## C&O 487/687: Assignment #5

Due date: December 4, 2023 (Monday)

1. [5 marks] **Reusing $k$ in ECDSA.**

   Suppose that Alice is trying to sign messages using ECDSA. She is not careful when generating the random integer $k$ and accidentally uses the same $k$ to encrypt multiple messages.

   (a) [2 marks] Show how an attacker intercepting her signatures could detect this.

   (b) [3 marks] Explain how an attacker could recover the secret key.

2. [8 marks] **Ring learning with errors.**

   In this question, we will modify the learning-with-errors based public key encryption scheme from class so that it is done over rings of polynomials instead of matrices. We will be working with polynomials in $R_p = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ where $q$ is a prime and $n$ is a positive integer. This means that the coefficients of the polynomials are reduced mod $q$ and any power of $x$ larger than $x^n$ can be removed through the relation $x^n + 1 = 0$; in other words, replace all occurrences of $x^n$ with $-1$. Let $\chi$ be some distribution on $\mathbb{Z}_q$ and $k$ be an integer. A message will be bitstring $m \in \{0,1\}^n$.

   - $\mathsf{KeyGen}()$: Select a secret polynomial $s(x) \in_R \chi[x]/\langle x^n + 1\rangle$. That is, $s(x) \in R_p$ where each coefficient is sampled independently from $\chi$. Select a polynomial $a(x) \in_R R_p$ sampled uniformly at random. Select a polynomial $e(x) \in_R \chi[x]/\langle x^n + 1\rangle$ in the same way as for $s$. Compute $b(x) = a(x)s(x) + e(x)$ (remember: the polynomial operations are done mod$x^n + 1$ mod $q$). Return $pk = (a(x), b(x))$ and $sk = s(x)$.
   - $\mathsf{Enc}(pk = (a(x), b(x), m \in \{0,1\}^m)$: Select $s'(x)$,$e'(x)$, and $e''(x)$ all randomly from $\chi[x]/\langle x^n + 1\rangle$. Compute $b'(x) = s'(x)a(x) + e'(x)$. Convert the message $m$ from a bitstring $m = m_0 m_1 \dots m_{n-1}$ to a polynomial $m(x) = m_0 + m_1 x + \dots + m_{n-1}x^{n-1}$ and set $c(x) = b(x)s'(x) + e''(x) + \lfloor \frac{q}{2} \rfloor m(x)$. Return the ciphertext $ctxt = (b'(x), c(x))$.
   - $\mathsf{Dec}(sk = s(x), ctxt = (b'(x), c(x)))$: ???

   (a) [2 marks] Reduce the polynomial $x^6 + 4x^4 + 3x^3 + x + 1$ modulo $x^4 + 1$. (To do this, replace $x^4$ by $-1$ wherever you can.) Show your work.

   (b) [2 marks] Write the decryption algorithm.

   (c) [2 marks] Let $n = 3$, $q = 113$, and $\chi$ be the uniform distribution on $\{-1, 0, 1\}$. Given the following public key, secret key and ciphertext, decrypt the ciphertext to recover the message.

   $$pk = \left(a(x) = 71 + 33x + 25x^2, b(x) = 78 + 17x + 100x^2\right), \quad sk = s(x) = 1 + x - x^2$$
   $$ctxt = \left(b'(x) = 64 + 50x + 62x^2, \quad c(x) = 104 + 64x + 105x^2\right)$$

   You may use computational tools to help with this. Indicate which tools you use, if any. There is a small Sage notebook available on LEARN that can help carry out these computations.

   (d) [2 marks] For the parameter choices in part (c), show that public key encryption scheme is correct: in other words, show that decryption of an honestly generated ciphertext will always yield the original message.

3. [11 marks] **Public key infrastructure.**

   The hackers from the Really Super Awesome Association of Engineering Students are at it again. This time, some members of their organization, calling themselves the Engineering Committee for Disrupting Student Assessment (ECDSA), have been trying to impersonate TAs and make fake posts on Piazza to trick math students into believing that their final exams have been cancelled.

