Applied Cryptography

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

Exercises with answers and grading.

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).

Begin Secret Info: .....

(a) **(2.5 points)** Since $x \xleftarrow{\$} \mathbb{Z}_{p-1}$, $y = g^x \pmod{p}$ is a generator as well, so y^k is uniformly distributed in $\mathbb{Z}_p^*$ when k is uniformly distributed in $\mathbb{Z}_{p-1}$. Since multiplication by by an element of $\mathbb{Z}_p^*$ is simply a permutation, we conclude that C_2 is uniformly distributed over $\mathbb{Z}_p^*$.

(b) **(2.5 points)** Let there be an adversary $\mathcal{A}$ that breaks the OW-CPA security of the scheme. We show that we can construct a adversary $\mathcal{B}$ that breaks CDH.
Suppose $\mathcal{B}$ is given a CDH instance (p, g, g_a, g_b) with $g_a, g_b \in \mathbb{Z}_p^*$ and $g_a = g^a, g_b = g^b$ for some unknown a, b. The goal of $\mathcal{B}$ is to find g^{ab} with non-negligible advantage.
Upon receiving the instance, $\mathcal{B}$ prepares as follows:

- Sets public key $\mathsf{pk} = g_a$, and ciphertext $(C_1, C_2) = (g_b, C_2)$ for some random C_2.
- Gives pk and (C_1, C_2) to $\mathcal{A}$

After some time, $\mathcal{A}$ outputs a message M, which with non-negligible advantage satisfies $M = C_2 \cdot C_1^{-a}$, i.e. $M = C_2 \cdot g^{-ab}$. From here $\mathcal{B}$ can calculate $g^{ab} = C_2 \cdot M^{-1}$.

(c) **(2.5 points)** Suppose r is the order of the group generated by g. The for every element a in this group it holds that $a^r = 1$. Then for a valid ciphertext (C_1, C_2) we have $C_2^r = y^{kr} M^r = g^{xkr} M^r = M^r$, i.e. we managed to remove the random mask that hides the message in the ciphertext. Now note that

$$M_1^r \cdot M_2^r = (M_1 M_2)^r$$

i.e. this transformed ciphertext satisfies multiplicativity, so we can use Joux et al.'s attack as in the lecture slides on RSA. (The students must show the exact procedure of the attack).

(d) **(2.5 points)** Suppose Malice has intercepted (C_1, C_2). Then Malice picks $r \xleftarrow{\$} \mathbb{Z}_p^*$, computes $C_2' = C_2 \cdot r$ and sends (C_1, C_2) to Alice. The decrypted message from Alice is $Mr \pmod p$. Since Malice knows r, it is easy to find M.

End Secret Info .....

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 $r \in \langle g \rangle$, $s < p$ 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 exists an existential forgery attack on the cryptosystem

Begin Secret Info: .....

(a) **(2.5 points)** First note that $g^M = y^r \cdot r^s = g^{xr} \cdot r^s \pmod p$, so picking a random $k \xleftarrow{\$} \mathbb{Z}_{p-1}$, and setting $r = g^k \pmod p$ we obtain $g^M = g^{xr} \cdot g^{ks} \pmod p$, i.e. $M = xr + ks \pmod{p-1}$. The last is an equation in s which can easily be solved.

(b) **(2.5 points)** It can immediately be argued that the scheme is existentially forgeable since the message is not hashed. But here are two different forgeries:

- $(r, s) = (g^e y \pmod p, -r \pmod{p-1})$ is a valid signature of $M = es \pmod{p-1}$ for $e \xleftarrow{\$} \mathbb{Z}_{p-1}$
- $(r, s) = (g^e y^v \pmod p, -rv^{-1} \pmod{p-1})$ is a valid signature of $M = es \pmod{p-1}$ for $e, v \xleftarrow{\$} \mathbb{Z}_{p-1}$, and $v \neq 0$

End Secret Info .....

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 of the commitment (binding, hiding) and the flavor (perfect, statistical, computational).
Begin Secret Info:.....

