Applied Cryptography

Public Key Cryptography, Assignment 3, Wednesday April 13, 2022

Remarks:

- Hand in your answers through Brightspace.
- Hand in format: PDF. Either hand-written and scanned in PDF, or typeset and converted to PDF. Please, **do not** submit photos, Word files, LaTeX source files, or similar.
- Assure that the name of **each** group member is **in** the document (not just in the file name).

Deadline: Wednesday, May 11, 23.59

Goals: After completing these exercises you should have understanding in the security of public key encryption, digital signatures and Σ-protocols

1. **(10 points)** Given is the following encryption cryptosystem:

KeyGen:

(a) Choose a random prime p

(b) Compute a random multiplicative generator g of $\mathbb{Z}_p^*$

(c) Choose a random $x \xleftarrow{\$} \mathbb{Z}_{p-1}$

(d) Compute $y \leftarrow g^x \pmod{p}$

(e) Output public key $\mathsf{pk} = y$ and private key $\mathsf{sk} = x$ and public parameters (p, g)

Encrypt: To encrypt a message $M < p$

(a) Choose a random $k \xleftarrow{\$} \mathbb{Z}_{p-1}$ and

(b) Compute ciphertext pair (C_1, C_2) as

$$C_1 \leftarrow g^k \pmod{p}$$
$$C_2 \leftarrow y^k M \pmod{p}$$

Decrypt: Decrypt ciphertext as $M \leftarrow C_2 \cdot C_1^{-x} \pmod{p}$

(a) Show that C_2 is uniformly distributed over $\mathbb{Z}_p^*$

(b) Show that the cryptosystem is OW-CPA secure if the computational Diffie-Hellman (CDH) problem is hard

(c) In practice, in order to improve the efficiency, instead of using a generator of $\mathbb{Z}_p^*$, one can use a g of much smaller order than p. Show that in such a case, if a message M is not in the subgroup generated by g, the scheme is vulnerable to a Meet-in-the-middle attack (see the Meet-in-the-middle attack in the lecture slides).

(d) Suppose Malice has eavesdropped (C_1, C_2) sent to Alice in a previous communication. Suppose further that Malice can use Alice as a decryption oracle, such that Alice will decrypt any message sent to her that she has not seen before. Show that by sending an appropriate ciphertext to Alice for decryption, Malice can learn the plaintext of (C_1, C_2).

2. **(5 points)** Consider the digital signature related to the scheme from the previous exercise.

> **KeyGen**: Same as in Exercise 1
> **Sign**: To sign a message M
>
> (a) Find a pair (r, s) such that $g^M = y^r \cdot r^s \pmod{p}$
>
> (b) Output (r, s) as the signature of M
>
> **Verify**: In order to verify a signature (r, s) of M, check that $r \in \langle g \rangle$, $s < r$ and the validity of the equation $g^M = y^r \cdot r^s \pmod{p}$

   (a) Write down the procedure of finding (r, s), i.e. the exact steps of how it can be generated

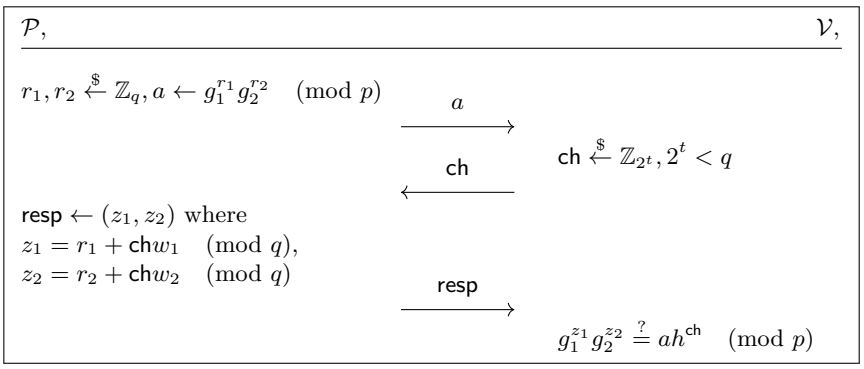
   (b) Show that there exist an existential forgery attack on the cryptosystem

3. **(5 points)** Consider the followig modification of Pedersen bit commitment scheme

$$\mathsf{Comm}(\mathsf{pk}, B, R) = g^R \cdot h^{B+R} \text{ where } R \xleftarrow{\$} \mathbb{Z}_n \text{ and } h \in \langle g \rangle, |\langle g \rangle| = n$$

   where h is such that $\log_g h$ is unknown to the parties. Investigate the properties ofthe commitment (binding, hiding) and the flavor (perfect, statistical, computational).

