Department of Computer Science

University of California, Berkeley

CS 294-90, Cryptanalysis

John Manferdelli

Final Exam

Date distributed: May 2, 2013

Date due: May 8, 2013, 6PM PDT

Send completed exam to: [JohnManferdelli@hotmail.com](mailto:JohnManferdelli@hotmail.com)

**Rules of the Road:** Before you read this exam, you may prepare a two-sided (one sheet) set of notes (the “Cheat Sheet”). Once you begin the exam, you may only consult the Cheat Sheet. During the exam, you may use a manual or computer based -calculator. You may not consult others during the exam but you can send me email if you have any questions. You may also use “Excel” or similar tools, using functions you define and simple packaged functions like “mod” and “power,” but not complex functions like “GCD.” Please comply with UC’s policies on academic conduct. You do not have to prove standard results you use but please state any assumptions or theorems you rely on as clearly as you can.

Complete all problems

1. On a machine where you plan to generate symmetric keys for, say, AES, you have access to a single entropy source E. E has the following statistical properties: You can draw four bits of entropy, a1,a2,a3,a4, from E per call. The process generating these bits is not “memory-less” and has the following distribution: Pr(a1=0)= Pr(a1=1)= 1/2. For i>1, Pr(ai=0|ai-1=0)= 2/3, Pr(ai=1|ai-1=0)= 1/3, Pr(ai=1|ai-1=1)= 3/4 and Pr(ai=0|ai-1=1)=1/4. You can call access E many times and each time you receive a four-bit response with the foregoing distribution but there is a time delay between calls and you don’t want to “waste” entropy.
   1. What is the entropy (or information rate) of E? Remember if E consisted of independent, identically distributed bits (which is doesn’t) each with equal probability for drawing 0 or 1, the information rate would be 4 bits.
   2. You wish to use E efficiently to generate keys. Key bit should be independent, identically distributed bits with maximum information rate (i.e.-for 128 bit key, you want 128 bits, **k**, with H(**k**)=128). Design a mechanism to obtain **k** from E. You may assume a cryptographic hash, like SHA-1 is a “perfect” mixer that loses no entropy.
   3. What is the most likely and least likely sequences, a1,a2,a3,a4, that E would generate?
   4. Your ambitious, but lazy, companion decides to use concatenated outputs from E for key generation. If E had been perfect, AES-128, exhaustive search (given a corresponding plain and cipher text) would find the key in about 2127 trials on average. How many trials, on average, would a key search from this flawed distribution take?

|  |  |  |
| --- | --- | --- |
| **Sequence (s)** | **ps** | **ps lg(ps)** |
| 0000 | 0.148148148 | -0.408131482 |
| 0001 | 0.074074074 | -0.278139815 |
| 0010 | 0.027777778 | -0.143609028 |
| 0011 | 0.083333333 | -0.298746875 |
| 0100 | 0.027777778 | -0.143609028 |
| 0101 | 0.013888889 | -0.085693403 |
| 0110 | 0.03125 | -0.15625 |
| 0111 | 0.09375 | -0.320159766 |
| 1000 | 0.055555556 | -0.2316625 |
| 1001 | 0.027777778 | -0.143609028 |
| 1010 | 0.010416667 | -0.068593359 |
| 1011 | 0.03125 | -0.15625 |
| 1100 | 0.0625 | -0.25 |
| 1101 | 0.03125 | -0.15625 |
| 1110 | 0.0703125 | -0.269302148 |
| 1111 | 0.2109375 | -0.473578418 |

Summing column 3 (and multiplying by -1), we get H(X)= 3.583584849 bits

One way to use this source to generate keys (roughly) is to input raw entropy and state to a one-way random function like (SHA-1). For example, let si be a sample from E. Let state1= SHA1(s1) and statei+1= SHA1(statei||si). statei has 3.583584849i bits of entropy and because of the properties of SHA-1, we can sample any [3.583584849i] bits of statei to obtain a [3.583584849i] bit “cryptographically good” random number. For AES-128, where we want a cryptographically secure 128 number, we’d sample E 26 times.

Consulting the table above, the most likely sequence from E is 1111, which occurs with probability ~.21, the least likely sequence is 1010, which occurs with probability ~.01.

If E were “perfect,” the average number of keys we’d need to try to find a 4 bit key is is

nPerfect= = ~2-3.

