Ungraded Homework Solutions

CSC 152 – Cryptography

Please notify me of any errors you find. If you need help, ask.

1) *Write one of the following reductions: from PreimageFinder(H, y) to 2ndPreimageFinder(H, x), or from 2ndPreimageFinder(H, x) to PreimageFinder(H, y). What security implication does your reduction establish?*

Showing Find2ndPreimage(H, x) reduces to FindPreimage(y) is straightforward.

```
Find2ndPreimage(H,x)
    do
        x' = FindPreimage(H,H(x))
    while (x' == x)
    return x'
```

I used a loop in case FindPreimage($H, H(x)$) returns x. This reduction will fail if FindPreimage($H, H(x)$) always returns x which could happen if FindPreimage($H, H(x)$) is deterministic and finds x as its first answer, or if x is the only preimage of y (unlikely).

This establishes that the existence of an efficient FindPreimage implies the existence of an efficient Find2ndPreimage. It's contrapositive tells us if there is no efficient Find2ndPreimage then there is no efficient FindPreimage, or in other words second-preimage resistance implies preimage resistance.

The other direction would look like this.

```
FindPreimage(H,y)
    ...
    x' = Find2ndPreimage(x)
    ...
```

The problem is that FindPreimage is given an element of the range and Find2ndPreimage requires an element of the domain, so we'd have to find an x where $H(x) = y$ in order to ask Find2ndPreimage to do its work.

2) *Recall that you can compute a value that is equivalent to* $x \bmod (2^a - b)$ *as* $(x \operatorname{div} 2^a) \cdot b + (x \bmod 2^a)$. *Use this fact to reduce (base-10)* $123456789 \bmod (2^{12} - 2)$ *to a smaller, equivalent number. If after doing this reduction once, the result is more than 12 bits, do it a second time to reduce it further.*

There are a couple of ways to do this problem.

The mathematical way goes like this. If you divide $123456789/2^{12}$ you get 30140 and remainder 3349. This means $123456789 = 30140 \cdot 2^{12} + 3349$. If you reduce the 2^{12} term mod $2^{12} - 2$ you get $123456789 = 30140 \cdot 2 + 3349 = 63629$. This means 123456789 and 63629 are equivalent mod $2^{12} - 2$. Since this result is not less than 2^{12} (ie, needs more than 12 bits) it needs further reduction. If you divide $63629/2^{12}$ you get 15 and remainder 2189. This means $63629 = 15 \cdot 2^{12} + 2189$. If you reduce the 2^{12} term mod $2^{12} - 2$ you get $63629 = 15 \cdot 2 + 2189 = 2219$. Since this result is less than 2^{12} it needs no further reduction.

If you wanted to write this more concisely, you might format it like this.

$123456789/2^{12} = 30140$ and $123456789 \bmod 2^{12} = 3349$
$123456789 = 2^{12} \times 30140 + 3349 = 2 \times 30140 + 3349 = 63629$

$63629/2^{12} = 15$ and $63629 \bmod 2^{12} = 2189$
$63629 = 2^{12} \times 15 + 2189 = 2 \times 15 + 2189 = 2219$

Optimized on a computer, it goes like this. 123456789 in binary is 111010110111100110100010101. If we split this into the low 12 bits 110100010101 and the remaining high bits 111010110111100, we can combine

them as $(2 \times 111010110111100) + 110100010101 = 1111100010001101$. Since this is more than 12 bits long, we know that it needs further reduction. If we split 1111100010001101 into the low 12 bits 100010001101 and the remaining high bits 1111, we can combine them as $(2 \times 1111) + 100010001101 = 100010101011$. Since this is not more than 12 bits long, we know that it needs no further reduction.

3) Below is Horner's method of polynomial evaluation with four different variations. For each variation write the equivalent polynomial. Write "..." to indicate "the pattern continues until", and use x_i instead of $x[i]$. *Hint: The first one is* $x_0k^n + x_1k^{n-1} + \cdots + x_{n-1}k$.

```
                        A      B           C                    res
res = A
                        0  res + x[i]  res * k     ----------------------------------------
for (i=0; i<n; i++)
                        0  res * k     res + x[i]  ----------------------------------------
    res = B
                        1  res + x[i]  res * k     ----------------------------------------
    res = C
                        1  res * k     res + x[i]  ----------------------------------------
```

Using the same technique shown in class, we slowly expand the polynomial being formed by the loop until we see the pattern. So, for the second one, $(0k + x_0) = x_0$ is the value of *res* after one iteration, $x_0k + x_1$ after two, $(x_0k + x_1)k + x_2 = x_0k^2 + x_1k + x_2$ after three, etc. This is enough to see the pattern and after n iterations *res* will contain the value of $x_0k^{n-1} + x_1k^{n-2} + \cdots + x_{n-2}k^1 + x_{n-1}$. Similar analysis shows that the third polynomial is $k^n + x_0k^n + x_1k^{n-1} + x_2k^{n-2} + \cdots + x_{n-1}k$ and the fourth polynomial is $k^n + x_0k^{n-1} + x_1k^{n-2} + x_2k^{n-3} + \cdots + x_{n-1}$.

4) *Recall that H is ε-almost-universal if the probability $h(a) = h(b)$ is no more than ε when $a \neq b$ and $h \in H$ is chosen randomly. The following H is a family of functions all with domain $\mathbb{Z}_6$ and co-domain $\mathbb{Z}_4$. For what value of ε is H ε-almost-universal? Show your work. H is defined as follows:*

	h1	h2	h3	h4	h5
0	2	3	0	1	3
1	3	2	1	0	0
2	0	1	3	2	1
3	0	0	2	2	3
4	2	1	1	3	2
5	0	3	3	2	0

If the adversary chooses 2 and 5, then when h is chosen randomly, there is a $3/5$ chance that $h(2) = h(5)$ because h_1, h_3, h_4 all cause a collision. No other pair of inputs has a higher probability of collision when h is chosen randomly, so the collection of functions is ε-almost-universal for $\varepsilon = 3/5$. On an exam with a small domain you should list all possible pairs of domain elements, give each probability, and identify the maximum.

For easy grading on a test, you might format it like this.

```
(0,1)  0/5
(0,2)  0/5   (1,2)  0/5
(0,3)  1/5   (1,3)  0/5   (2,3)  2/5
(0,4)  1/5   (1,4)  1/5   (2,4)  1/5   (3,4)  0/5
(0,5)  1/5   (1,5)  1/5   (2,5)  3/5   (3,5)  2/5   (4,5)  0/5

(2,5) has 3/5 chance of collision, and no other pair has higher, so H is (3/5)-almost-universal.
```