RSA, RC4, and Caesar Cipher

Hands on Coding Project for Cyber Security

Akash Dansinghani[#1], Tri Nguyen[*2]

[#]*Department of Computer Science, Georgia State University*
Atlanta Georgia

[1] adansinghani1@student.gsu.edu

Abstract— **This document gives formatting instructions for authors preparing papers for publication in the Proceedings of an IEEE conference. Our document is about comparing and contrasting three different algorithms that are implemented in cybersecurity. We will analyse them, write code for each algorithm, and talk about the efficiency of the programs and which are more favourable compared to each other.**

Keywords— **crypto, algorithm, RC4, RSA, Cipher, public key, private key**

I. Introduction

In cybersecurity, we are interested in protecting information. However, there are unforeseen events that may occur and there is no way to stop it. This is why we are analysing certain algorithms that are covered in our cybersecurity class to see how secure they are, how efficient they are, and why they became obsolete or why they are still used today.

II. RSA

(Rivest- Shamir - Adelman) crypto algorithm is used to encrypt and decrypt messages. What is special about this crypto algorithm is that it is an asymmetric cryptographic algorithm which means that to run this crypto algorithm, it will need two different keys. One key will be public and the other key will be private. The public key can be given to anyone where it must be kept privately. Before the keys can be generated, however, a public key can be an exception where it can be given, we must pick two prime numbers and we can call it "p" and "q" .These numbers must be kept as secret because these two numbers will determine the keys, especially, private key. These two numbers can be two different prime numbers and we can multiply them to get the modulus or we call these modulus "n" for the public key and public key. However, since the public key can be shared, our algorithms take an input for the public key rather than generating it which could save us a little process time. The next thing we need for the RSA algorithm to work is totient which can be generated from doing (p-1)(q-1). With these informations, we should be able to find out the public key and the private key. If we want to generate a public key, we can choose any number for e but 1<e < totient and "e" must not have no factor other than 1. For private key which can also be called "d", we can use the formula d=(1+tonient)/e. From here, you should have every variable you need to do encryption and decryption. For encryption, we can do message^{E}(mod n); And for decryption we can do encryptResult^{D}(mod n).

To be able to measure the time for key generation, we have set up a timer system where we started the timer at the beginning and stopped after we find the multiplicative inverse of 'e' and 'r' process. This process; however; could be factor by the type speed of the person since this algorithm requires the user to enter inputs such as q,p,and e. To measure the encryption and decryption time, we have set up a timer where we started after the input of 'q', 'p',and 'e' and we ended the timer after the result was shown. The timer for encryption and decryption could also be affected by typing speed as well as the size of the input. Obviously the longer the word you want to encrypt, the longer the user will need to type along with the loop of through each character as well. Therefore, to keep the time efficient and

ACCURATE, WE HAVE USED THE SAME NUMBER FOR 'Q', 'P',AND 'E' THROUGHOUT ALL THE TRIALS. FOR THE ALL THE TRIAL, WE WILL USE THE 3 FOR 'P', 11 FOR 'Q' AND 7 FOR 'E', THEREFORE THE KEY GENERATION TIME WILL MOST LIKELY BE SIMILAR THROUGHOUT ALL RUNS; HOWEVER, WE WILL BE USING DIFFERENT INPUT SIZES TO MEASURE THE TIME FOR ENCRYPTION AND DECRYPTION. THE EACH TRIAL, WE WILL BE RUNNING IT TWICE BUT TO TEST DIFFERENT INPUT SIZES, THE FIRST RUN, WE WILL BE USING 5 LETTERS INPUT WHICH IS 'HELLO' TO ENCRYPT AND THE TIME TO ENCRYPT 5 LETTERS 'HELLO' IS ROUGHLY 9.5367431640625E-07 SECONDS AND THE KEY GENERATION IS 2.4181711673736572 SECONDS INCLUDING MANUALLY INPUT OF 3,11 AND 7. THE SECOND RUN FOR THE ENCRYPTION, WE WILL BE USING BIGGER INPUT WHICH IS 'GEORGIA' WHICH HAS 7 LETTERS. THE TIME FOR 7 LETTERS IS 2.1457672119140625E-06 SECONDS AND IT IS AS EXPECTED, WHICH TAKES A VERY LITTLE BIT LONGER. TO DECRYPT, WE WILL ALSO BE RUNNING THE DECRYPTION OF EACH INPUT TWICE TO MAKE SURE THE VALUE IS CORRECT, THE NUMBER FORM OF 'HELLO' WHICH IS [2, 14, 12, 12, 27], IT TAKES ROUGHLY 9.5367431640625E-07 SECONDS. TO DECRYPT A NUMBER FORM OF A LARGER INPUT OF 7 LETTERS 'GEORGIA' WHICH IS [28, 14, 27, 6, 28, 15, 1], IT TAKES ROUGHLY 1.9073486328125E-06 SECONDS. BELOW ARE THE SCREENSHOTS OF THE KEY GENERATION TIME, ENCRYPTION TIMES AND DECRYPTION TIMES FOR ALL RUNS.