4. **(10 points)** Let p-prime, $q|p-1$, $g_1, g_2 \in \mathbb{Z}_p^*$ of order q. Suppose the prover $\mathcal{P}$ gets as input $w_1, w_2 \in \mathbb{Z}_q$ and $h = g_1^{w_1} g_2^{w_2} \pmod{p}$. Consider the following protocol



   Prove that this is a Σ-protocol for the relation $\{(x, (w_1, w_2)) | x = (p, q, g_1, g_2, h), h = g_1^{w_1} g_2^{w_2}\}$

5. **(10 points)** Recall from Introduction to Cryptography the basics of elliptic curve cryptography: we have an elliptic curve

$$E : y^2 = x^3 + ax + b$$

   over some finite field $\mathbb{F}_p$, where $a, b \in \mathbb{F}_p$ are *curve parameters*. We denote with $E(\mathbb{F}_p)$ the points $P = (x, y)$ on the curve (so $y^2 = x^3 + ax + b$ holds) with $x, y \in \mathbb{F}_p$. This $E(\mathbb{F}_p)$ forms a group, so we can add $P_1, P_2 \in E(\mathbb{F}_p)$ together: $P_3 = P_1 + P_2 \in E(\mathbb{F}_p)$. There is also the *point at infinity*, denoted by $\mathcal{O}$, that acts a bit like 0: for any point P we get $P + \mathcal{O} = P$. Adding a point to itself a number of times is denoted by $[n]$, so $[3]G = G + G + G$.

   Then, we pick a certain point $G \in E(\mathbb{F}_p)$, called the *generator*, that has a large prime order; the order of a point was the first n such that

$$[n]G = \underbrace{G + G + \ldots + G}_{n \text{ times}} = \mathcal{O}.$$

   If the order n for G is very large and prime, we can do Diffie-Hellman key exchange with this setup:

- Alice picks a secret key $a \in [1, \ldots, n]$ and computes her public key: the point $P = [a]G$.
- Bob picks a secret key $b \in [1, \ldots, n]$ and computes his public key: the point $Q = [b]G$.
- Alice computes the shared key by multiplying Bob's point by her secret key: $R = [a]Q$,
- Bob computes the shared key by multiplying Alice's point by his secret key: $R' = [b]P$.

As a warm-up before we go to ECDSA:

(a) Show that the shared secrets are the same: $R = R'$.

(b) Complete the following toy-example using sage: the parameters will be $p = 17$, $a = 2$, $b = 2$. Making the finite field $\mathbb{F}_p$ and the curve E in sage can be done using

```
F = FiniteField(p)
E = EllipticCurve(F, [a,b])
```

i. Find out the cardinality of the group $E(\mathbb{F}_p)$ using the right command in sage.

ii. Compute the order of the point $G = (5, 1)$ using the right command in sage, explain why this is a good generator. In sage, creating a point on E can be done by `G = E(5, 1)`.

iii. Let Alice's secret key $a = 6$. What's her public key?

iv. Bob's public key is $Q = (7, 6)$. What's the shared secret?

v. What is Bob's secret key?

(c) A real-life example is similar but has much larger parameters. We take the well-known and often-used `Curve25519` over a field with prime $p = 2^{255} - 19$. Build the curve E using `E = EllipticCurve(F, [0,486662, 0, 1, 0])` and let the generator G be the point

```
Gx = 9
Gy = 43114425171068552920764898935933967039370386198203806730763910166200978582548
G = E(Gx, Gy)
```

i. Compute the cardinality n of $E(\mathbb{F}_p)$ and the order q of G. What is n/q and what does this value imply?

ii. Your secret key a will be
`28111612080006492741872676474874404260743807513041351699201661077095088937 27`
What is your public key?

iii. Bob's public key Q is
`(43943787516356481247689314833207090678477943925840872948547454252824441637727, 21356658827884665961183205449166535999349292496264273491054739595546217830 3)`
What is the shared secret?