(a) **(2.5 points)** Computational binding: Suppose $\mathsf{Comm}(\mathsf{pk}, B, R) = \mathsf{Comm}(\mathsf{pk}, 1-B, R')$. This means $g^R \cdot h^{B+R} = g^{R'} \cdot h^{1-B+R'}$ and from here $log_g h = (R-R')/(1-2B+R'-R)$, i.e. DL can be solved

(b) **(2.5 points)** Perfect hiding: $g^R \cdot h^{B+R}$ is statistically independent of the value of B

Note that actually this is exactly the Pedersen commitment if you take $\mathsf{Comm}(\mathsf{pk}, B, R) = (gh)^R \cdot h^B$.
End Secret Info.....

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

$\mathcal{P}$,		$\mathcal{V}$,
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_q, a \leftarrow g_1^{r_1} g_2^{r_2} \pmod p$	$\xrightarrow{a}$	
		$\mathsf{ch} \xleftarrow{\$} \mathbb{Z}_{2^t}, 2^t < q$
	$\xleftarrow{\mathsf{ch}}$	
$\mathsf{resp} \leftarrow (z_1, z_2)$ where $z_1 = r_1 + \mathsf{ch} w_1 \pmod q$, $z_2 = r_2 + \mathsf{ch} w_2 \pmod q$	$\xrightarrow{\mathsf{resp}}$	
		$g_1^{z_1} g_2^{z_2} \stackrel{?}{=} a h^{\mathsf{ch}} \pmod p$

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

Begin Secret Info:.....
This is just a very simple generalization of Schnorr's protocol. Namely,

- **(2 points) Completeness/Correctness**: If the prover knows the witnesses $w_1, w_2 \in \mathbb{Z}_q$, and performs the protocol honestly, the verifier accepts since:

$$g_1^{z_1} g_2^{z_2} = g_1^{r_1 + \mathsf{ch} w_1} g_2^{r_2 + \mathsf{ch} w_2} = a \cdot g_1^{\mathsf{ch} w_1} g_2^{\mathsf{ch} w_2} = a h^{\mathsf{ch}} \pmod p$$

- **(4 points) Special soundness**: Given two accepting transcripts for the same commitment $\mathsf{trans} = (\mathsf{com}, \mathsf{ch}, (\mathsf{resp}_1, \mathsf{resp}_2)) = (a, \mathsf{ch}, (r_1 + \mathsf{ch} w_1, r_2 + \mathsf{ch} w_2))$, and $\mathsf{trans}' = (\mathsf{com}, \mathsf{ch}', (\mathsf{resp}_1', \mathsf{resp}_2')) = (a, \mathsf{ch}', (r_1 + \mathsf{ch}' w_1, r_2 + \mathsf{ch}' w_2))$, we have

$$w_1 = \frac{\mathsf{resp}_1 - \mathsf{resp}_1'}{(\mathsf{ch} - \mathsf{ch}')}, \; w_2 = \frac{\mathsf{resp}_2 - \mathsf{resp}_2'}{(\mathsf{ch} - \mathsf{ch}')} \quad \Rightarrow \text{ the witness can be extracted with probability 1.}$$

- **(4 points) Special HVZK**: For given $\mathsf{ch} \in \mathbb{Z}_q$
  - Choose $(\mathsf{resp}_1, \mathsf{resp}_2) \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q$ and calculate $a \leftarrow g_1^{\mathsf{resp}_1} g_2^{\mathsf{resp}_2} h^{-\mathsf{ch}}$
  - The distributions of the real transcripts and the simulated transcripts are the same – in both a given valid transcript occurs with prob. $1/q^2$ (in one first r_1, r_2 is chosen a random, in the other first $\mathsf{resp}_1, \mathsf{resp}_2$ is chosen at random.)

End Secret Info .....

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
`(439437875163564812476893148332070906784779439258408729485474542528244416377 27,`
`213566588278846659611832054491665359993492924962642734910547395955462178303)`
What is the shared secret?

Begin Secret Info: .....

(a) **(2 points)** Just $R = [a]Q = [ab]G = [ba]G = [b]P = R'$.

(b) **(4 points)**

i. First set up the right curve: `p = 17; a = 2; b = 2; F = FiniteField(p); E = EllipticCurve(F, [a,b])` Then, the size of $E(\mathbb{F}_p)$ is just `E.cardinality()`: 19

ii. Define the point G on E by `G = E(5,1);` We compute the order of G by `G.order()`: 19. So G generates every point on $E(\mathbb{F}_p)$, and as 19 is prime, G is a good generator for ECDH.

iii. `6*G` gives the point $(16, 13)$

iv. First define Q by `Q = E(7, 6);` Then `6*Q` gives the point $(10, 11)$.

v. This is simplest to do by brute-force, $Q = [9]G$.

