Crypto - HW 2

Hagai Ben Yehuda, ID num: 305237000
Jonathan Bauch, ID num: 204761233

1

1.a

Let A be an efficient algorithm such that given $h = g^x$, A outputs x with probability ϵ.
Define:

$$A'(g^x, g^y) = (g^x)^{A(y)}$$

Note that if $A(g^y) = y$ then

$$A'(g^x, g^y) = (g^x)^{A(y)} = (g^x)^y = g^{xy}$$

Thus

$$\Pr(A' \text{ is correct}) = \Pr(A'(g^x, g^y) = g^{xy})$$
$$= \Pr((g^x)^{A(y)} = g^{xy}) = \Pr(A(g^y) = y) = \epsilon$$

Since A is efficient, A' is an efficient algorithm with the required property.

1.b

Let A be an efficient algorithm as described in the question. Construct A' as follows:

$$A'(g^x, g^y, g^z) = \begin{cases} 1 & \text{if } g^z = A(g^x, g^y) \\ 0 & \text{else} \end{cases}$$

Then we have:

$$\Pr(A'(g^x, g^y, g^{xy}) = 1 | (x, y) \in \mathbb{Z}_{|G|} \times \mathbb{Z}_{|G|}) = \Pr(A(g^x, g^y) = g^{xy}) = \epsilon$$

On the other hand we have:

$$\Pr(A'(g^x, g^y, g^z) = 1 | (x, y, z) \in \mathbb{Z}_{|G|} \times \mathbb{Z}_{|G|} \times \mathbb{Z}_{|G|}) = \Pr(A(g^x, g^y) = g^z) = \frac{1}{|G|}$$

Where the last equality stems from the fact that for every x, y there is a single value $z \in \mathbb{Z}_{|G|}$ (as g is a generator) such that $A(g^x, g^y) = g^z$ and since z is chosen uniformly from $\mathbb{Z}_{|G|}$ the equality follows.
Note that even if A is not deterministic this equality holds as:

$$\begin{aligned}
&\Pr(A(g^x, g^y) = g^z) = \\
&\sum_{a \in \mathbb{Z}_{|G|}} \Pr(A(g^x, g^y) = g^z | A(g^x, g^y) = g^a) \Pr(A(g^x, g^y) = g^a) = \\
&\sum_{a \in \mathbb{Z}_{|G|}} \Pr(z = a) \Pr(A(g^x, g^y) = g^a) = \\
&\frac{1}{|G|} \sum_{a \in \mathbb{Z}_{|G|}} \Pr(A(g^x, g^y) = g^a) = \frac{1}{|G|}
\end{aligned}$$

Thus we obtain:

$$\begin{aligned}
&|\Pr_{(x,y)\leftarrow \mathbb{Z}_{|G|}\times\mathbb{Z}_{|G|}}(A'(g^x,g^y,g^{xy})=1)\\
&-\Pr_{(x,y,z)\leftarrow \mathbb{Z}_{|G|}\times\mathbb{Z}_{|G|}\times\mathbb{Z}_{|G|}}(A'(g^x,g^y,g^z)=1)| =\\
&=\epsilon-\frac{1}{|G|}
\end{aligned}$$

Thus A' is a distinguisher with the requested properties.

2

Assume such an efficient algorithm A exists, construct:

$$A'(v)=\begin{cases}1 & \text{if } f(A(v))=v\\ 0 & \text{else}\end{cases}$$

Note that A' is also efficient.

