Homework 4

Erik Brakke

October 8, 2015

Collaborators: Kyle Hogan.

Answer 1

Proof: Supposed G_3 was not a PRG
This means that we have some distinguisher D that can tell $G_3(s)$ from random
We know that $G_3 = G_1(s_1) \circ G_2(s_2)$
Therefore, $\Pr[D(G_1(s_1) \circ G_2(s_2)) = PRG] > 1/2 + \epsilon$ Where epsilon is non negligible
We also know that $\Pr[D(random) = random] > 1/2 + \epsilon$
We can now give D just the input of G_1 or G_2 concatinatied with random bits
There must be some point between $D(random)$ and $D(G_3(s_3))$ Where D will be able to tell random from PR
Let's split the input of D by G_1 and G_2
So $D(G_1(s_1) \circ random)$ or $D(random \circ G_2(s_2))$
$\Pr[D(G_1(s_1) \circ random) = PRG] = 1/2 + \epsilon/2$ (because half of the bits are random and half are psuedorandom)
This is the same with using G_2
Therefore, given D, and $G_3(s)$) we can determine with non-negligible probability whether or not the output of G_1 and G_2 is psuedorandom
This is a contradiction though because we assumed G_1 and G_2 were PRGs
Therefore, G_3 is a PRG

□

Answer 2

(a) We know from HW1 that computing a^b mod c will take $(n-1) + \frac{n-1}{2}$ multiplications where n is the number of bits for a, b and half of b bits are 1. We know that c and d are both $2k$ bits, so in total we will have $3k$ multiplications to perform. We know that each multiplication takes $4k^2$ bits, so in total, computing c^d mod n will take $12k^3$

(b) To compute $c_1^{d_1}$ mod p_1
c_1 is k bits, d_1 is k bits, so we have $k + \frac{k}{2}$ multiplications. With each multiplication taking k^2 this will take $\frac{3}{2}k^3$

$m = m_1 \bmod p_1$
$m = m_2 \bmod p_2$

Now we can choose an m_x, m_y such that:
$m_x = m_1 \bmod p_1$
$m_x = 0 \bmod p_2$
$m_y = m_2 \bmod p_2$
$m_y = 0 \bmod p_1$
Solving this will be a solution for m because $m_x + m_y = m_1 + 0 \bmod p_1$ and $m_x + m_y = m_2 + 0 \bmod p_2$
$m_x + m_y = m_1 \bmod p_1, m_x + m_y = m_2 \bmod p_2$ Which is what were originally solving
By writing it this way, we know that m_x is a multiple of p_2 and that m_y is a multiple of p_1
$m_x = m_1 * p_2 * p_2^{-1} \bmod p_1p_2$
$m_y = m_2 * p_1 * p_1^{-1} \bmod p_1p_2$
$m = m_1 * p_2 * q_2 + m_2 * p_1 * q_1 \bmod p_1p_2$

We can split this problem up into:
$m = c_1^{d_1} \bmod p_1$
$m = c_2^{d_2} \bmod p_2$
Now, we just want to know $m = (c_1^{d_1}) + (c_2^{d_2}) \bmod p_1p_2$

The runtime of this will just be double that of what we solved for in the first part of $2(b)$. Therefore using CRT, it will run in $3k^3$ which is $\frac{1}{4}$ of the time.

(c) Proof: We want to prove that the value $m_2 + hp_2$ is equivalent to $m_1 \pmod{p_1}$ and equivalent to $m_2 \pmod{p_2}$ and is in the range $0...n-1$. If we can prove all of these, then we can say this value is unique in n and that value is m
First working mod p_1:
$m_2 + hp_2$
$m_2 + (m_1 - m_2)q_2p_2$
$m_2 + m_1 - m_2$ (Because q_2 is the multiplicative inverse of p_2 in p_1
m_1 So we know that this is in fact equivalent to m_1

Now working mod p_2
$m_2 + (m_1 - m_2)q_2p_2$ m_2 (Because the second term is a multiple of p_2
So we know that this is in fact equivalent to m_2

Because h is mod p_1, we know that h must be in the range $0...p_1 - 1$
We also know that m_2 is in the range $0...p_2 - 1$
We also know that $n = p_1p_2$.
In the worst case, h could be $(p_1 - 1)$ and $m_2 = (p_2 - 1)$
$(p_2 - 1) + (p_1 - 1)p_2$
$p_1p_2 - 1$
Therefore, $m_2 + hp_2$ is between $0...n-1$ We have show that $m_2 + hp_2$ is unique in $0...n-1$ and that it is equivalent to m_1 and m_2, therefore $m_2 + hp_2$ is m

$\square$

Answer 3

(a) Proof: Because $p_1 \equiv 3(mod4)$ then $p_1 + 1$ is divisible by 4 This means that u_1 is divisible by 4
This means that u_1 is even
We proved in Homework 2 that any number mod a prime with an even exponent is a square mod p
$t = s^{u_1} \bmod p_1$
Therefore, t is a square

$\square$

(b) Proof by induction: 2^l-th root of s is equal to $s^{u_1^l}$
Base case: $l = 1$
$s^{1/2} \bmod p_1 = s^{\frac{p_1+1}{4}}$
This is true, we proved it in HW2
Now let's assume that $s^{(1/2)^l} \bmod p_1 = s^{(\frac{p_1+1}{4})^l}$
We will prove that this property still holds in $l + 1$
$s^{(1/2)^{l+1}} \bmod p_1 = s^{(\frac{p_1+1}{4})^{l+1}}$
$(s^{(1/2)^l})^{1/2} \equiv (s^{(\frac{p_1+1}{4})^l})^{\frac{p_1+1}{4}}$
This is true because $s^{(\frac{p_1+1}{4})^l}$ is a square mod p_1, and from the proof in HW2, raising this to $\frac{p_1+1}{4}$ is the same as taking the square root
There fore the assumption is true

Let's assume this wasn't a square mod p_1
Then this means that there is some l-th root of s where the $l + 1$ root is not a square
We just proved using induction that the l+1-th root has to be a square, otherwise the l+2-th root would not be a square and out induction would have failed
Therefore the squareroot is itself a square modulo p_1

$\square$

(c) Given x^{2^l}, u_1^l, u_2^l we can compute every x_l mod P_1 and p_2 by computing $(x^{2^l})^{u_{1,2}^l}$ for every l.
This will give us all x's mod each prime.
We can then combine each x using CRT to get all x's mod $p_1 p_2$. Then we can compute each hard-core bit and finally decrypt the message

---

References

http://cs-people.bu.edu/lapets/235-2015-spr/s.php