By sorting the 16 probabilities of sequences from E so that p1≥…≥p16, we can calculate the average number of 4-bit keys we’d need to try to find keys from the “real” E is

nE= ≈ 5.19 trials.

Computing the average number of trials for a bit stream consisting of k-E samples comprising 4k bits by computing the 16k joint probabilities p(1) •…• p(k), where each p(i) takes on all 16 possible 4-bit outcome probabilities for 4-bit sequences generated by E. For k=1, we calculated this value above.

For k=2, we can calculate (with excel or a program)

nE||E≈72 trials

This is substantially lower than the approximately 128 trials expected from a “perfect” distribution. In fact, for a single E-sample, we need to examine 5.19/8≈.648 of the trials we’d have with a perfect source and for two samples, we need only examine 72/128≈.5625 of the number of trials we’d expect for a perfect 8-bit sample. We can make estimates based of the exact ratio but you should have noted that for the 32 E-samples required for AES-128 key generation,

n32E= p(1) •…• p(32),

where pi is the sorted (from largest to smallest) sequence p(1) •…• p(32) where each p(i) takes on the 16 probabilities of E-sequences.

Notice that 2•(3.583584849) ≈7.16 and 27.16≈143. If we had 143 equally likely values, the expected number of trials for finding a preselected one, which we’d find by computing , is about 72, the same value we computed “exactly” above. Since 32•(3.583584849) ≈114.6, this lead us to suspect that n32E≈2113 instead of the approximately nPerfect≈2127 for the perfect distribution. Thus we’d expect to have to try about .00006 of the keys needed for a perfect entropy source.

1. What is a cryptographic hash? What three properties should a cryptographic hash have? Let’s explore one of these.

Suppose we have n objects (n is large) and we select r of them (with replacement). Let’s calculate the probability that we will have r *distinct* (i.e.-non-colliding) objects when we’re done: There are nr ways to pick the r objects. There n ways to pick the first object (without duplication), (n-1) ways to pick the second and so on, the rth object can be selected (n-(r-1)) ways so the probability we will have *no* duplication is: (n/n) ((n-1)/n)…((+1-r)/n). We write this as

Prno-collision(n,r)=  1- (1-)

Now   = ex, so Prno-collision(n,r)= 

1. Set r=a. For what a, is Prno-collision(n, a )= 1/2, 3/4, 99/100? These represent a 50%, 25% and 1% probability of collision
2. Suppose we use two Merkle-Damgard hashes of output size 256 bits ~~(say, SHA-256 and SHA-3 with 256 bit output)~~ in which any collision could be used to produce other collisions by appending the same bit strings to the original (non-identical) colliding strings. ~~Assuming (which is the best case) each has a 50% chance of collision, what is the probability that~~ *~~both~~* ~~have a collision? For what a, is there a 50% probability that both have a collision? What is r in this case? Finally, what r is there a 50% chance that a (good) hash with 512 bits of output has a collision?~~ How much time (in the size of the output hash) does it take to find a string that produces the same hash for both hash algorithms with 25% probability? 50% probability? What do you conclude about building cryptographic hashes out of independent hash functions is?
3. What do you conclude about building cryptographic hashes out of independent hash functions is?

A cryptographic hash is a function h: {0,1}\* 🡪 {1,1}n with the following properties.

One way: It is computationally difficult given y: y=h(x) (x unspecified) to find x.

Pre-image resistance: It is computationally difficult given x,y: y=h(x) (x specified) to find x’≠x: h(x)=h(x’).

Collision resistance: It is computationally difficult to find x’≠x: h(x)=h(x’).

If P(n, a)= t=e so a= √(-ln(t))

|  |  |
| --- | --- |
| t | a |
| 1/2 | 1.177 |
| 3/4 | .759 |
| 99/100 | .142 |

Suppose we find a collision (m1, m2) to H1 in time t1, finding t1 with probability p1 is determined by the above analysis. We can append arbitrary strings, x, to get additional collisions, (m1||x, m2||x) to H1 and eventually, one of these collides for H2; again, the time is dictated by the analysis above. Thus if it takes t1 to find the first collision with p1= .5 and t2 to find the first collision with p2= .5, it will take t1+ t2 with probability .25. To find the probability of both collisions with probability .5, we need to find t1 for p1=1/√2.

Breaking a 512 bit has takes time about 2-256 but breaking two 256 bit hashes seems to takes only ≈2x2-128 so this isn’t a super efficient way to design hashes.

