Math 406 Section 8.3: Exponentiation Ciphers

1. **Introduction:** In section 8.1 we did $C \equiv aP + b \bmod 26$ but this is not the only operation we could do.

   First off we'll modify the table of letters slightly:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

   We can then group letters together with no ambiguity, for example `JU` can be assigned the number 0920 or just 920. Without the modification it would be unclear what 111 meant, 1 followed by 11 or the reverse.

   Also a reminder:

   Fermat's Little Theorem: If p is prime and $a \in \mathbb{Z}$ with $p \nmid a$ then $a^{p-1} \equiv 1 \bmod p$.

2. **Exponentiation Ciphers:**

   (a) **Encryption:**

   Let p be an odd prime and let e be a positive integer with $\gcd(e, p-1) = 1$.

   We take the plaintext and group the letters into groups such that the value of no group could possibly be greater than or equal to p.

   So for example if $p = 3001$ then we group into blocks of 2 since the largest value would then be $2525 < 3001$ where 2525 corresponds to `ZZ`. If $p = 377173$ then we group into blocks of 3 since the largest value would then be $252525 < 377173$ where 252525 corresponds to `ZZZ`.

   We pad with junk letters at the end if needed so that the plaintext length is a multiple of the block length. Traditionally `X` is used but it doesn't matter.

   For encryption Alice needs to know the encryption key pair (e, p) and then for a ciphertext block C she does:

   $$C \equiv P^e \bmod p$$

   Note that the result may not be convertible back to characters so we just send the numbers.

   Example: Alice uses $(e, p) = (479, 3001)$. To encrypt `LOVENOTE` she divides it up into blocks of two and encrypts using

   $$C \equiv P^{479} \bmod 3001$$

	LO	VE	NO	TE
	1114	2104	1314	1904
	1114^{479}	2104^{479}	1314^{479}	1904^{479}
$\equiv$	0169	0317	0017	1697

   The overall cyphertext is then `0169 0317 0017 1697`. the spaces aren't necessary they just make it clearer.

   (b) **Decryption:**

   This process is invertible since the fact that $\gcd(e, p-1)$ guarantees that there exists some d with $de \equiv 1 \bmod p-1$ then then for a ciphertext block C we have:

   $$C^d \equiv (P^e)^d \equiv P^{ed} \equiv P^{1+k(p-1)} \equiv P(P^{p-1})^k \equiv P(1)^k \equiv P \bmod p$$

   Here the fact that $P^{p-1} \equiv 1 \bmod p$ is guaranteed by Fermat's Little Theorem. Note that $p \nmid P$ since $P < p$.

Thus for decryption Bob needs to know the decryption key pair (d, p).

Example: Bob knows $(d, p) = (119, 3001)$ corresponding to Eve's $(e, p) = (479, 3001)$. He receives `2672 0317 1665 2110 0246 1749 0017 2112` which he decrypts using:

$$P \equiv C^{119} \bmod 3001$$

	2672	0317	1665	2110	0246	1749	0017	2112
	2672^{119}	0317^{119}	1665^{119}	2110^{119}	0246^{119}	1749^{119}	0017^{119}	2112^{119}
$\equiv$	1800	2104	2414	2017	1804	1105	1314	2223
	`SA`	`VE`	`YO`	`UR`	`SE`	`LF`	`NO`	`WX`

The message is obviously `SAVE YOURSELF NOW` padded with an `X` to make the length a multiple of two characters.

3. **Breaking Exponentiation Ciphers** If Eve knows (e, p) she then knows $p - 1$ and then d can be found by the Euclidean Algorithm. Just a reminder, this is because $\gcd(e, p-1) = 1$ and so Eve can find α, β with:

$$\alpha e + \beta(p-1) = 1$$

and if she reduces mod $p - 1$ she gets:

$$\alpha e \equiv 1 \bmod p - 1$$

and she can let $d = \alpha$.

Example: If Alice uses $(e, p) = (689, 3343)$.

If Eve finds this out then she knows that $\gcd(e, p-1) = \gcd(689, 3342) = 1$ and so she uses the Euclidean Algorithm to solve:

$$689\alpha + 3342\beta = 1$$

and finds:

$$689(941) + 3342(-194) = 1$$

She then reduces mod 3342 to get:

$$689(941) \equiv 1 \bmod 3342$$

and so $d = 941$. She can then decrypt anything Alice sends.

As long as Alice and Bob keep this information to themselves then things are safe, but if Alice uses this same (e, p) for someone else then Bob can decrypt it. What we see happening here is that knowing the encryption key pair means that the decryption key pair can be easily calculated.

Would it be possible to make the encryption key pair public but have it still practically impossible to calculate the decryption key pair?