Lily Haas, Ashok Khare, & Emma Roskopf

**Diffie-Hellman**

Alice and Bob agree on g = 11 and p = 59.

Alice sent Bob the number 57.

Bob sent Alice the number 44.

Alice's $A = g^x \bmod p$

- So we are going to solve this for x, where $x < p$
- $57 = 11^x \% 59$
- It can only go up to 59, so I have a finite range to check
    - I didn't want to do each of these calculations so instead I had my computer check it
    - So I ran this code:

```
for x in range (1,59):
    if (11**x)%59 == 57:
        print(x)
```

    - The output was 36, so x=36
- Alice is now going to compute $B^x \bmod p$
    - $44^{36} \bmod 59$
    - This is 36, so that's the secret key

Bob's $B = g^y \bmod p$

- So we are going to solve this for y, where $y < p$
- $44 = 11^y \% 59$
- It can only go up to 59, so I have a finite range to check
    - I didn't want to do each of these calculations so instead I had my computer check it
    - So I ran this code:

```
for y in range (1,59):
    if (11**y)%59 == 44:
        print(y)
```

    - The output was 15, so y=15
- Bob is now going to compute $A^y \bmod p$
    - $57^{15} \bmod 59$
    - This is 36, so that's the shared secret key, and it's the same as before

**Where we would have failed if the integers were larger**

As the integers we're dealing with get much larger, the loop checking for valid values also gets a lot longer, rather than 59 values, Eve's loop might have to check millions. This loop then becomes much more costly because it's far longer, and each loop is also more costly because we're exponentiating these really big numbers too.

**RSA**

```
Bob's public key (e_Bob, n_Bob) = (13, 5561)

[1516, 3860, 2891, 570, 3483, 4022, 3437, 299,
 570, 843, 3433, 5450, 653, 570, 3860, 482,
 3860, 4851, 570, 2187, 4022, 3075, 653, 3860,
 570, 3433, 1511, 2442, 4851, 570, 2187, 3860,
 570, 3433, 1511, 4022, 3411, 5139, 1511, 3433,
 4180, 570, 4169, 4022, 3411, 3075, 570, 3000,
 2442, 2458, 4759, 570, 2863, 2458, 3455, 1106,
 3860, 299, 570, 1511, 3433, 3433, 3000, 653,
 3269, 4951, 4951, 2187, 2187, 2187, 299, 653,
 1106, 1511, 4851, 3860, 3455, 3860, 3075, 299,
 1106, 4022, 3194, 4951, 3437, 2458, 4022, 5139,
 4951, 2442, 3075, 1106, 1511, 3455, 482, 3860,
 653, 4951, 2875, 3668, 2875, 2875, 4951, 3668,
 4063, 4951, 2442, 3455, 3075, 3433, 2442, 5139,
 653, 5077, 2442, 3075, 3860, 5077, 3411, 653,
 3860, 1165, 5077, 2713, 4022, 3075, 5077, 653,
 3433, 2442, 2458, 3409, 3455, 4851, 5139, 5077,
 2713, 2442, 3075, 5077, 3194, 4022, 3075, 3860,
 5077, 3433, 1511, 2442, 4851, 5077, 3000, 3075,
 3860, 482, 3455, 4022, 3411, 653, 2458, 2891,
 5077, 3075, 3860, 3000, 4022, 3075, 3433, 3860,
 1165, 299, 1511, 3433, 3194, 2458]
```

Each character in Alice's message was computed using $x^{eB} \bmod n_B$ where x is the ASCII representation of the character, $e_B$ is 13 and $n_B$ is 5561.
We can figure out Bob's secret character by finding the prime factors of 5561. There are only two, 67 and 83. From this we can find $\lambda(n_B) = \text{lcm}(66, 82) = 2706$
With $\lambda(n_B)$ we can find $d_B$ such that $e_B d_B \bmod \lambda(n_B) = 1$. So $13 * d_b \bmod 2706 = 1$

We run the following code to make the computer do this for us because we do not want to brute force this by hand. We did not know the number would be as low as 6 and did not want to risk spending the entire night doing math (because we have technology).

```
x = 0
while ((((x*2706 + 1)/13)) % 1 != 0):
  x+= 1
print(x)
```

The code above gives us x=6. If we multiply 6 by 2706, add 1, and then divide by 13, we get 1249. Thus, $d_B = 1249$. We checked our work by having Python calculate (13*1249) % 2706, which was equal to 1.

To decrypt the list, we take the list of encrypted letters (named list) and run them through this code, which computes $y^{(d_B)} \bmod n_B$

```
db = 1249
nb = 5561
ans = []
for num in list:
  temp = (num**db) % nb
  ans.append(temp)
print(ans)
```

Here's is the decrypted list:
[72, 101, 121, 32, 66, 111, 98, 46, 32, 73, 116, 39, 115, 32, 101, 118, 101, 110, 32, 119, 111, 114, 115, 101, 32, 116, 104, 97, 110, 32, 119, 101, 32, 116, 104, 111, 117, 103, 104, 116, 33, 32, 89, 111, 117, 114, 32, 112, 97, 108, 44, 32, 65, 108, 105, 99, 101, 46, 32, 104, 116, 116, 112, 115, 58, 47, 47, 119, 119, 119, 46, 115, 99, 104, 110, 101, 105, 101, 114, 46, 99, 111, 109, 47, 98, 108, 111, 103, 47, 97, 114, 99, 104, 105, 118, 101, 115, 47, 50, 48, 50, 50, 47, 48, 52, 47, 97, 105, 114, 116, 97, 103, 115, 45, 97, 114, 101, 45, 117, 115, 101, 100, 45, 102, 111, 114, 45, 115, 116, 97, 108, 107, 105, 110, 103, 45, 102, 97, 114, 45, 109, 111, 114, 101, 45, 116, 104, 97, 110, 45, 112, 114, 101, 118, 105, 111, 117, 115, 108, 121, 45, 114, 101, 112, 111, 114, 116, 101, 100, 46, 104, 116, 109, 108]

Using an ASCII converter, we get back this message:

> Hey Bob. It's even worse than we thought! Your pal, Alice.
> https://www.schneier.com/blog/archives/2022/04/airtags-are-used-for-stalking-far-more-than-previously-reported.html

**Where would we have failed if the integers were larger?**
If the integers were larger, we would have failed because we would not have been able to easily determine $p_{Bob}$ and $q_{Bob}$ from $n_{Bob}$. This is because if $n_{Bob}$ were sufficiently large, it could have multiple pairs of prime factors - if there were enough pairs, then this problem would become computationally infeasible to solve.

**Why would RSA one letter at a time still not be secure?**
By encoding one letter at a time we are essentially using a substitution cipher (each letter and symbol is replaced with a sequence of numbers) that could be broken with letter frequency.