1. The k-linear feedback shift register LFSR(a1, a2,… ak) is defined by xk+n+1= xk+1a1Å xk+2a2Åxk+n ak, where ak0, and all arithmetic is over GF(2). We say xt=LFSRt(a1, a2,… ak). (x1, x2 ,…, xk) is the key and the plaintext message is m1, m2 ,…, ml, the ciphertext message is m1Åx1, m2Åx2 ,…, xlÅml.. Given a corresponding plaintext and ciphertext messages of length t, “bread” this cipher system. What is the minimum size to t for k?

An LFSR with k stages can be broken given 2k consecutive bits since we can then solve for the ci as element d of GF(2) in:

a1x1+…+akxk= xk+1

a1x2+…+akxk+1= xk+2

… … …

a1xk+…+akx2k-1= x2k

We know the xi because we have corresponding plain, ri, and cipher text , si, with ri+ xi = si (these equations are over GF(2)) and so riÅ si= xi. Even if we don’t know the length of the shift register, we can do trial solutions. We know we’ve gotten enough because

x1, x2, …, xk

det x2, x3, …, xk+1

… … ….

xk, xk+1, …, x2k-1

will be 0 in GF(2) when k is exceeds the number of shift registers.

1. Describe its construction of DES in terms of basic transformations. What is the role of the key schedule? S-boxes? What is linear cryptanalysis? Consider a DES like cipher with the same key schedule and same high level single round, namely, ri: (L,R)🡪(R, L+ f(K(i), R), however, f is different in the following respects: (1) there is no expansion matrix and the first 32 key bits of the traditional DES round key is xored with R to produce the S-box input, (2) the permutation matrix P is replaced by the identity permutation, and (3) there is a single S-box, S, which takes four bit inputs and produces four bit outputs. The S-box is applied to each four bits of K(i)+R, in succession. Thus f(K(i), R)= S(K(i)+R)1,2,3,4 || S(K(i)+R)5,6,7,8 ||… || S(K(i)+R)29,30,31,32. S is defined as:

S(t,u,v,w)= (t+tw, u+uv, v+uvw, w+tw).

Discuss the linear and differential characteristics of this per round function and analyze generally, but without implementing, the prospects for linear and differential cryptanalysis of this modified cipher.

DES is a map from E: {0,1)64 x {0,1}56 🡪 {0,1)64. We write E(P,K)=C. For fixed key, K, E is a bijection from {0,1)64 to {0,1)64. For fixed K, DES is a composition of 33 permutations: E= IP-1 s16 t s15 t… t s1 IP. t maps (L,R)🡪(R,L) where L and R are respectively the left and right 32 bit subwords of the input to t. si: (L,R)🡪 (LÅf(K(i)ÅE(R)), where K(i) is the 48 bit keys schedule for round i, specified by the key schedule algorithm in FIPS-46 and E is the “expansion” operation (taking {0,1}32 🡪 {0,1)48) in FIPS-46. Each si is an involution – that is si2=1. t is also and involution and as a result, running DES backwards produces the inverse permutation: E-1(C,K)= IP-1 s1 t s2 t… t s16 IP. The round transformation is thus ri= t s15 (the last round omits t).

The f function is built from 8 S-boxes, S1,…,S8. Each S box is a map {0,1}6🡪{0,1}4 and a permutation, P (specified in FIPS-46) which permutes the bits in a 32 bit block. f(b1, …, b48)= P(S1(b1, …, b6)|| S2(b7, …, b12)||…|| S2(b41, …, b48)).

The key schedule mixes key bits so each key bit affects almost every output disk-the key schedule *diffuses* the effect of key bits. The S-boxes act as non-linear transformations, which *confuses* the contribution of each input bit to the final output.

Linear cryptanalysis is a technique, which consists of finding linear relations

a, b, g between the inputs of an S-box, the outputs of an S-box and the key bits used by the S-box in a round [a(p)Åb(c)=g(k)] which hold with probability p= 1/2+eij and then“stitching” these together to obtain a linear constraint among the plaintext, cipher and key bits of the form:

PR[17]ÅPL[3,8,14,25]ÅCL[3,8,14,25]ÅCR[17]= K1[26]ÅK3[26].

These equations are used to find key bits faster than exhaustive search.