   So far, I have managed to delete all their fake posts before anyone saw them, but to prevent this from happening in the future, I am considering having all announcements on Piazza be signed

using elliptic curve digital signatures. To distribute the TA's public keys, I would set up a public key infrastructure to issue certificates containing the TA's public key and the certificates would be signed by me (`dstebila`).

In `a5pki.zip` (available on LEARN), you'll find most of the code for this system, as well as some public keys, certificates, and messages. These scripts again use the Python cryptography library that we previously used in assignments 3 and 4. (Follow the instructions from assignment 3 in order to get Python up and running for this question or use the UW Jupyter server.)

This ZIP file contains the following files:

- `key_gen.py`: A Python file for generating a user's public key and secret key.
- `dstebila.pk`: The public key of the root certificate authority.
- `generate_certificate.py` A Python file for generating a user's certificate by using the CA's signing key to sign the user's public key and other data fields.
- `*.cert`: The certificates of the TAs, containing their public keys.
- `sign.py`: A Python file for generating signatures on messages.
- `message*.signed.txt`: Some alleged messages from the TAs with information about the final exam.
- `verify.py` or `verify.ipnyb`: The skeleton for a Python file for verifying signed messages, with some bits missing for you to fill in. (These two files contain the same contents, and you can use either of them, depending on whether you want to use Python or Jupyter.) This will be the only file you need to edit.

Your task is to finish writing `verify.py` or `verify.ipynb` and determine which messages are legitimate.

(a) [4 marks] Describe (in words or in pseudocode) the things you need to check in order to see if you should trust the signature on a message.

(b) [2 marks] Finish writing `verify.py` (if you are working on your own computer) or `verify.ipynb` (if you are working on UW's Jupyter server at `https://jupyter.math.uwaterloo.ca`).

If you are working on your own computer, make sure all the files are in the same directory and then run `python3 verify.py`

If you are working on UW's Jupyter server `https://jupyter.math.uwaterloo.ca`, upload all the files from the ZIP file into the same directory. Make sure you're using the "Python 3 extras" kernel for the Jupyter notebook.

Submit the source code for your script as a PDF or screenshot to Crowdmark.

(c) [5 marks] Which of the 5 signed messages in the ZIP file can you verify as legitimate? For the messages that are not legitimate, what is the reason?

4. [4 marks] **Man-in-the-middle attack on Signal.**

Throughout this question, assume we are working over a group $G$ of prime order with generator $g$. Let $\mathsf{KDF}$ be a key derivation function.

The Really Super Awesome Association of Engineering Students (RSA-AES) is still causing trouble! Through trial and error, they have discovered that an effective way to drive Prof. Stebila to insanity is to attach jingle bells to his belongings. Their theory is that, if they do drive him crazy, he will not be able to write a fair exam for CO 487, so the class will all fail, and the engineering students will get the better co-op jobs. Up until now, the TAs for the course have been working tirelessly to prevent the jinglebelling, but they've heard rumours that another jingle bell attack is in the works! The leaders of RSA-AES, Eve and Mallory, have been using the Signal messaging app to communicate, since Signal is end-to-end encrypted, and they want to prevent the cryptographically-savvy CO 487 course staff and students from figuring out their plans.

Recall that a secure channel on Signal between two parties Alice and Bob is set up using Triple Diffie–Hellman (X3DH). Alice and Bob have multiple keys:

- Alice and Bob each have long-term identity key pairs, consisting of secret identity keys, $IKA_{sk}$ and $IKB_{sk}$, and corresponding public keys $IKA_{pk} = g^{IKA_{sk}}$ and $IKB_{pk} = g^{IKB_{sk}}$.
- Alice and Bob each have "pre-key" key pairs, consisting of secret prekey keys $SPKA_{pk}$ and $SPKB_{sk}$, and corresponding public prekey keys $SPKA_{pk} = g^{SPKA_{sk}}$ and $SPKB_{pk} = g^{SPKB_{sk}}$, for corresponding secret keys.
- Alice and Bob each sign their public prekey key with their secret identity key, producing signatures $\sigma A$ and $\sigma B$.