A. Runtimes

Here are the runtimes for the RSA Algorithm

```
Tris-MacBook-Air:desktop tringuyen$ python3 RSA.py
PLEASE ENTER THE 'p', 'q', 'e'  VALUES:
Enter a prime number for p: 3
Enter a prime number for q: 11
Enter a prime number for e: 7
Enter message you would like to encrypt or Enter the number values of decrypt separate by ',':
hello
Your message is: hello
Type 'encrypt' for encryption and 'decrypt' for decrytion.
encrypt
Your encrypted message is: [2, 14, 12, 12, 27]
encryption time is:  9.53674316406625e-07 seconds
the generation time is:  2.1177599430084233 seconds
Tris-MacBook-Air:desktop tringuyen$ python3 RSA.py
PLEASE ENTER THE 'p', 'q', 'e'  VALUES:
Enter a prime number for p: 3
Enter a prime number for q: 11
Enter a prime number for e: 7
Enter message you would like to encrypt or Enter the number values of decrypt separate by ',':
hello
Your message is: hello
Type 'encrypt' for encryption and 'decrypt' for decrytion.
encrypt
Your encrypted message is: [2, 14, 12, 12, 27]
encryption time is:  9.53674316406625e-07 seconds
the generation time is:  2.4181711673736572 seconds
Tris-MacBook-Air:desktop tringuyen$
```

B.

```
Tris-MacBook-Air:desktop tringuyen$ python3 RSA.py
PLEASE ENTER THE 'p', 'q', 'e'  VALUES:
Enter a prime number for p: 3
Enter a prime number for q: 11
Enter a prime number for e: 7
Enter message you would like to encrypt or Enter the number values of decrypt separate by ',':
georgia
Your message is: georgia
Type 'encrypt' for encryption and 'decrypt' for decrytion.
encrypt
Your encrypted message is: [28, 14, 27, 6, 28, 15, 1]
encryption time is:  1.9073486328125e-06 seconds
the generation time is:  2.992979049682617 seconds
Tris-MacBook-Air:desktop tringuyen$ python3 RSA.py
PLEASE ENTER THE 'p', 'q', 'e'  VALUES:
Enter a prime number for p: 3
Enter a prime number for q: 11
Enter a prime number for e: 7
Enter message you would like to encrypt or Enter the number values of decrypt separate by ',':
georgia
Your message is: georgia
Type 'encrypt' for encryption and 'decrypt' for decrytion.
encrypt
Your encrypted message is: [28, 14, 27, 6, 28, 15, 1]
encryption time is:  2.1457672119140625e-06 seconds
the generation time is:  2.7595572471618652 seconds
Tris-MacBook-Air:desktop tringuyen$
```