Now:

$$\begin{aligned}
\Pr_{v\leftarrow f(U_n)}(A'(v)=1) &= \Pr_{v\leftarrow f(U_n)}(f(A(v))=v)\\
&= \Pr_{v\leftarrow f(U_n)}(A(v)\in f^{-1}(v))\geq\epsilon
\end{aligned}$$

On the other hand:

$$\begin{aligned}
\Pr_{v\leftarrow U_{n+s}}(A'(v)=1) &= \Pr_{v\leftarrow U_{n+s}}(f(A(v))=v)=\\
&= \Pr_{v\leftarrow U_{n+s}}(A(v)\in f^{-1}(v))
\end{aligned}$$

But note that $\Pr(f^{-1}(v)\neq\emptyset)\leq\frac{1}{2^s}$ Since there are 2^{n+s} elements in U_{n+s} and f is a function, $f:\{0,1\}^n\to\{0,1\}^{n+s}$ there are at most 2^n elements in f's image, thus the probability that $v\in U_{n+s}$ has an origin according to f, is at most $\frac{2^n}{2^{n+s}}=\frac{1}{2^s}$ (this bound is of-course achieved if f is one-to-one). thus $\Pr(f^{-1}(v)\neq\emptyset)\leq\frac{1}{2^s}$ and hence : $\Pr_{v\leftarrow U_{n+s}}(A(v)\in f^{-1}(v))\leq\frac{1}{2^s}$.
This yields that:

$$|\Pr_{v\leftarrow f(U_n)}(A'(v)=1)-\Pr_{v\leftarrow U_{n+s}}(A'(v)=1)|\geq\epsilon-\frac{1}{2^s}$$

As requested.

3

Assume $G:\{0,1\}^n\to\{0,1\}^l$ is a PRG and assume for contradiction that ε is not indistinguishable in the presence of an eavesdropper. Then there exists a PPT adversary A such that $\Pr(A \text{ Wins})>\frac{1}{2}+\epsilon$. We shall now construct a PPT adversary D that distinguishes $G(U_n)$ from U_l. On input v D will do the following:

- Execute A and obtain two messages, m_0,m_1.
- Draw $b\leftarrow\{0,1\}$.
- Supply A with $m_b\oplus v$, if A was correct return 1 else return 0.

We now show that D is a distinguisher between the distributions $G(U_n)$ and U_l and thus arrive at a contradiction to the fact that G is a PRG.

$$\Pr_{v \leftarrow G(U_n)}(D(v) = 1) = \Pr_{v \leftarrow G(U_n)}(A(m_b \oplus v) = b) =$$
$$= \Pr(A \text{ Wins}) > \frac{1}{2} + \epsilon$$

On the other hand

$$\Pr_{v \leftarrow U_l}(D(v) = 1) = \Pr_{v \leftarrow U_l}(A(m_b \oplus v) = \bar{b}) = \frac{1}{2}$$

Assume for the sake of contradiction that the last equality is incorrect, then there are two possibilities:

- $\Pr_{v \leftarrow U_l}(A(m_b \oplus v) = \bar{b}) > \frac{1}{2}$: In this case if we define $f(v) = \overline{A(v)}$ then $\Pr_{v \leftarrow U_l}(f(m_b \oplus v) = b) > \frac{1}{2}$ and thus F is a PPT adversary that show that one time pad is not adversarial indistinguishable which is a contradiction to a theorem we proved.
- $\Pr_{v \leftarrow U_l}(A(m_b \oplus v) = \bar{b}) < \frac{1}{2}$ then $\Pr_{v \leftarrow U_l}(A(m_b \oplus v) = b) > \frac{1}{2}$ and thus A is a PPT adversary that show that one time pad is not adversarial indistinguishable.

hence, indeed $\Pr_{v \leftarrow U_l}(D(v) = 1) = \frac{1}{2}$. Thus:

$$|\Pr_{v \leftarrow G(U_n)}(D(v) = 1) - \Pr_{v \leftarrow U_l}(D(v) = 1)| > \epsilon$$

which is a contradiction to the fact that G is a PRG, and thus ε is indeed indistinguishable in the presence of an eavesdropper.

4

We will construct a PPT adversary D that distinguishes between oracle access to a function from F and oracle access to a random function. We define D for an input f:

- Query the value of f for the string 0^n and save that result as A_1
- Query the value of f for the string 10^{n-1} and save that result as A_2
- If A_1 and A_2 are identical except for the first bit (for which $A_1[0] = \overline{A_2[0]}$) return 1, else return 0.

