(Team details can be found at the end of the assignment)

Cipher Method: *RSA*

Passcode: jhvkHJKKJf

Justification:

i. On the first screen only *exit2* is open, so we enter the command, ***exit2***.
ii. On the next screen it says, that there are 5 exits and exit 5 is closed with a panel next to it. We tried many commands from *exit1* to *exit4* but we weren't proceeding to any new screens. Then it caught our attention that the two characters on top were changing. They were in Chinese, so we searched what they stand for. Only one command from each screen was taking us to a new screen with new combination of characters. Other exits took us to any of the previously encountered screens.
iii. And at the last screen no exit worked, hence we tried commands like enter, read etc. and **read** worked. And we got to the panel.
iv. Following are the commands we inserted to reach the panel:

Screen	Characters On Screen	Command
1	六五	exit2
2	五七	exit4
3	七六	exit3
4	七五	exit1
5	六犬	exit4
6	五六	exit4
7	犬五	exit2
8	犬七	exit2
9	六七	exit1
10	七犬	read

v. On the next screen we got following problem:

n=3665612238752088973428342512040280866180876521016748157
676717987121762462558702989000030906031706869041772288316
637853776541229449921758787761044652698362841759808868204
706301539672683780672062357856481509380205841326089007126
536659711979070365055685036739582065217508838960471544262
444328591568687961

Wannabe_Eves: This door has RSA encryption with exponent 5 and the password is
200277673179114748200829385598541462951441634361742595673
233220555425791752976395446371756859418416131397297091355
364943695617535599254024629898255118661869793199136422649
325241375809360875589098380669177595887258785184110062708
65040544819491508605933821295536906185838933939345187355
7454259534077799

vi. RSA encryption and decryption works as follows:
    a. Encryption: $C = M^e \bmod N$
    b. Decryption: $M = C^d \bmod N$

vii. For decrypting the above password, we can either find factors of *N* which is impossible to find as n as its length is too large (999 bits). Next, we could try and compute *d*, but as N couldn't be factorized, we cannot find *phi(N)* and thus cannot efficiently compute *d* as well.

viii. Now, as the public exponent is 5 (which is small) we can use low-exponent attack (Coppersmith's Algorithm).

ix. This algorithm requires a polynomial as an input; thus, we need to formulate the same. For this, we first need to check if any *padding* is added to the Message. This can be done by checking if $C^{1/e}$ is an integer or not.

x. We computed the same and found that *padding* is added. Let *p* be the *padding*, thus final equation becomes:

$$(p + M)^e = C \bmod N$$

xi. In the above equation, e, C, and N are known. We will try to guess *p* as Coppersmith says that if we are looking for $N^{1/e}$ of the message, it is then a *small root* and we should be able to find it pretty quickly.

Coppersmith's Theorem:

Let N be an integer and f be a polynomial of degree δ. Given N and f, one can recover in polynomial time all x_0 such that $f(x_0) = 0 \bmod N$ and $x_0 < N$ 1/δ.
So, we can form the problem as follows $f(M) = (p + M)^e \bmod N$.
For solving this, we use the code from coppersmith.sage. We modified this code as follows to compute the polynomial modulo N:

1. We translated *padding p* to its binary form *p_bin*.
2. The length of password M is unknown, but from our assumption; $x_0 < N^{1/e}$ (= 10^6) thus, M can't be longer than 200 bits.
3. Thus, the final polynomial becomes: $((p\_bin \ll \text{length\_M}) + M)^e - C$
4. Root of the above polynomial is the required password and can be calculated using Coppersmith's Algorithm and LLL (Lattice reduction).

Different paddings *p* tried were as follows:
The password given to us in the problem was of the form:
Wannabe_Eves: This door has RSA encryption with exponent 5 and the password is
So we tried:

1. “Wannabe_Eves: This door has RSA encryption with exponent 5 and the password is”
2. “This door has RSA encryption with exponent 5 and the password is”
3. “Wannabe_Eves: This door has RSA encryption with exponent 5 and the password is-”
4. “Wannabe_Eves: This door has RSA encryption with exponent 5 and the password is:”
5. “Wannabe_Eves: This door has RSA encryption with exponent 5 and the password is ”
6. “WANNABE_EVES: THIS DOOR HAS RSA ENCRYPTION WITH EXPONENT 5 AND THE PASSWORD IS”
7. “wannabe_eves: this door has rsa encryption with exponent 5 and the password is”

8. “Wannabe_Eves: This door has RSA encryption with exponent 5 and the password is ”

xii. Before we got the correct padding *p* which is **8 (with two spaces at the end),** we tried various combinations of the first 7 strings by varying the way Wannabe_Eves is written and toggling the case.

xiii. The root found by the modified Coppersmith's Algorithm was: **11010100110100001110110011010110100100001001010010010 110100101101001010011001 10**

Now, we picked 8 bit chunks at a time and looked at the corresponding ASCII value and the decrypted password found out was:

jhvkHJKKJf

References:

1. [Lattice Reduction on Low-Exponent RSA]()

Team Details:

- Ashish Pal (18111010)
- Darshit Vakil (18111013)
- Mayank Rawat (18111040)