Cryptanalysis of Book-Caesar-RSA Cipher

Samanvitha Basole and Shubham Pachpute

CS 265 Cryptography and Computer Security

San Jose State University

March 24, 2018

1 Problem Statement

From the Mystery Twister website, we attempt to solve the BCR code challenge. In this challenge, we help Alice and Bob find the treasure. Given the book text:
$'SIERRA-ZERO-JULIET-SIX-YANKEE-ONE-ROMEO-PAPPAEIGHT-KILO-FIVE-UNIFORM-XRAY-XXX-BRAVO-VICTORTWO-FOUR-TANGO-MIKE-OSCAR-HOTEL-DELTA-QUEBECKFOXTROT-ALPHA-YYY-LIMA-INDIA-THREE-WHISKEYNOVEMBER-ECHO-CHARLIE-GOLF-ZULU'$,
our task is to find the two words marked as XXX and YYY then use Book, Caesar, RSA ciphers to get the final output.

2 Approach

In this paper, we solve the BCR cipher consisting of a three-stage cascade: Altered book, Caesar, and RSA. Each stage uses the output of the previous one as input as shown in Figure 1. We discuss the high-level algorithm design of Book, Caesar, and RSA ciphers, then we

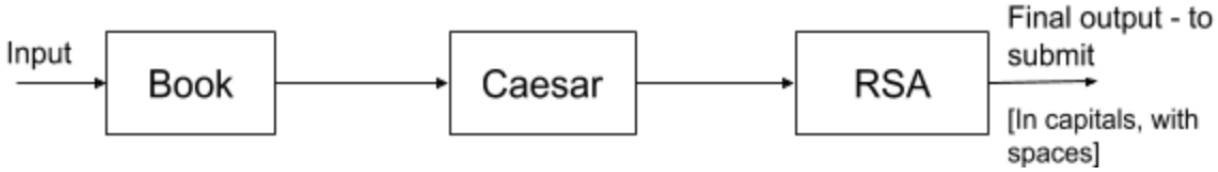


Figure 1: General Architecture

present the algorithm pseudocode and part of the implementation. Next, we explain our decryption process and analysis achieved through computational experiments. Finally, we provide screenshots of our solution and submission.

3 Algorithms Discussion

3.1 Book Cipher

Book cipher uses letters of subsequent words in a text to encode messages. Essentially, these words form the key. Book cipher is a cipher where the algorithm uses a book or a piece of text for encryption. In this algorithm we replace the words in the plaintext with the location of the words in the book or text used for encryption. It is necessary to have the word in the plaintext present in the book to be used or else, sometimes it is also replaced by using letters in the words. The main idea of the book cipher is the key used for encryption. Both the parties exchanging the information should agree to use a book or a publication available to both of them which will be used as the key. The attacker intercepting the message should be able to guess the key to be able to decrypt the message. Widely used books are the Dictionary and the Bible.

3.2 Caesar Cipher

Caesar cipher is a simple substitution cipher in which the alphabet is shifted a certain number of characters based on the key. It is a symmetric key algorithm, that is, the same key is used for encryption and decryption [1].
We show the following example to demonstrate encryption and decryption using Caesar cipher: Given a Caesar cipher with a left shift of 3, we first convert letters to numbers:

Letters:	A	B	C	D	E	F	...	X	Y	Z
Numbers:	0	1	2	3	4	5	...	23	24	25

To encrypt the plaintext 'ATTACK AT DAWN', we

- Transfom each letter in the plaintext into a number using the above scheme
- Apply the rule: (number + key) mod 26 to each transformed number
- Tranform each number back to its corresponding letter using the same scheme above

The result is the following ciphertext XQQXZH XQ AXTK.
Decryption is done in a similar way except the following rule is applied in the second step: (number - key) mod 26.

3.3 RSA

RSA is an algorithm that provides digital signature as well as encryption. RSA is based on factoring, that is, it is easy to factor two numbers but it is computationally NP-C to find two exact factors for a given number. This is a public key cryptosystem, and thus, the encryption key and the decryption key are different. [3].
The RSA algorithm is as follows [2]:

- Select p, q two large prime numbers
- Calculate $N = pq$
- Select integer e relatively prime to $(p-1)(q-1)$
- Calculate d as the multiplicative inverse of e modulo $(p-1)(q-1)$

- Public key (n, e)
- Private key d
- Consider plaintext M
- Convert plaintext to ciphertext by $C = M^e mod\ N$
- Convert ciphertext to plaintext by $M = C^d mod\ N$