Each party publishes all of the above public keys, plus their signature, to a server.

When Alice wants to establish a shared secret key with Bob, she does the following (slightly simplified compared to the real Signal protocol):

(1) Alice obtains $IKB_{pk}$, $SPKB_{pk}$, and $\sigma B$ from the server.

(2) Alice verifies that $\sigma B$ is a valid signature for $SPKB_{pk}$.

(3) Alice generates an ephemeral (i.e. short term) secret key, $EKA_{sk}$ and computes the corresponding ephemeral public key $EKA_{pk} = g^{EKA_{sk}}$.

(4) Alice computes the shared secret

$$K = \mathsf{KDF}(SPKB_{pk}^{IKA_{sk}}, IKB_{pk}^{EKA_{sk}}, SPKB_{pk}^{EKA_{sk}}).$$

(5) Alice sends a chat message to Bob which includes $IKA_{pk}$, $EKA_{pk}$, $SKB_{pk}$, and a ciphertext $c$ formed by encrypting the first chat message that she wants to send to Bob, using a symmetric key encryption scheme and key $K$.

(6) Upon receiving Alice's message, Bob computes the shared secret

$$K = \mathsf{KDF}(IKA_{pk}^{SPKB_{sk}}, EKA_{pk}^{IKB_{sk}}, EKA_{pk}^{SPKB_{sk}}),$$

and decrypts $c$ using $K$.

(7) Before continuing to communicate, Alice and Bob meet up in person and scan each other's QR code. The QR code is computed as

$$\mathsf{qrcode} = H(IKA_{pk} \| IKB_{pk})$$

using a cryptographic hash function $H$. If they don't have the same QR code when they scan each other's QR codes, then they abort.

Since Eve and Mallory didn't take CO 487, they didn't remember to do step 7 before beginning to communicate their evil jinglebelling plans. Describe how Prof. Stebila's loyal TAs could exploit this mistake and thwart their plans.

5. [6 marks] **One-time passwords**

In 2020, the University of Waterloo made two-factor authentication mandatory, to increase security against weak/reused passwords, password database breaches, and phishing attacks. UW's two-factor system uses a commercial product called Duo, which supports one of three authentication methods: (i) a push notification to a proprietary mobile app, (ii) a telephone call or SMS message, or (iii) a one-time passcode. The third option, one-time passcodes, is based on a standardized protocol called HOTP (HMAC-Based One-Time Password, `https://tools.ietf.org/html/rfc4226`, see also Wikipedia).

HOTP works as follows to generate a 6-digit one-time password.

When a user initially registers with a server, the server assigns the user a random secret 128-bit key $k$, which the server stores in its database, and which the user embeds in their HOTP authenticator application. The user and the server initialize a counter $c$ to 0.

Each time the user wants to log in to a server, they press the "generate OTP button" in their application which does the following operations:

(1) Compute $x \leftarrow \text{HMAC-SHA256}(k, c)$.

(2) Let $i$ be the last 4 bits of $x$, represented as a non-negative integer.

(3) Let $y$ be bits $i$ through $i+30$ inclusive of $x$ (i.e., $y$ is 31 bits long), represented as a non-negative integer.

(4) Compute $z \leftarrow y \bmod 10^6$.

(5) Return $z$ as the one-time password.

(6) Increment counter $c$ and save the new counter value.

When the server receives a login request, it looks up the $k$ and $c$ for the user in question and does the same operations as above, then compares what it has computed against what the user sent. If the values match, the server grants the user access, and the server increments the counter $c$ and saves the new counter value. If the values do not match, the server rejects the login request and does not update the counter $c$. Many servers also impose rate-limiting on login requests, for example, locking the account after more than 10 failed login attempts.