(c) **(4 points)**

i. `n = E.cardinality(); q = G.order();`
`n = 57896044618658097711785492504343953926856930875039260848015607506283634007912`
`q = 72370055773322622139731865630429942408571163593799076060019509382854542509 89`
`n/q = 8`
This implies that the group generated by G is large, but not all of $E(\mathbb{F}_p)$. There is a cofactor $h = 8$, and we need to make sure we are working in the group generated by G, denoted $\langle G \rangle$, because it has a large prime order (check that q is prime). We should be careful not to use and of the points in $E(\mathbb{F}_p) \setminus \langle G \rangle$, we want to work in a group of prime order!

ii. `a*G =`
`(370484147435197330251932637838311092126393044511008041040949775077803257718 37,`
`794834830337363353107434051241084017433212320278651644581218764599622394 2543)`

iii. `a*Q =`
`(126940578630294099105188551275625377034579047944268895640086047772203784659 81,`
`135544562430831272322804504171207674604446455529519622387817443285870770807 33)`

End Secret Info .....

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 = 32560625091655743179598362635611063129400811572784880556002338716792723350 4
T_y = 32026303591712962751241307547882981359278212717910236697706316503764922500307
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)?

Begin Secret Info:.....

(a) **(1.5 point)** We need to show that x_1 of X is indeed equal to $r(\mod q)$. By writing out we get

$$\begin{aligned} u_1G + u_2Q &= (u_1 + u_2d)G \\ &= (ms^{-1} + rds^{-1})G \\ &= (m + rd)s^{-1}G \quad \text{where } s = k^{-1}(m + rd) \\ &= \frac{(m + rd)}{(m + rd)} \cdot kG \\ &= K. \end{aligned}$$

We know that r is the x-coordinate of K, and so it equals the x coordinate of $u_1G+u_2Q(\mod q)$.

(b) **(1 point)** This depends on the random k you sample to compute the signature, but essentially all that is left to do is to compute s from the values we have: `s = k**-1*(m + r*d)`

(c) **(1.5 point)** This code is simply the verification part:

```
u1 = m*s**-1
u2 = r*s**-1
T = int(u1)*G + int(u2)*Q
r == mod(T[0], q)  //prints true
```

(d) **(1.5 point)** When k is static, we get $s_1 - s_2 = k^{-1}(m_1 - m_2)$. But we know m_1, m_2, s_1 and s_2 so we can compute which k is used every time. Hence, given some s and a static k, we can compute the secret key d by $s = k^{-1}(m + rd)$.

(e) **(1 point)** It completely breaks the security and allows for a recovery of the private key, essentially the worst that can happen to a cryptosystem.

(f) **(1 point)** Simply repeat what we did in (c).

(g) **(1.5 point)** This special point T has order 8, and so it is on $E(\mathbb{F}_p)$, but not in the group generated by G! It's exactly one of thesre points in $E(\mathbb{F}_p) \setminus \langle G \rangle$ that we should be worried about. But then also $Q_{\text{oh no}}$ is on $E(\mathbb{F}_p)$ but not in the group generated by G. However, when computing K, we multiple $Q_{\text{oh no}}$ by u_2 which happens to a multiple of 8. We get

$$u_2 Q_{\text{oh no}} = u_2(Q_b + T) = u_2 Q_b + \mathcal{O} = u_2 Q_b$$

and so the rest of the computation will give a valid signature.

(h) **(1 point)** An attacker can now create multiple valid signatures for the same message m for multiple public keys. This is essentially double spending in such a cryptocurrency.

(i) **(+ 1 point)** Instead of blindly assuming a public key Q is in $\langle G \rangle$, we check that a public key actually isn't in $E(\mathbb{F}_p) \setminus \langle G \rangle$. We can check this by the order of a point Q, if the order is q, it is in $\langle G \rangle$. So we verify that $[q]Q = \mathcal{O}$.

(j) **(+ 1 point)** Monero had a market cap of more than 300 million euros when this bug was still feasible, so stealing a million or so using double-spending attacks was doable. Equally so for ZCash and ABCMint. Not many careers give you the option to earn 3 millions in such a short amount of time, hence cryptography is a valid career path. (All of this is modulo market economics.)

End Secret Info .....