C.

```
Tris-MacBook-Air:desktop tringuyen$ python3 RSA.py
PLEASE ENTER THE 'p', 'q', 'e'  VALUES:
Enter a prime number for p: 3
Enter a prime number for q: 11
Enter a prime number for e: 7
Enter message you would like to encrypt or Enter the number values of decrypt separate by ',':
2, 14, 12, 12, 27
Your message is: 2, 14, 12, 12, 27
Type 'encrypt' for encryption and 'decrypt' for decrytion.
decrypt
Your decrypted message is: HELLO
decryption time is:  9.5367431640625e-07 seconds
the generation time is:  2.2436201572418213 seconds
Tris-MacBook-Air:desktop tringuyen$ python3 RSA.py
PLEASE ENTER THE 'p', 'q', 'e'  VALUES:
Enter a prime number for p: 3
Enter a prime number for q: 11
Enter a prime number for e: 7
Enter message you would like to encrypt or Enter the number values of decrypt separate by ',':
2, 14, 12, 12, 27
Your message is: 2, 14, 12, 12, 27
Type 'encrypt' for encryption and 'decrypt' for decrytion.
decrypt
Your decrypted message is: HELLO
decryption time is:  9.5367431640625e-07 seconds
the generation time is:  2.4835031032562256 seconds
```

```
Tris-MacBook-Air:desktop tringuyen$ python3 RSA.py
PLEASE ENTER THE 'p', 'q', 'e'  VALUES:
Enter a prime number for p: 3
Enter a prime number for q: 11
Enter a prime number for e: 7
Enter message you would like to encrypt or Enter the number values of decrypt separate by ',':
28, 14, 27, 6, 28, 15, 1
Your message is: 28, 14, 27, 6, 28, 15, 1
Type 'encrypt' for encryption and 'decrypt' for decrytion.
decrypt
Your decrypted message is: GEORGIA
decryption time is:  1.1920928955078125e-06 seconds
the generation time is:  2.670337677001953 seconds
Tris-MacBook-Air:desktop tringuyen$ python3 RSA.py
PLEASE ENTER THE 'p', 'q', 'e'  VALUES:
Enter a prime number for p: 3
Enter a prime number for q: 11
Enter a prime number for e: 7
Enter message you would like to encrypt or Enter the number values of decrypt separate by ',':
28, 14, 27, 6, 28, 15, 1
Your message is: 28, 14, 27, 6, 28, 15, 1
Type 'encrypt' for encryption and 'decrypt' for decrytion.
decrypt
Your decrypted message is: GEORGIA
decryption time is:  1.9073486328125e-06 seconds
the generation time is:  2.776524066925049 seconds
```

D. Figures and Tables

Figures and tables must be centered in the column. Large figures and tables may span across both columns. Any table or figure that takes up more than 1 column width must be positioned either at the top or at the bottom of the page.

III. RC4

(Rivest- Shamir - Adelman) crypto algorithm is used to encrypt and decrypt messages. What is special about this crypto algorithm is that it is an asymmetric cryptographic algorithm which means that to run this crypto algorithm, it will need two different keys. One key will be public and the other key will be private. The public key can be given to anyone where it must be kept privately. Before the keys can be generated, however, a public key can be an exception where it can be given, we must pick two prime numbers and we can call it "p" and "q" .These numbers must be kept as secret because these two numbers will determine the keys, especially, private key. These two numbers can be two different prime numbers and we can multiply them to get the modulus or we call these modulus "n" for the public key and public key. However, since the public key can be shared, our algorithms take an input for the public key rather than generating it which could save us a little process time. The next thing we need for the RSA algorithm to work is totient which can be generated from doing (p-1)(q-1). With these informations, we should be able to find out the public key and the private key. If we want to generate a public key, we can choose any number for e but 1<e < totient and "e" must not have no factor other than 1. For private key which can also be called "d", we can use the formula d=(1+tonient)/e. From here, you should have every variable you need to do encryption and decryption. For encryption, we can do message^{E}(mod n); And for decryption we can do encryptResult^{D}(mod n).

III. To be able to measure the time for key generation, we have set up a timer system where we started the timer at the beginning and stopped after we find the multiplicative inverse of 'e' and 'r' process. This process; however; could be factor by the type speed of the person since this