For the S above, we have S(t,u,v,w)[1]ÅS(t,u,v,w)[4]=tÅw. Let switch to GF(2) notation. A[a,b] means add bit a of A to bit b of A (over GF(2)). We thus have:

Li[1+4k, 4+4k]ÅRo[1+4k, 4+4k]= Ri[1+4k, 4+4k]Åkr[1+4k, 4+4k], k=0,1,…,7

This holds with p=1 (You should be so lucky in real life). Note that Ri= Lo. Extending the first constraint to two rounds (see figure below), we get:

L1[1, 4]ÅR2[1, 4]= R1[1, 4]Åk1[1, 4], R1= L2

L2[1, 4]ÅR3[1, 4]= R2[1, 4]Åk2[1, 4], R2= L3

Adding and cancelling, we get:

L1[1, 4]~~ÅR~~~~2~~~~[1, 4]ÅL~~~~2~~~~[1, 4~~]ÅR3[1, 4]=

~~R~~~~1~~~~[1, 4]Å~~k1[1, 4]~~ÅR~~~~2~~~~[1, 4~~]Åk2[1, 4], since R1= L2

Or,

L1[1, 4]ÅR3[1, 4]= k1[1, 4]Åk2[1, 4].

Similarly,

L4[1, 4]ÅR6[1, 4]= k4[1, 4]Åk5[1, 4],

L7[1, 4]ÅR9[1, 4]= k7[1, 4]Åk8[1, 4].,

L10[1, 4]ÅR12[1, 4]= k10[1, 4]Åk11[1, 4]

L13[1, 4]ÅR15[1, 4]= k13[1, 4]Åk14[1, 4].

Noting that R6= L7, R9= L10, R12= L13, and R15= L16, and adding, this becomes:

L1[1, 4]ÅL16[1, 4]= k1[1, 4]Åk2[1, 4]Åk4[1, 4]Åk5[1, 4]Åk7[1, 4]Åk8[1, 4]

Åk10[1, 4]Åk11[1, 4]Åk13[1, 4]Åk14[1, 4].

There is no switch in the last round and CL= L17, CR= R16= R17, so

L16[1, 4]Å CL[1, 4]= CR[1, 4]Åk16[1, 4].

Finally, L1= PL and R1= PR. Adding all these, we get:

PL[1, 4]ÅCL[1, 4]= k1[1, 4]Åk2[1, 4]Åk4[1, 4]Åk5[1, 4]Åk7[1, 4]Åk8[1, 4]

Åk10[1, 4]Åk11[1, 4]Åk13[1, 4]Åk14[1, 4]Åk16[1, 4]ÅCR[1, 4].

Further,looking at the inverse cipher we get another relation. Now similar

Equations occur for positions 1+4k, 4+4k, for k= 0,1,…7, so, all together we have 16 linear constraints. We need only guess 40 bits and these equations (plus corresponding plain and cipher text) give us the remainder!

Calculating the differential: D(x,y)= |{t: S(x)ÅS(xÅt)=y}|, we get

SBox, S(t,u,v,w)= (t+tw, u+uv, v+uvw, w+tw)

x: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

y: 00 01 02 03 04 05 02 01 08 00 0a 02 0c 04 0a 00

The convention, as with DES, is that the high order bit is first, that is, S(x1x2x3x4)=y1y2y3y4. Thus 0x01 has t=0, u=0, v=0 and x=1.

Differences, S(t,u,v,w)= (t+tw, u+uv, v+uvw, w+tw)

