CSCI971 Advance Computer Security: Homework #2

Mei Wangzhihui 2019124044

Problem 1

Solution

We define the outputs as O_0, O_1
for 0^{64}, there is

$$R_0 = 0^{32},\ L_0 = 0^{32}$$

$$L_1 = R_0 = 0^{32},\ R_1 = F(k_1, R_0) \oplus L_0 = F(k_1, R_0)$$

$$L_2 = R_1 = F(k_1, R_0),\ R_2 = F(k_2, R_1) \oplus L_1 = F(k_2, F(k_1, 0^{32}))$$

Similarly, for $1^{32}0^{32}$, there is

$$L_2 = \bar{F}(k_1, 0^{32}),\ R_2 = F(k_2, \bar{F}(k_1, 0^{32}))$$

thus we can define, $m_0 = F(k_1, 0^{32})$, $c_0 = F(k_2, m_0)$, $m_1 = \bar{F}(k_1, 0^{32}) = \bar{m_0}$, $c_1 = F(k_2, m_1)$
if two outputs are from PRP, then the left 32 bits of $O_1 \oplus O_2$ is 1^{32}
we can easily find that 2) is from PRP, and the other 3 is from random permutation.
the program code are here:

```
def xor_left32bit(a, b):
    return not (~((a >> 32) | (b >> 32)) & 0xffffffff)
if __name__ == "__main__":
    output = [(0x9d1a4f78cb28d863,0x75e5e3ea773ec3e6), (0xe86d2de2e1387ae9,0x1792d21db645c008),
              (0x2d1cfa42c0b1d266, 0xeea6e3ddb2146dd0), (0x4af532671351e2e1, 0x87a40cfa8dd39154)]

    for idx, outputpair in enumerate(output, start=1):
        if xor_left32bit(outputpair[0],outputpair[1]):
            print(idx)
```

Problem 2

Solution

We can draw the whole process of the protocol

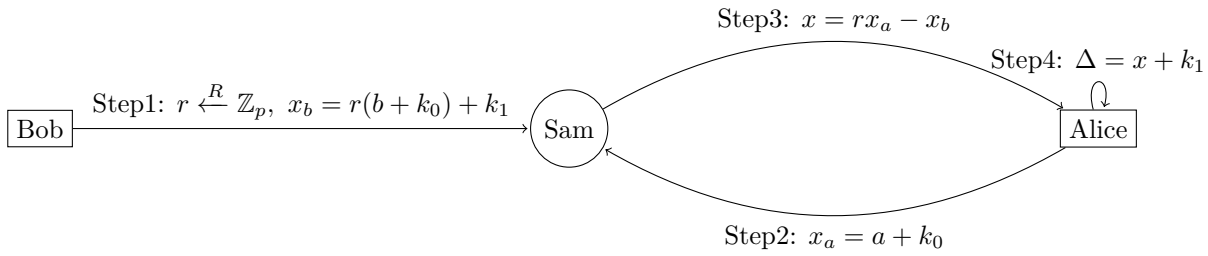

Figure 1: The protocol procedure

As $\Delta = x + k_1 = r(a - b)$ so we get the condition that $r \neq 0$
if (k_0, k_1) are used for more than once. We assume it was tested if $a = b$ and $a' = b'$
For Sam:

$$\begin{cases} x_a = a + k_0 \\ x_b = r(b + k_0) + k_1 \end{cases}$$

$$\begin{cases} x'_a = a' + k_0 \\ x'_b = r'(b' + k_0) + k_1 \end{cases}$$

Problem 2 continued on next page...

So Sam learned that $a' - a = x'_a - x_a$.
For Alice:

$$\begin{cases} x = r(a-b) - k_1 \\ x' = r'(a'-b') - k_1 \end{cases}$$

Alice learned ration of $(a-b)/(a'-b')$ which reveal b/b'

Bugfix
the core of the issue was the reusing of the (k_0, k_1), so we should generate new indenpendent key from it by PRF. We define a secure PRF F defined at $\{K, \mathbb{Z}_b, \mathbb{Z}_p^2\}$ we can derive the key-pair (k_p, k_q) from F and initial seed key k as: $F(k, n) = (k_p, k_q)$ and n is value of counter.
We should let Alice and Bob get the value of counter synchronically for the syncronicity of key-pair. each time Alice and Bob finish the comparison, they increase the their counter by 1. so they get the syncronical key-pair.

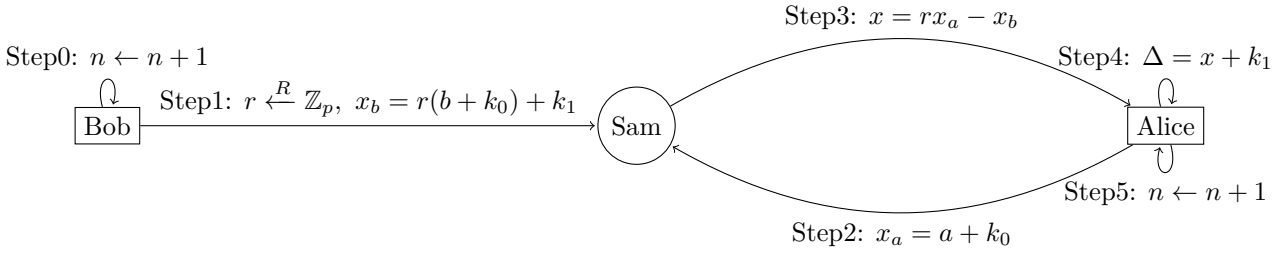

Figure 2: The fixed protocol procedure