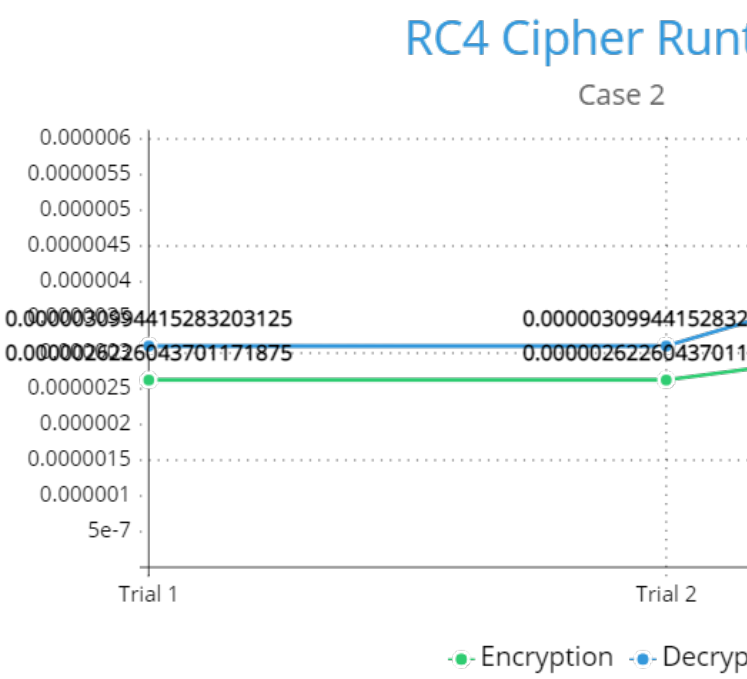
ALGORITHM REQUIRES THE USER TO ENTER INPUTS SUCH AS Q,P,AND E. TO MEASURE THE ENCRYPTION AND DECRYPTION TIME, WE HAVE SET UP A TIMER WHERE WE STARTED AFTER THE INPUT OF 'Q', 'P',AND 'E' AND WE ENDED THE TIMER AFTER THE RESULT WAS SHOWN. THE TIMER FOR ENCRYPTION AND DECRYPTION COULD ALSO BE AFFECTED BY TYPING SPEED AS WELL AS THE SIZE OF THE INPUT. OBVIOUSLY THE LONGER THE WORD YOU WANT TO ENCRYPT, THE LONGER THE USER WILL NEED TO TYPE ALONG WITH THE LOOP OF THROUGH EACH CHARACTER AS WELL. THEREFORE, TO KEEP THE TIME EFFICIENT AND ACCURATE, WE HAVE USED THE SAME NUMBER FOR 'Q', 'P',AND 'E' THROUGHOUT ALL THE TRIALS. FOR THE ALL THE TRIAL, WE WILL USE THE 3 FOR 'P', 11 FOR 'Q' AND 7 FOR 'E', THEREFORE THE KEY GENERATION TIME WILL MOST LIKELY BE SIMILAR THROUGHOUT ALL RUNS; HOWEVER, WE WILL BE USING DIFFERENT INPUT SIZES TO MEASURE THE TIME FOR ENCRYPTION AND DECRYPTION. THE EACH TRIAL, WE WILL BE RUNNING IT TWICE BUT TO TEST DIFFERENT INPUT SIZES, THE FIRST RUN, WE WILL BE USING 5 LETTERS INPUT WHICH IS 'HELLO' TO ENCRYPT AND THE TIME TO ENCRYPT 5 LETTERS 'HELLO' IS ROUGHLY 9.5367431640625E-07 SECONDS AND THE KEY GENERATION IS 2.4181711673736572 SECONDS INCLUDING MANUALLY INPUT OF 3,11 AND 7. THE SECOND RUN FOR THE ENCRYPTION, WE WILL BE USING BIGGER INPUT WHICH IS 'GEORGIA' WHICH HAS 7 LETTERS. THE TIME FOR 7 LETTERS IS 2.1457672119140625E-06 SECONDS AND IT IS AS EXPECTED, WHICH TAKES A VERY LITTLE BIT LONGER. TO DECRYPT, WE WILL ALSO BE RUNNING THE DECRYPTION OF EACH INPUT TWICE TO MAKE SURE THE VALUE IS CORRECT, THE NUMBER FORM OF 'HELLO' WHICH IS [2, 14, 12, 12, 27], IT TAKES ROUGHLY 9.5367431640625E-07 SECONDS. TO DECRYPT A NUMBER FORM OF A LARGER INPUT OF 7 LETTERS 'GEORGIA' WHICH IS [28, 14, 27, 6, 28, 15, 1], IT TAKES ROUGHLY 1.9073486328125E-06 SECONDS. BELOW ARE THE SCREENSHOTS OF THE KEY GENERATION TIME, ENCRYPTION TIMES AND DECRYPTION TIMES FOR ALL RUNS.