4 Algorithm Design

4.1 Book

The main logic of the book cipher algorithm is listed below in Python:

```
finalText1 = []
for elements in decodedText1:
    # print len(decodedText1[j])
    # print len(elements)
    if len(elements) > 1:
        elements = elements[0]
        finalText1.append(elements)
        # print elements
        continue
    if len(elements) == 1:
        finalText1.append(elements)
finalNum1 = []
for elements in num:
    elements = finalText1[int(elements) - 1]
    finalNum1.append(elements)
print "\nFinal encrypt Num1: "
print finalNum1
```

4.2 Caesar

The hex input consists of letters and numbers while the key consists of numbers as offset. The below code demonstrates the main logic of this cipher:

```
output_str = ""
for eachChar in hex_string:
    try:
      # if eachChar is a number
        digit = int(eachChar)
        output_str += str((digit - int(YY)) % 10)
    except:
      # if eachChar is a letter
        output_str += letters[(letters.index(eachChar) + int(XX)) % 26]
```

4.3 RSA

RSA works on the principle of factoring. Below, we have shown the code for modular inverse which is an important concept in RSA.

```
def modularInverse(a, m):
    g, x, y = modInv_part2(a, m)
    if g != 1:
        raise Exception('No modular inverse')
    return x % m

def modInv_part2(a, b):
    if a == 0:
        return (b, 0, 1)
    g, y, x = modInv_part2(b % a, a)
    return (g, x - (b // a) * y, y)
```

5 Decryption

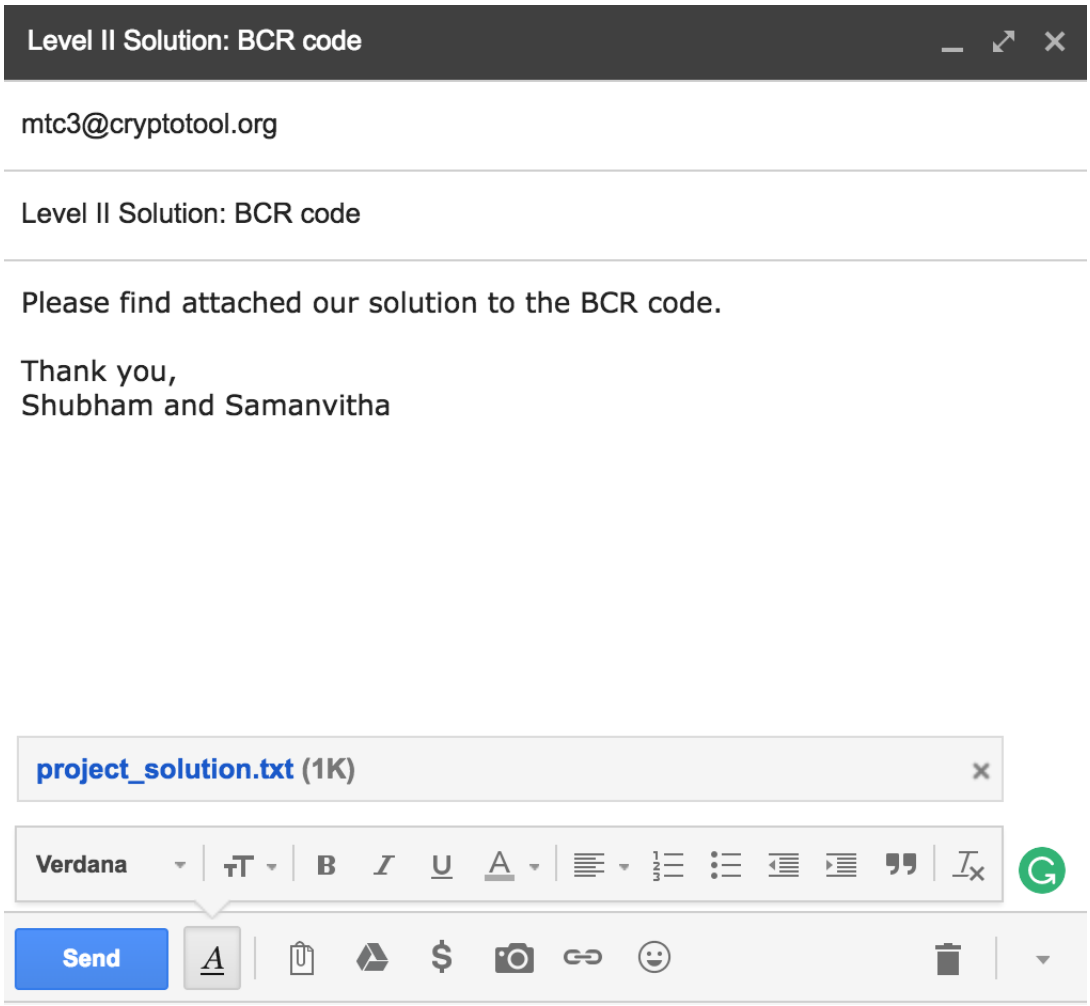
Figure 2 displays the results of the program consisting of the entire pipeline as illustrated in the architecture image in Figure 1.