y': 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00, x'= 00

00 06 00 02 00 00 00 00 06 00 02 00 00 00 00 00, x'= 01

00 00 08 00 04 00 04 00 00 00 00 00 00 00 00 00, x'= 02

00 00 00 04 00 02 00 02 00 00 04 00 02 00 02 00, x'= 03

04 00 04 00 08 00 00 00 00 00 00 00 00 00 00 00, x'= 04

00 02 00 02 00 04 00 00 02 00 02 00 04 00 00 00, x'= 05

04 00 04 00 00 00 08 00 00 00 00 00 00 00 00 00, x'= 06

00 02 00 02 00 00 00 04 02 00 02 00 00 00 04 00, x'= 07

00 08 00 00 00 00 00 00 08 00 00 00 00 00 00 00, x'= 08

06 00 02 00 00 00 00 00 00 06 00 02 00 00 00 00, x'= 09

00 00 00 04 00 04 00 00 00 00 04 00 00 00 04 00, x'= 0a

00 00 04 00 02 00 02 00 00 00 00 04 00 02 00 02, x'= 0b

00 00 00 04 00 04 00 00 04 00 00 00 04 00 00 00, x'= 0c

02 00 02 00 04 00 00 00 00 02 00 02 00 04 00 00, x'= 0d

00 04 00 00 00 00 00 04 00 00 04 00 00 00 04 00, x'= 0e

02 00 02 00 00 00 04 00 00 02 00 02 00 00 00 04, x'= 0f

There are good intra-s-box differentials; for example 0x00🡪0x02, p=.5, 0x04🡪0x04, p=.5, 0x08🡪0x00, p=.5 and 0x08🡪0x08, p=.5. Because of the lack of mixing (provided by the P transformation in DES), these differentials are easy to track through an entire encryption – the p=.5 Sbox differentials, for example, give a full cipher characteristic that holds with p= 2-16 - so differential analysis offers a significant speedup. Finally, the intra-Sbox differentials happen in each S-box position so there are lots of them across a round.

1. Describe the RSA public key system. Key generation, basis for safety, encryption process. Suppose p= 1493 and q= 1499. Calculate n and f(n). If e=5 is the encryption exponent, calculate the decryption exponent, d.

n=p⋅q=1493⋅1499=2238007.

p-1= 1492= 4⋅373, q-1= 1498=2⋅749

φ(2238007)= 4⋅373⋅749= 558754

Want to find x, y: 5x+558754y=1

558754= 111750+4, 5=4+1 so 5(111751)+ 558754(-1)=1

So d= 111751.

1. Factoring using the x2=y2 (mod n) a la quadratic sieve. Suppose n=3837523. Observe that 93982= 55 x191 (mod n), 190952= 22x51x111x131x191 (mod n), 19642= 32 x 133 (mod n), and 170782= 26x32x111(mod n). Use these to find (x,y): x2=y2 (mod n). Finally, calculate (x-y, n) where (a,b) is the gcd of a and b to find the factors of n.

93982= 55 x191 (mod 3837523)

190952= 22x51x111x131x191 (mod 3837523)

19642= 32 x 133 (mod 3837523)

170782= 26x32x111(mod 3837523)

So

(9398•19095•1964•17078)2= (24•32 •53•11•132•19)2.

x=9398•19095•1964•17078= 2230387 and y=24•32 •53•11•132•19= 2586705.

y-x= 356318 and (356318, 3837523)= 1093.

3837523/1093=3511. 1093•3511=3837523.

1. Describe a discrete log public key cipher over a finite field of characteristic p. What is the public key? The private key? Describe the encryption process using a “small” p (for example, p= 3467). Set k=[]=58. Suppose a=5 and b =2717 where b= ax (mod p). Find x as follows (Baby step, giant step). Compute a table (a, aj (mod p)), for j= 1,2,…,k. Now compute, ba-kj (mod p), for j= 1,2,…,k and find the intersection in the first table. Compute x from this. Finally, describe the Diffie Hellman key exchange protocol using exponentiation mod p.

|  |  |
| --- | --- |
| **n** | **5n (mod 3467)** |
| 1 | 5 |
| 2 | 25 |
| 4 | 625 |
| 8 | 2321 |
| 16 | 2790 |
| 32 | 685 |
| 64 | 1180 |

Using this, we compute the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **5n (mod 3467)** | **n** | **5n (mod 3467)** |
| 1 | 5 | 30 | 2801 |
| 2 | 25 | 31 | 137 |
| 3 | 125 | 32 | 685 |
| 4 | 625 | 33 | 3425 |
| 5 | 3125 | 34 | 3257 |
| 6 | 1757 | 35 | 2417 |
| 7 | 1851 | 36 | 1684 |
| 8 | 2321 | 37 | 1486 |
| 9 | 1204 | 38 | 496 |
| 10 | 2553 | 39 | 2480 |
| 11 | 2364 | 40 | 1999 |
| 12 | 1419 | 41 | 3061 |
| 13 | 161 | 42 | 1437 |
| 14 | 805 | 43 | 251 |
| 15 | 558 | 44 | 1255 |
| 16 | 2790 | 45 | 2808 |
| 17 | 82 | 46 | 172 |
| 18 | 410 | 47 | 860 |
| 19 | 2050 | 48 | 833 |
| 20 | 3316 | 49 | 698 |
| 21 | 2712 | 50 | 23 |
| 22 | 3159 | 51 | 115 |
| 23 | 1927 | 52 | 575 |
| 24 | 2701 | 53 | 2875 |
| 25 | 3104 | 54 | 507 |
| 26 | 1652 | 55 | 2535 |
| 27 | 1326 | 56 | 2274 |
| 28 | 3163 | 57 | 969 |
| 29 | 1947 | 58 | 1378 |

ax =558= 1378 (mod 3467).