E. Runtimes

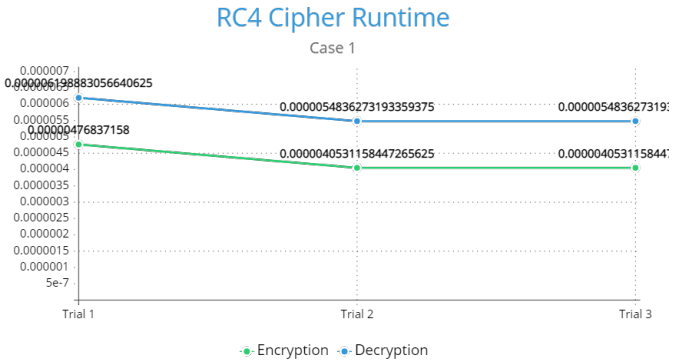
Here are the runtimes for the RC4 Algorithm

```
Input your encryption key : hi
Input your plaintext : hello
This is your ciphertext : kdmkk
encryption time is:  4.76837158203125e-06 seconds
key generation time is:  4.291534423828125e-06 seconds

Welcome to my RC4 Algorithm
1. Encrypt
2. Decrypt
Please type in 1 or 2 if you want to encrypt or decrypt : 2

Input your encryption key : hi
Input your ciphertext : kdmkk
This is your plaintext : hello
decryption time is:  6.198883056640625e-06 seconds
key generation time is:  4.291534423828125e-06 seconds
```

```
Input your encryption key : yoloswagalicious
Input your plaintext : chupappimunyanyobby
This is your ciphertext : okipwaprvirsgqmgbhg
encryption time is:  2.6226043701171875e-06 seconds
key generation time is:  2.6226043701171875e-06 seconds

Welcome to my RC4 Algorithm
1. Encrypt
2. Decrypt
Please type in 1 or 2 if you want to encrypt or decrypt :

Input your encryption key : yoloswagalicious
Input your ciphertext : okipwaprvirsgqmgbhg
This is your plaintext : chupappimunyanyobby
decryption time is:  3.0994415283203125e-06 seconds
key generation time is:  2.6226043701171875e-06 seconds
```



F. *Figures and Tables*



IV. Caesar Cipher

Caesar cipher uses a substitution method which uses keys to shift the alphabetical letters index by the value of the key. Say the key = 3, then all 26 letters in the alphabet will be shifted 3 indexes.

plain: 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

cipher: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

It is said this is what Julius Caesar used himself to communicate with his fellow comrades. We can assign each character a numerical value where 0 = A, 1 = B, and so on. Below is a table where each character is assigned a value:

Even though there are 26 letters in the alphabet, we can see that max value is 25 because our first index value is 0. To encrypt text we use the following line of code below:

And to decrypt it we simply subtract the key which becomes:

When using the cipher, you are asked to give a plain text string which would be a message that the user inputs. In our example for checking the runtime, we used the string hello and the key 3. Our encrypted text (which would be the ciphertext) becomes khoos. To decrypt the text, we can shift the key back a value of three and we would get the plaintext again.

We had some inconsistencies with the encrypt and decrypt times but it stayed in a range (+- 2.38418574e^-7seconds off). The encrypt time was slightly faster than the decrypt time which makes sense because it takes in ambiguous letters and has to convert it and run it through to become plaintext.

Plaintext = mi nombre es jefe

Key = 6

Ciphertext = so tushxk ky pklk

As we can see with the results below, the first run and the last run had equal runtimes while the second run did not have an equal runtime. Like mentioned before, the runtimes may vary. In this case the runtime for run two has a difference of (4.76837158e^-7). We can assume this as an outlier.

We had some inconsistencies with the encrypt and decrypt times but it stayed in a range (+- 2.38418574e^-7seconds off). The encrypt time was slightly faster than the decrypt time which makes sense because it takes in ambiguous letters and has to convert it and run it through to become plaintext.

Plaintext = mi nombre es jefe

Key = 6

Ciphertext = so tushxk ky pklk

As we can see with the results below, the first run and the last run had equal runtimes while the second run did not have an equal runtime. Like mentioned before, the runtimes may vary. In this case the runtime for run two has a difference of (4.76837158e^-7). We can assume this as an outlier.