We now prove that D is in-fact a PPT distinguisher.
First note it in-fact runs in polynomial time as the oracles perform in polytime and n bit comparison is also done in polytime. Now note that:

$$\Pr_{f \leftarrow F}(D(f) = 1) = 1$$

This equality holds since if $f \in F$ then $f(0^n) = G(k) \oplus 0^n = G(k)$ for some $k \in \{0,1\}^n$ and $f(10^{n-1}) = G(k) \oplus 10^{n-1} = \overline{G(k)[0]}G(k)[1...n-1]$ for the same k and thus D will return 1 for f. We denote URF the uniform distribution over the functions $f : \{0,1\}^n \to \{0,1\}^n$.

$$\Pr_{f \leftarrow URF}(D(f) = 1) = Pr_{f \leftarrow URF}(f(0^n) = \overline{f(10^{n-1})[0]}f(10^{n-1})[1...n-1]) = \frac{1}{2^n}$$

Where the last equality is true since f was chosen in random and thus the probability of it having a certain value for input x is identical for each value and there are 2^n possible values, hence:

$$|\Pr_{f\leftarrow F}(D(f)=1) - \Pr_{f\leftarrow URF}(D(f)=1)| = 1 - \frac{1}{2^n}$$

And thus F is indeed not a PRF.

5

5.a

This claim is incorrect.
Assume for the purpose of contradiction that such a hard core predicate exists, we denote the hard-core predicate for an OWF $f:\{0,1\}^n \to \{0,1\}^l$ by HC_f Note that HC_f operates on f's input. We now define a new function $g:\{0,1\}^n \to \{0,1\}^{l+1}$ as $g(x) = (f(x), HC_f(x))$. Note that since HC_f only operates on the input and is a generic HCP, $HC_f = HC_g$, thus if g is indeed an OWF HC_f is a hard-core predicate for g, but given $g(x)$, $HC_f(x)$ is fully known (it is simply the last bit of $g(x)$) thus HC_f is not a hard core predicate for g, it now remains to show that g is in-fact a OWF which will contradict the assumption that a generic hard-core predicate exists.
We will show that if g is not a OWF then f is not a OWF in contradiction to how we chose it.

Assume that g is not a OWF, then there exists an algorithm A, that inverts g with probability $\geq \epsilon$. We define D, and show that it inverts f with probability ϵ (which will contradict the fact that f is a OWF). On input $y \in \{0,1\}^n$ we define D as follows:

- Execute A on $y0$, denote the result x. If $f(x) = y$ return y.
- Execute A on $y1$, denote the result x. If $f(x) = y$ return y.
- return 0^n

$$\begin{aligned}\Pr(D \text{ inverts } f(x)) &\geq \Pr(A \text{ inverts } f(x)0) + \Pr(A \text{ inverts } f(x)1) \\ &\geq \Pr(A \text{ inverts } g(x)) \geq \epsilon\end{aligned}$$

Where the second to last inequality is correct since either $f(x)0$ or $f(x)1$ is exactly $g(x)$. Thus we arrive at a contradiction to the fact that f is a OWF.

5.b

This claim is incorrect. Define $f(x) = 0^n$ for all $x \in \{0,1\}^n$ and define a HCP for f: $HC_f(x) =$ x's first bit. We first show that this is indeed a HCP:
Note that for all $x \in \{0,1\}^n$ $f(x)$ assumes the same value, thus if A is a PPT adversary that proves HC_f is not a HCP for f, it must predict the value of x's first bit with no information at all (as $f(x)$ is constant) in other words A predicts a random bit with probability greater then $\frac{1}{2}$, which is a contradiction to the bit being random and thus no such A exists.
Note that since f is a constant function, it can be inverted with probability 1 as an adversary which always returns 0^n is always successful in finding an inverse to $f(x)$ (f is constant). Thus f is not a OWF but it does have a HCP as shown.