(a) [2 marks] Suppose an attacker has compromised your password, and observed several past OTPs you have used, but does not have your OTP key $k$. Describe the best attack (with feasible runtime) you can think of to remotely break into the account. What is the probability of success of your attack? Assume that HMAC-SHA256 is a secure pseudorandom function.

(b) [2 marks] Suppose that the counter in HOTP was not updated by either party, so that each time the user wants to log in to a server it computes $x \leftarrow \text{HMAC-SHA256}(k, 0)$. Under what attack conditions does this change make it easier for an attacker to gain access to your account? Would UW's system be vulnerable if this was used?

(c) [2 marks] An alternative to HOTP is TOTP (Time-based One-Time Password, `https://tools.ietf.org/html/rfc6238`, see also Wikipedia). Briefly investigate TOTP. What is the main difference between TOTP and HOTP? Do you think one is more secure than the other? Is one easier to use than the other?

6. [7 marks] **Let's steal a Tesla.**[1]

One feature that Tesla cars are known for is their passive keyless entry and start (PKES) system, which automatically unlock and start a user's car when the user (in possession of the paired key fob) is in close physical proximity. The key fob and car share a 2-byte car identifier $id$ and a 40-bit secret key, $k$.

The system employs a (proprietary and obsolete) cipher called DST40 as a pseudorandom function. In particular, the DST40 pseudorandom function takes as input a 40-bit key and a 40-bit challenge, and produces a 24-bit response. For the purposes of this question, you can assume that DST40 produces output indistinguishable from random, other than the fact that it has very small inputs and outputs.

The PKES protocol for a Tesla Model S is as follows:

Step 1. The car periodically broadcasts (over radio frequency) its 2-byte car identifier $id$. The broadcast is low powered, so only entities in a given (and small) range will be able to hear these broadcasts.

[1] Please do not steal cars.

Step 2. The key fob is always listening for broadcasts of car identifiers. If it hears the identifier $id$ of the car it is paired with, it will send the car a response, letting it know that it is in range and requesting a challenge.

Step 3. The car will broadcast a 40-bit challenge, $ch$.

Step 4. Upon receiving $ch$, the key fob will send back a response, which it computes using the DST40 cipher and the shared secret key. In particular, the key fob will send back the 24-bit response $rsp = \mathsf{DST40}(k, ch)$.

Step 5. Upon receiving $rsp$, the car will verify that $rsp = \mathsf{DST40}(k, ch)$. If so, it will unlock and start.

(a) First, let's look at the DST40 cipher.

(i) [1 marks] For a given challenge $ch$ and a response $rsp$, how many on average keys map $ch$ to $rsp$? Assume that there is a uniform distribution of responses over the possible challenges and keys.

(ii) [2 marks] How many challenge/response pairs do you need to observe to determine the key (with high probability)?

(iii) [2 marks] Describe a key-recovery attack on the DST40 pseudorandom function that can be executed quickly. You may assume that you have the number of challenge/response pairs computed in the previous part of the question, and that you get to pick what the challenges are in each pair. You may assume that you have access to a precomputed list $L = [\mathsf{DST40}(k_i, ch), k_i, ch]$ for a fixed, public challenge $ch$, and all keys $k_i$, which is sorted by the response given. You may also assume that it takes about 2 seconds to compute $2^{16}$ DST40 operations on your computer.

(b) [2 marks] Describe how you can create your own key fob clone in order to steal a Tesla Model S. You may assume you have the list $L$ from part (a), and that you have the opportunity to be in close proximity to the car and to the key fob whenever needed.

---

## Academic integrity rules

You should make an effort to solve all the problems on your own. You are also welcome to collaborate on assignments with other students in this course. However, solutions must be written up by yourself. If you do collaborate, please acknowledge your collaborators in the write-up for each problem. *If you obtain a solution with help from a book, paper, a website, or any other source, please acknowledge your source. You are not permitted to solicit help from other online bulletin boards, chat groups, newsgroups, or solutions from previous offerings of the course.*

---

## Due date

The assignment is due via Crowdmark by 8:59:59pm on December 4, 2023. Late assignments will not be accepted.