If the group order is composite, DDH is easier to solve than CDH by looking for contradictions modulo the prime divisors of the group order.

Pohlig-Hellman shows how this simplifies attacking the DLP by solving it modulo each of the prime factors.See pdf on course page.

Question: Why is g^((2a'+1)*(p-1)/2)=-1?
the order of F_p^* is p-1, so g^(p-1) = 1 mod p
(2a'+1)*(p-1)/2 = 2a'*(p-1)/2 + (p-1)/2 = a'*(p-1) + (p-1)/2

g^((2a'+1)*(p-1)/2) =g^(p-1) * g^((p-1)/2) = 1* g^((p-1)/2) = -1

if g is a generator then the smallest exponent of g that gives 1 is p-1
aka the order of g is p-1, so g^((p-1)/2) \ne 1, thus it is the other
square root of +1, namely -1

Example in pari:

```
p=1013
g=Mod(3,p)
h=Mod(321,p)
znorder(g)
factor(%)  //% is the result of znorder(g) [2,  2]
                [11,1]
                [23,1]
//PH solves the DLP in subgroups of order 2 (twice), 11, and 23

2*2+11+23=38 steps // this is the number of steps needed in the worst case for PH, much less than p-1 by using brute force.

//set up the target and base  in the subgroup of order 23:
g23 = g^((p-1/23))  // takes 23 steps to 1, so this has order 23
h23 = h^((p-1/23))
g23^2
g23^3
g23^4
...
g23^13 // same result as h23
a23 = Mod(13,23)

//now the same for the prime divisor 11 of p-1
g11 = g^((p-1)/11)
znorder(g11) // verification, yes, this does indeed have order 11
h11 = h^((p-1)/11)
g11^2
...
% * g11 // more efficient way (one mult rather than one exp per step), get same result as h11 at 6 iteration (power 6)
g11^6 - h11 // verification
g23^13 - h23 // verification
```

```
a11 = Mod(6,11)

//now we handle 2 and 2^2; folloing the steps as in the Pohlig-Hellman notes on the course page
h2=h^((p-1)/2) // argue that g2 is -1, see on top of this page
a2 = Mod(0,2)
hp = h/g^0 // same as h ; hp stands for h'
hp^((p-1)/4) // two possiblities, +1 or - 1
a2 = Mod(0+1*2, 4)
chinese(a2, a11)
chinese(%, a23) // output Mod(358,1012)
a = 358 // from previous result
g^a // same as h, correct
```

Another example, generated on the fly, so this includes the generation process

```
q=2*3*3
l=11
isprime(q*l+1)
l=nextprime(l+1)
isprime(q*l+1)
l=nextprime(l+1)
isprime(q*l+1)
p=q*l+1
factor(p-1) //[2,  1]
               [3,  3]
               [17,1]
znorder(Mod(2,p))
znorder(Mod(3,p))
znorder(Mod(5,p))
znorder(Mod(7,p))
g=Mod(7,p)
h=Mod(731,p)// randomly picked

//handle divisor 2
h2=h^((p-1)/2)
a2=Mod(1,2)

//handle divisor 3^3, by computing the coefficients of the base-3 expansion of a mod 27
h3 = h^((p-1)/3)
g3 = g^((p-1)/3)
%^2
a3 = 2
hp = h/g^2
// we know that hp has ap = 0 mod 3
hp^((p-1)/9) // result is 1
a3 = 2 + 0*3
hp = hp/(g^(0*3)) // not actualy an update as we got 0
hp^((p-1)/27) // result is 866, mathing g3
a3 = a3 + 1*3^2//equals 11
```

```
g^(11*(p-1)/27)
h^((p-1)/27) // same
a3 = Mod(a3,27)

//handle divisor 17
h17 = h^((p-1)/17)
g17 = g^((p-1)/17)//well, that was easy, match on first try
a17 = Mod(1,17)

//combine the results
chinese(a17,a3)
chinese(%, a2)
g^443 - h // verification, it is 0
```

Some more comments on Pohling-Hellman

There are 3 versions for handling l^e (l prime, l^e | (p-1)

1. solve one big DLP in the group of order l^e --- not a good idea
2. solve e DLPs in groups of size l by updatinng the target to h' but keeping the same table
3. solve e DLPs in groups of size l by updating the tables

The middle option is what I want you to use, as it is 1 computation to update h' while it is
l operations to update the tables.

I showed the 3rd option in the process of reinventing PH, but this is not the final version!

Here is the difference, explained on our second example:

We know a = 2 + 3* ...
want to find a mod 9 , so the next coefficient in the base-3 expansion

Third option:
target h^((p-1)/9) is one of the values of g^2, g^(2+(3*(p-1)/9)), g^(2+(2*3*(p-1)/9))
so this means updating the table for the comparisons to g^2, g^(2+(3*(p-1)/9)), g^(2+(2*3*(p-1)/9))
which costs 3 multiplications (by g^2) starting from the table g^0, g^((p-1)/3), g^(2*(p-1)/3).

Secnd option:
updating h to h' gets

h' = h/g^2 = g^(3*(...))
g^(3a'*(p-1)/9)=
g^(a'*(p-1)/3) this matches g^((p-1)/3) or one of its powers, so we can use the old table.
after one division by g^2 (or, rather, one multiplication by (g^(-1))^2 for precomputed g^(-1)

Both methods need the exponentiation ^((p-1)/9) but the base differs.

Rewriting things mod l|(p-1), l large
<g> subgroup of order l in F_p^*
get such a g by

a) if given G generating F_p^* then putting g = G^((p-1)/l)
b) by picking random r^((p-1)/l) and putting g = r^((p-1)/l) if this is \ne 1
   else, pickickinng another r
   This works in (l-1) of l cases, so much faster than first finding G and then doing a)

DH and keygen for ElGamal: all the same as before, but using g and exponents in [0,l-1] (probably don't want to choose 0 or 1; definitely don't choose 0)

ElGamal enc: g^k with 0<k<l, c = h_A^k *m <- still all modulo p

ElGamal sign: g^k mod p with 0<k<l, s = k^(-1) (h(m) + ra) mod l <- this one is updated to using l

Stay tuned for DSA to see how to get a signature scheme that needs less space for the signature -- just two elements mod l rather than one mod p and one mod l.