6

6.a

To show $QR \leq \mathbb{Z}_p^*$ it is enough to show (i) $1 \in QR$, (ii) closure under multiplication, and (iii) closure under inversion.

i Indeed $1 \in QR$, because $1 \equiv 1^2 \pmod p$

ii Let $s_1, s_2 \in QR$. Then are $r_1, r_2 \in \mathbb{Z}_p^*$ s.t. $s_1 \equiv r_1^2 \pmod p$, $s_2 \equiv r_2^2 \pmod p$. Then follows:

$$s_1 s_2 \equiv r_1^2 r_2^2 = (r_1 r_2)^2 \pmod p$$

and $r_1 r_2 \in \mathbb{Z}_p^*$ because $\mathbb{Z}_p^*$ is a group. Therefore $s_1 s_2 \in QR$ by definition.

iii Let $s \in \mathbb{Z}_p^*$. Then there is $r \in \mathbb{Z}_p^*$ s.t. $s \equiv r^2 \pmod p$. Therefore:

$$s^{-1} \equiv \left(r^2\right)^{-1} = \left(r^{-1}\right)^2 \pmod p$$

and $r^{-1} \in \mathbb{Z}_p^*$, therefore$s^{-1} \in QR$.

6.b

Let $g \in \mathbb{Z}_p^*$, $\mathbb{Z}_p^* = \langle g \rangle$. Assume by contradiction that $g \in QR$. Then there is an $r \in \mathbb{Z}_p^*$ s.t. $g \equiv r^2 \pmod p$. Because g generates $\mathbb{Z}_p^*$ there exists $i \in \mathbb{Z}$ s.t. $r \equiv g^i \pmod p$. Therefore $g \equiv g^{2i} \pmod p \iff g^{2i-1} \equiv 1 \pmod p$. Therefore $2i-1 | o(g) = \varphi(p) = p-1$. Note that for prime $p > 2$ we know p is odd, therefore $p-1$ is even. On the other hand $2i-1$ is odd, therefore we get a contradiction (that odd divides even), meaning $g \notin QR$. For the edge case where $p = 2$, $\mathbb{Z}_p^*$ is the trivial group (of 1 element), and in this case the claim is not true (because $\mathbb{Z}_p^* = QR = \langle 1 \rangle$). From now on we will assume $p > 2$.

6.c

Let $g \in \mathbb{Z}_p^*$, $\mathbb{Z}_p^* = \langle g \rangle$.

i Let $a \in \mathbb{Z}_p^*$. Assume $a \in QR$. Then there exists $r \in \mathbb{Z}_p^*$ s.t. $a \equiv r^2 \pmod p$. Because g generates $\mathbb{Z}_p^*$, there exists a $k \in \mathbb{Z}$ s.t. $r \equiv g^k \pmod p$. Therefore $a \equiv r^2 \equiv \left(g^k\right)^2 = g^{2k} \pmod p$.

ii Let $a \in \mathbb{Z}_p^*$. Assume that $a \equiv g^{2k} \pmod p$ for some k. Then $a \equiv \left(g^k\right)^2 \pmod p$, $g^k \in \mathbb{Z}_p^*$, therefore by definition $a \in QR$.

6.d

Let $a \in \mathbb{Z}_p^*$, $\mathbb{Z}_p^* = \langle g \rangle$.

i Assume $a \in QR$. Then by (c) there is a $k \in \mathbb{Z}$ s.t. $a \equiv g^{2k} \pmod p$. Therefore

$$a^{\frac{p-1}{2}} \equiv \left(g^{2k}\right)^{\frac{p-1}{2}} = \left(g^{p-1}\right)^k \equiv 1^k = 1 \pmod p$$

(Note that $g^{p-1} \equiv 1$ because $o(g) = p - 1$)

ii Assume $a^{\frac{p-1}{2}} \equiv 1 \pmod p$. There is an $i \in \mathbb{Z}$ s.t. $a \equiv g^i \pmod p$. Therefore

$$1 \equiv \left(g^i\right)^{\frac{p-1}{2}} = \left(g^{\frac{p-1}{2}}\right)^i \overset{*}{=} (-1)^i \pmod p$$

Thus i is even (again assuming $p > 2$). Denote $i = 2k$, and now by (c) we get that $a \in QR$.