(-595)3467+(1497)1378=1, so 5-58= 1497 (mod 3467).

ba-kj(mod 3467)= 2717•1497= 558(mod 3467).

From the table above, 558= 515 (mod 3467). So 2717•5-58= 515 (mod 3467), and 2717= 558+15 (mod 3467). Thus x=73.

**Diffie-Hellman:** Alice and Bob agree on a base, g and modulus p. Alice generates a nonce, a, and sends ga (mod p). Bob generates a nonce, b, and sends gb (mod p). The shared key is gab (mod p).

1. Suppose Ep(a,b) is the set of points (including the point at ) on the equation y2=x3+ax+b. Recall Ep(a,b) is non-singular if D=4a3+27b20. How many points are on E23(2,13)? Describe ECC encryption on E23(2,13)? In the role of Alice, pick a public key for an ECC public key system on E23(2,13). Show how to embed the message m=7 in an point PM on E23(2,13). In the role of Bob encrypt the message and in the role of Alice, decrypt it. What was the most computationally expensive procedure called for in this problem?

|  |  |
| --- | --- |
| **n** | **n2 (mod 23)** |
| 0 | 0 |
| 1 | 1 |
| 2 | 4 |
| 3 | 9 |
| 4 | 16 |
| 5 | 2 |
| 6 | 13 |
| 7 | 3 |
| 8 | 18 |
| 9 | 12 |
| 10 | 8 |
| 11 | 6 |
| 12 | 6 |
| 13 | 8 |
| 14 | 12 |
| 15 | 18 |
| 16 | 3 |
| 17 | 13 |
| 18 | 2 |
| 19 | 16 |
| 20 | 9 |
| 21 | 4 |
| 22 | 1 |

|  |  |  |
| --- | --- | --- |
| **x** | **x3+2x+13 (mod 23)** | **y** |
| 0 | 13 | ±6 |
| 1 | 16 | ±4 |
| 2 | 2 | ±5 |
| 3 | 0 | 0 |
| 4 | 16 | ±4 |
| 5 | 10 | --- |
| 6 | 11 | --- |
| 7 | 2 | ±5 |
| 8 | 12 | ±9 |
| 9 | 1 | ±1 |
| 10 | 21 | --- |
| 11 | 9 | ±3 |
| 12 | 17 | --- |
| 13 | 5 | --- |
| 14 | 2 | ±5 |
| 15 | 14 | --- |
| 16 | 1 | ±1 |
| 17 | 15 | --- |
| 18 | 16 | ±4 |
| 19 | 10 | --- |
| 20 | 3 | ±7 |
| 21 | 1 | ±1 |
| 22 | 10 | --- |

So the curve has 28 elements, counting ∞.

Recall the addition laws:

If Ep(a,b): y2=x3+ax+b (mod p), P1=(x1,y1), P2=(x2,y2), then P1+P2= P3 and

P3=∞, if x1=x2 and y1=-y2.

Otherwise P3=(x3,y3) where

λ= (y2-y1)/(x2- x1) (mod p), if x1≠x2

λ= (3x12+a)/(2y1), if x1=x2

x3= λ2-x1-x2  (mod p),

y3= λ(x1-x3)- y1  (mod p).

(1,4) is a point of order 14 in the elliptic curve group and (0,0) is an element of order 2. 7(1,4)= (3,0) is another element of order 2.

For m=7, we check to see if there is a point on Ep(a,b) whose x coordinate is 14. There is, namely (14,5), so PM= (14,5). Suppose Alice’s secret is 3 so the public key parameters are E23(2,13). Alice picks G=(1,4) as the base point and publishes PA=3(1,4)=(18,4). Bob picks the secret b=5 and sends (5G, 5PA+PM). 5G=(20,16), 5PA=5(18,4)=(0,17) and 5PA+PM=(21,1)+(14,5)=(1,4). Alice computes 3(5G)= (21,1) and subtracts this from (1,4) to recover the message point (14,5). She divides the x-coordinate by 2 to get the message 7.

I hope you enjoyed the class. Please feel free to contact me if you have any questions in the future.

John