{com}} & \\
 & & \mathsf{ch} \xleftarrow{\$} \mathbb{Z}_{2^t} \text{ with } 2^t < q \\
 & \xleftarrow{\mathsf{ch}} & \\
\mathsf{resp} \leftarrow z = r + \mathsf{ch} \cdot w \pmod q & & \\
 & \xrightarrow{\mathsf{resp}} & \\
 & & g_1^z \stackrel{?}{=} a_1 h_1^{\mathsf{ch}} \pmod p \\
 & & g_2^z \stackrel{?}{=} a_2 h_2^{\mathsf{ch}} \pmod p
\end{array}
$$

(a) **(3pt)** Show formally that the protocol is complete.

(b) **(3pt)** Show formally that the protocol is special sound under the computational discrete logarithm assumption.

(c) **(2pt)** How many times should the protocol be repeated in order to achieve a $1/2^{128}$ soundness error?

(d) **(3pt)** Show that the protocol is honest verifier zero-knowledge.

(e) **(4pt)** Turn the protocol into a digital signature scheme. Write explicitly the signing and verification algorithms.

(The exam continues on the next page!)

6. **(18 points; Hash-Based Signatures)** In lecture 10 we talked about hash-based signatures, specifically about Lamport's one time signature (OTS) and its use in the Merkle Signature Scheme. Recall that Lamport's OTS is very inefficient. Here we will consider an optimized version, that we will call *optimized Lamport OTS*.

Let $H : \{0,1\}^n \to \{0,1\}^n$ be a one-way function. We construct a signature scheme that signs s-bit messages $m \in \{0,1\}^s$.

- **KeyGen:** Let $\ell = s + \lfloor \log_2 s \rfloor + 1$ where $\lfloor a \rfloor$ denotes the closest integer to a that is smaller than a.
For every $i \in \{1, \ldots, \ell\}$, choose a random binary string x_i of length n, and compute $y_i = H(x_i)$. The private key is $sk = (x_1, \ldots, x_\ell)$ and the public key is $pk = (y_1, \ldots, y_\ell)$.
- **Sign:** To sign a message $m \in \{0,1\}^s$, first compute a checksum $c = \mathit{binary}(\sum_{i=1}^{s} \neg m_i)$ where $\neg m_i$ denotes the negation of the bit m_i and $\mathit{binary}(a)$ denotes the binary representation of a, and construct the encoded message $M = m \| c$.
For example, for $m = 010$, $c = \mathit{binary}(1+0+1) = \mathit{binary}(2) = 10$ and $M = m\|c = 01010$.
The signature is then formed as:
$$\sigma = (\sigma_1, \ldots, \sigma_\ell),\ \textit{where}\ \sigma_i = \begin{cases} x_i, \text{if } M_i = 1, \\ y_i, \text{if } M_i = 0. \end{cases}$$
- **Verify:** To verify a signature σ of a message m, first compute a checksum $c = \mathit{binary}(\sum_{i=1}^{s} \neg m_i)$ and construct the encoded message $M = m\|c$.
Output 1 if for all $i \in \{1, \ldots, \ell\}$,
$$y_i = \begin{cases} H(\sigma_i), \text{if } M_i = 1, \\ \sigma_i, \text{if } M_i = 0. \end{cases}$$

Suppose now we have the scheme with the following parameters: $s = 16$, $n = 256$.

(a) **(3pt)** Determine the value of ℓ and the sizes of the public key, private key, and signatures in bits.

(b) **(3pt)** Compare the answer to question (a) to the sizes of the public key, private key, and signatures in bits for the Lamport OTS scheme of the lectures for $s = 16$, $n = 256$.

(c) **(4pt)** As you can see from the answer to (b), it is not immediately obvious where do we optimize compared to the Lamport OTS scheme of the lectures. However, these optimizations become clearer if one would use this optimized Lamport OTS in the Merkle signature scheme (MSS). More concretely, when used in MSS, two clear optimizations are possible: reduction in the MSS signature size and reduction in the verification time. Describe in detail these optimizations and argue exactly how much is saved in terms of memory and time.
Hint: Focus on what can be removed from the MSS signature, and which operations do not need to be performed during verification if one uses the optimized Lamport OTS (as opposed to using the Lamport OTS from the lectures).

(d) **(2pt)** Prove that whenever at least one bit in the encoded message M flips from 1 to 0, at least one other bit in M flips from 0 to 1.

(e) **(3pt)** The relevance of encoding the message may not be clear yet. Show that without the encoding, i.e., if $M = m$, the scheme is not one-time EUF-CMA secure. Explain exactly how to perform a forgery.
Hint: Show that given only one signature, the attacker can forge other signatures.

(f) **(3pt)** Show why the forgery attack in (e) is not possible when the message is encoded as $M = m\|c$.