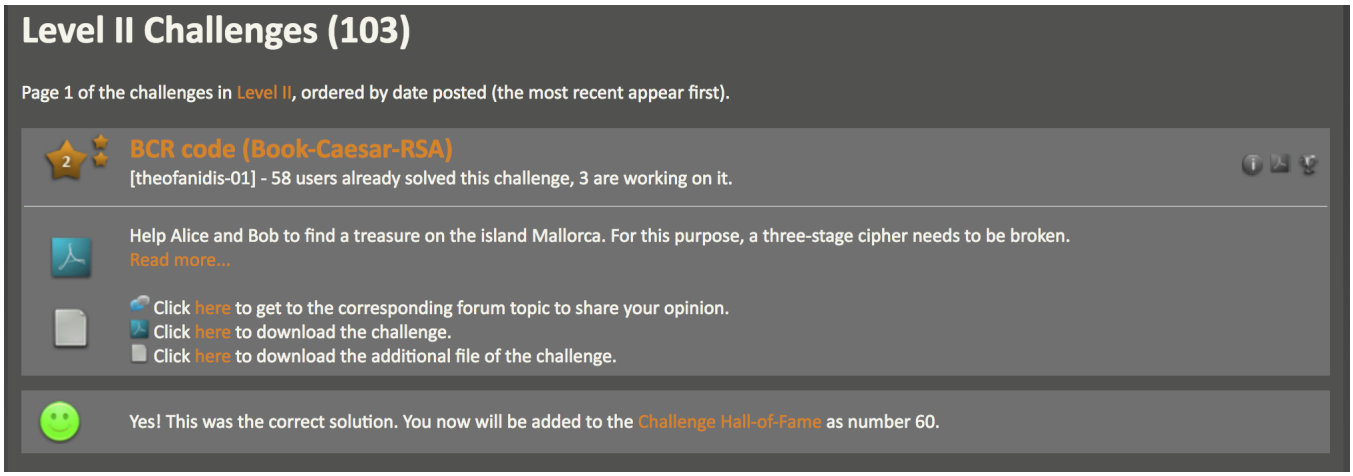
```
################ Outputs of Book Cipher ################

Final encrypt Num1:
0TQ6QP59RT710Q56S666P347T6R4R7UUQTRT28U1Q19SR196736P84210UPT7617P8P2476944P316P8R7212928Q3Q88UPT2879SQ653R16T1Q8S5841T54091T

Final encrypt Num2:
0TQ6QP57RT910Q56S666P349T6R4R9UUQTRT28U1Q17SR176936P84210UPT9619P8P2496744P316P8R9212728Q3Q88UPT2897SQ653R16T1Q8S5841T54071T


################ Outputs of Caesar Cipher ################

Outputs from input 1:
Output 1: 3EB9BA82CE043B89D999A670E9C7C0FFBECE51F4B42DC429069A17543FAE0940A1A5709277A649A1C0545251B6B11FAE5102DB986C49E4B1D8174E87324E
Output 2: 7EB3BA26CE487B23D333A014E3C1C4FFBECE95F8B86DC863403A51987FAE4384A5A9143611A083A5C4989695B0B55FAE9546DB320C83E8B5D2518E21768E

Outputs from input 2:
Output 1: 3EB9BA80CE243B89D999A672E9C7C2FFBECE51F4B40DC409269A17543FAE2942A1A5729077A649A1C2545051B6B11FAE5120DB986C49E4B1D8174E87304E
Output 2: 7EB3BA24CE687B23D333A016E3C1C6FFBECE95F8B84DC843603A51987FAE6386A5A9163411A083A5C6989495B0B55FAE9564DB320C83E8B5D2518E21748E

################ Outputs of RSA Cipher ################

The plaintext Message encrypted was:
F R O M D G O N O R T H 3 7 W E S T 2 3 D I G 5
The treasure is found :)

Process finished with exit code 0
```

Figure 2: Program Output

6 Solution Submission

Below is the proof of submission to the Mystery Challenge website





References

[1] Bruce Busta. Encryption in theory and practice. *The CPA Journal*, 72(11):42, 2002.

[2] S. A. Jaju and S. S. Chowhan. A modified rsa algorithm to enhance security for digital signature. In *2015 International Conference and Workshop on Computing and Communication (IEMCON)*, pages 1–5, Oct 2015.

[3] Karthik, Chinnasamy, and Deepalakshmi. Hybrid cryptographic technique using otp:rsa. In *2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)*, pages 1–4, March 2017.