6. **(10 points)** In this exercise we will perform ECDSA on `Curve25519` from exercise 5. ECDSA was discussed in lecture 8 on the last slide. It can be given using the following diagram:

KeyGen:

(a) Let G be a base point of $E(\mathbb{F}_p)$ of order q, where p is an odd prime or a power of 2.

(b) Choose a random $d \in \mathbb{Z}_q^*$ and compute $Q = dG$

(c) Output public key $\mathsf{pk} = Q$ and private key $\mathsf{sk} = d$

Sign: Given message M,

(a) $k \xleftarrow{\$} \mathbb{Z}_q, (x_1, y_1) \leftarrow kG$, and $r \leftarrow x_1 \pmod q$

(b) Set $h = H(M)$ and calculate $s \leftarrow (h + dr)k^{-1}$

(c) Set $\sigma = (r, s)$ and output message - signature pair (M, σ)

Verify: To verify the message - signature pair (M, σ)

(a) Parse $\sigma = (r, s)$, verify $0 < r, s < q$ and calculate $h = H(M)$

(b) Compute $u_1 = hs^{-1} \pmod q$ and $u_2 = rs^{-1} \pmod q$

(c) Compute $X = u_1 G + u_2 Q$

(d) If X is point of infinity, output Fail

(e) Take $X = (x_1, y_1)$, check $r \stackrel{?}{=} x_1 \pmod q$ and output Accept if check succeeds, otherwise Fail

(a) Show the correctness of ECDSA, i.e., that indeed $r = x_1 \mod q$ for valid signatures?

(b) Given the prime p, the curve E, the generator G and the order q of `Curve25519` (from exercise 5). We also use the same a as secret key for Alice, and the public key P as computed before. Assume we have some message M such that $h = H(M) = 5$. Create a signature (r, s) for $h = 5$ (i.e. compute s). We use h directly instead of a message M so that we avoid using a hash in this exercise. The following code in sage might help:

```
Fq = FiniteField(q)
h = 5
k = Fq.random_element()
K = int(k)*G //do you see why int() is necessary?
r = mod(K[0], q)
```

(c) Verify that the following pair (r, s) is a valid signature for h for the public key Q_b:

```
r = mod(193703920910737566124082452472035973752675889329152773817950202746572185897 3, q)
s = mod(237871765932909663400314091483268182570408279895507215238702918363472600614 1, q)
Qb_x = 532327828701224171975911730973085687978404961071535861994274005179240746097 69
Qb_y = 378168934365781821613162126862386083306981674847628153582332449607145168215 01
Qb = E(Qb_x, Qb_y)
h = mod(659207561052421550123056874400183606603413890117084054606099726355180250829 7, q)
```

When building the PlayStation 3, Sony decided to use ECDSA with their private key to sign software. This separates legitimate software from illegitimate. Sony decided not to take k randomly in the signature generation, but instead to use a static (secret) k for all signatures.

(d) Assume you have two signatures (r_1, s_1) and (r_2, s_2) where k is the same. What do you know about the relation between s_1 and s_2?

(e) Does Sony's decision to take k static impact the security of the signature?

Given the same signature (r, s) as before, the following point $Q_{\text{oh no}}$ will also verify for the same value h:

```
T_x = 325606250916557431795983626356110631294008115727848805560023387167927233504
T_y = 320263035917129627512413075478829813592782127179102366977063165037649225003 07
T = E(T_x, T_y)
Qohno = Qb + T
```

(f) Show that (r, s) also verifies for the point $Q_{\text{oh no}}$.

(g) What is special about the point T here? Why does $Q_{\text{oh no}}$ also work as a public key?

In the Monero cryptocurrency, ECDSA on `Curve25519` is used to create signatures and to sign transactions. In short, users spend their coins by creating a valid signature, and it prevents double spending of a coin by checking if the associated public key has already been used to spend that coin.

(h) Explain how an attacker can spend his coins multiple times using the above special point $Q_{\text{oh no}}$. (This attack was only prevented in May 2017, and so it was possible perform this attack for more than three years)

(i) **(bonus)** How do we prevent this attack?

(j) **(bonus)** How much money can you make by breaking and misusing bad crypto in cryptocurrencies (for example, using the breaks on Monero, ZCash and ABCMint)?