IT486 Blockchains and Cryptocurrencies

HOMEWORK 1

September 23, 2020

Jhanvi Chauhan
Student ID: 201701445

Problem 1. *Blinding with RSA*

Part A:

Assume:

- $n = 77, m_1 = 10, m_2 = 8, e = 7$.
- Alice has public key (n, e). Bank has private key (n, d).

Consider the following situation: Alice sends message m_1 to the bank to sign with necessary blinding using r_1. If challenged, she can reveal r_1. The bank returns $s = (m_1 r_1^7)^d \pmod{n}$. Alice can now compute $s^e r_2^{-1} \pmod{n}$ to obtain m_2 if she can find such an r_2 so that $r_1^e m_1 = r_2^e m_2 \pmod{n}$.

Now, we must only prove that there exists an r_1 and r_2 pair for $n = 77, m_1 = 10, m_2 = 8, e = 7$ such that $r_1^e m_1 = r_2^e m_2 \pmod{n}$.

we have to show that,

$8r_1^7 \pmod{77} \cong 10r_2^7 \pmod{77}$
$r_1^7 \pmod{77} \cong 8^{-1} 10 r_2^7 \pmod{77}$
$r_1^7 \pmod{77} \cong [8^{-1} \pmod{77}]*[10 \pmod{77}]*[r_2^7 \pmod{77}]$
$r_1^7 \pmod{77} \cong [29 * 10 \pmod{77}]*[r_2^7 \pmod{77}]$ $(\because 8 * 29 \cong 1 \pmod{77})$
$r_1^7 \pmod{77} \cong 59 r_2^7 \pmod{77}$
$r_1^7 \pmod{77} \cong (31 r_2)^7 \pmod{77}$ $(\because 59 \cong 31^7 \pmod{77})$
$r_1 \pmod{77} \cong 31 r_2 \pmod{77}$

So if Alice takes values of (r_1, r_2) such that $r_1 = 31 r_2$, then she can cheat the system.

Part B

While it was possible for us to find out values of r_1 and r_2 such that $r_1^e m_1 = r_2^e m_2 \pmod{n}$, in practice, the numbers e, n, m and d are large numbers of lengths around 256 bits and this is not an easy problem to solve for such large numbers. To solve this problem, we need to find the e^{th} root of $m_1 m_2^{-1}$ or $m_2 m_1^{-1}$ and it is known that modular root extraction is hard.

The fastest general method currently available for computing e^{th} modular roots under the conditions on n and e above is to factor n and use the fact that modular root extraction – the reverse of modular exponentiation – is easy **only** when given the prime factors to determine d (because we know that $x^{ed} = x \pmod{n}$ and Alice does not know d). It's possible that there may be methods that compute modular roots without factoring n or determining d. But so far no general methods have been found for doing so that are faster than factoring n.[1]

[1] http://mathaware.org/mam/06/Kaliski.pdf

Hence, mathematically speaking, since the problem of finding values of r_1 and r_2 such that $r_1^e m_1 = r_2^e m_2 \pmod{n}$ is **equivalent to the problem of finding modular roots**, which, in itself, is a hard problem to solve, Alice cannot realistically use this cheating method.

Problem 2. *Breaking ECDSA:*

Part A

- The x-coordinate of $kG = r$.
- $s = \frac{(z+re)}{k}$
- e is the private key.
- Hence, if x-coordinate of kG is 0, we have: $s = \frac{z}{k}$. Here, the adversary knows z already because it is the hash of what we are signing. And we found out a k that satisfies. Since $r = 0$, we are not dependent on private key e to get a signature and hence the adversary can easily forge one.
- In the verification step, we need $u = \frac{z}{s}$ and $v = \frac{r}{s} = 0$. We also know from above the $k = \frac{z}{s}$. Hence $u = k$.
- $uG + vP = uG = kG = (r, y)$.

Part B

- We know that $s = \frac{(z+re)}{k}$. So if $s = 0$, we can find e as $e = \frac{-z}{r}$.
  - We know z already because it is the hash of what we're signing.
  - We also have r from the signature (r, s).
- Hence, an adversary can easily find out the **private key** e.

Problem 3. *Multi-judge escrow:*

4-out-of-9 multisig

- Key division:
  - Alice: 3 keys
  - Bob: 3 keys
  - One key to each judge
- Payment:

- Alice transfers money to the common 4-out-of-9 multisig account.
- In the event of no cheating by Alice or Bob, a new transaction can be created that transfers the money from the 4-out-of-9 multisig account to Bob.
- We do not need the judges, since Alice and Bob combined have 6 keys in total.

- Cheating Prevention:
  - If Bob cheats and doesn't hold his end of the bargain, Alice can take one key from one of the judges randomly. Now that she has 4 keys, she will transfer back the money to herself.
  - If Alice cheats, Bob can can take one key from one of the judges randomly. Now that he has 4 keys, he will transfer back the money to himself.
  - The judges cannot cheat, even if they collaborate, because they only have three keys in total.

Problem 4. *Bitcoin scripts:*

Part A

Unlocking Script: OP_2 followed by OP_3

Operation	**Reaction**
	Stack: 2, 3
OP_2DUP	Stack: 2, 3, 2, 3
OP_ADD	Stack: 2, 3, 5
OP_5	Stack: 2, 3, 5, 5
OP_EQUALVERIFY	Stack: 2, 3
OP_SWAP	Stack: 3, 2
OP_DROP	Stack: 3
OP_3	Stack: 3, 3
OP_EQUAL	True

Hence, valid.

Part B

Let us assume the input is x_2 followed by x_1. (Top of stack is the right-most element.)

here, the top element is 1 and the second top element is $2(x_1+x_2)$. An even and odd number can never be equal hence the transaction is marked invalid.

Operation	Reaction
	Stack: x_2, x_1
OP_2DUP	Stack: x_2, x_1, x_2, x_1
OP_2DUP	Stack: x_2, x_1, x_2, x_1, x_2, x_1
OP_ADD	Stack: x_2, x_1, x_2, x_1, x_2+x_1
OP_ADD	Stack: x_2, x_1, x_2, $x_1 + x_2 + x_1$
OP_ADD	Stack: x_2, x_1, $x_2 + x_1 + x_2 + x_1$
OP_1	Stack: x_2, x_1, $x_2 + x_1 + x_2 + x_1$, 1
OP_EQUALVERIFY	False and mark invalid

Part C

Unlocking Script: OP_0 followed by OP_0

Operation	Reaction
	Stack: 0,0
OP_1	Stack: 0, 0, 1
OP_0	Stack: 0, 0, 1, 0
OP_EQUAL	$\rightarrow$ False—Stack: 0, 0
OP_IF	$\rightarrow$ not true
OP_3DUP	$\rightarrow$ skipped
OP_SUB	$\rightarrow$ skipped
OP_ADD	$\rightarrow$ skipped
OP_3	$\rightarrow$ skipped
OP_EQUAL	$\rightarrow$ skipped
OP_ELSE	$\rightarrow$ excuted
OP_DUP	Stack: 0, 0, 0
OP_SUB	Stack: 0, 0
OP_EQUAL	True
OP_ENDIF	

Hence, valid.