* - This can be explained as follows: $g^{\frac{p-1}{2}} \not\equiv 1$ because $o(g) = p-1 > \frac{p-1}{2}$, and $\left(g^{\frac{p-1}{2}}\right)^2 = g^{p-1} \equiv 1$. Therefore necessarily $g^{\frac{p-1}{2}} \equiv -1$, because ± 1 are the only square roots of $1 \pmod p$. (There are no other roots because $x^2 - 1$ as at most 2 roots)

6.e

Denote $a = g^x \mod p$. Then x is even $\iff$ there is a $k \in \mathbb{Z}$ s.t. $x = 2k \iff a \in QR$ (by c) $\iff a^{\frac{p-1}{2}} \equiv 1 \pmod p$ (by d).

Therefore given $f(x) = g^x \mod p$ we can compute $b := (f(x))^{\frac{p-1}{2}} \mod p$. If $b = 1$ then necessarily x is even, i.e. $parity(x) = 0$. Otherwise (if $b = 0$) x is odd, i.e. $parity(x) = 1$.

$a^n \mod p$ can be computed efficiently as follows: compute values a^{2^k} iteratively by squaring (mod p), until $2^k \geq n$ (note that we don't have to store a^{2^k} in memory, as we compute (mod p)). Then according to the binary representation of n, multiply these values for which the k'th bit of n is 1. The whole process involves $O(\log_2(n))$ multiplications mod p, which is linear in the number of bits of n.

7

Listing 1: Code

```
MAX_CYCLE = 2 ** 17 - 1

class LFSR:
    def __init__(self, initial_state, feedback_states):
        self.initial_state = initial_state
        self.feedback_states = feedback_states
        self.state = None
        self.reset()

    def next(self):
        r = self.state[0]
        feedback_bit = sum(self.state[x] for x in self.feedback_states) % 2
        for i in range(len(self.state) - 1):
            self.state[i] = self.state[i + 1]
        self.state[len(self.state) - 1] = feedback_bit
        return r

    def take(self, n):
        return (self.next() for _ in range(n))

    def reset(self):
        self.state = self.initial_state[:]


def calc_cycle(lfsr):
    lfsr.reset()
```

```python
    lfsr.next()
    c = 1
    while lfsr.state != lfsr.initial_state:
        lfsr.next()
        c += 1
    return c


def test(lfsr):
    print('first 30 bits: ', list(lfsr.take(30)))
    c = calc_cycle(lfsr)
    print('cycle length = %d' % c)
    print('precentage of max cycle: %d%%' % (float(c) / MAX_CYCLE * 100))


def main():
    print('max cycle length: %d' % MAX_CYCLE)
    initial_state = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    key1 = [0, 2, 3, 5]
    print('Test a')
    test(LFSR(initial_state, key1))
    key2 = [0, 8]
    print('Test b')
    test(LFSR(initial_state, key2))


if __name__ == '__main__':
    main()
```

Listing 2: Execution output

```
max cycle length: 131071
Test a
first 30 bits:  [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
cycle length = 131071
precentage of max cycle: 100%
Test b
first 30 bits:  [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
cycle length = 35805
precentage of max cycle: 27%
```

7.a

As seen from the execution output, LFSR1 has the maximum cycle length of $2^{17} - 1$

7.b

LFSR2 with the given input state has a cycle of 35805 steps, which is only 27% of the maximum $2^{17} - 1$.