G. Runtimes

Here are the runtimes for the RSA Algorithm

```
Enter message you would like to encrypt or Enter the number value
ecrypt separate by ',':
Hello
Your message is: Hello
Enter key:3
Type 'encrypt' for encryption and 'decrypt' for decrytion.
encrypt
Your encrypted message is: Khoor
encryption time is:  1.1920928955078125e-06 seconds
```

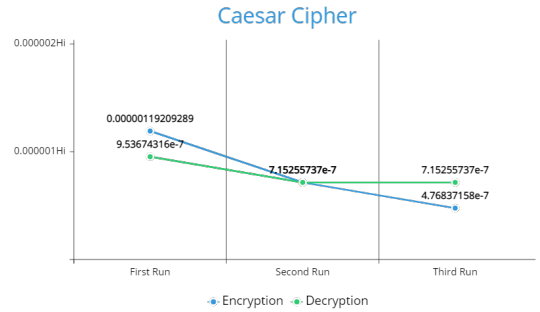
Encrypt Time: 1.1920928955078125e-06 seconds

```
Enter message you would like to encrypt or Enter the number value
ecrypt separate by ',':
Khoor
Your message is: Khoor
Enter key:3
Type 'encrypt' for encryption and 'decrypt' for decrytion.
decrypt
Your decrypted message is: Hello
encryption time is:  9.5367431640625e-07 seconds
```

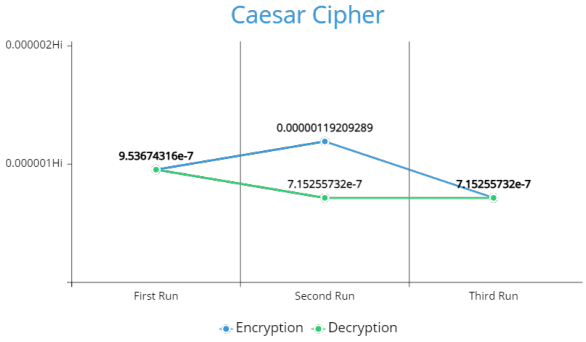
Decrypt Time: 9.5367431640625e-07 seconds

H.

I. Figures and Tables



J.



K. Conclusions

The Advantages of RSA is that it increases security because keys never have to be transferred and it provides digital signatures that cannot be duplicated and compared to other symmetric key algorithms, RSA is much stronger. The disadvantages are its speed in terms of encryption and decryption, compared to other symmetric key algorithms such as AES, and RC4, symmetric key algorithm has much faster key generation time where RSA is slower. Also, since RSA uses the two part key, it is vulnerable to attacks if poorly implemented. RC4 algorithms advantages are its simplicity where it is easy to use and implement. Its speed is much faster than other ciphers such as the RSA and DES algorithm. RC4's disadvantage is that it is not a secure algorithm because it is possible to obtain information about the key stream. With Caesar cipher, its advantage is that it is really easy method to created or implement and it faster than RSA; However, its disadvantage is that it is extremely easy to crack because there is only 26 letters in the alphabets and you technically can guess the key between 0 and 26 to decrypt the plaintext. Over to compare between the three algorithms, I believe RSA would be the most useful of them all since RC4 and Caesar Cipher are extremely vulnerable in terms of security. Even though RSA is the slowest of the three algorithms, it is the most secure of the three, and the goal of encryption and encryption is to secure whatever we want to send. Therefore, RSA is the best algorithm out of the three.

REFERENCES

[1] "RSA (cryptosystem)," Wikipedia, 05-Apr-2021. [Online]. Available: https://en.wikipedia.org/wiki/RSA_(cryptosystem). [Accessed: 15-Apr-2021].

[2] N. T, C. P. Price, and Pric, "What is RSA Algorithm in Cryptography? Key Generation, Encryption, Decryption, Advantages & Disadvantages," Binary Terms, 18-Jan-2021. [Online]. Available: https://binaryterms.com/rsa-algorithm-in-cryptography.html#KeyGeneration. [Accessed: 15-Apr-2021].

[3] How does RC4 works. [Online]. Available: https://paginas.fe.up.pt/~ei10109/ca/rc4.html#:~:text=RC4%20generates%20a%20pseudo%2Drandom,bit%2Dwise%20exclusive%2Dor.&text=The%20permutation%20is%20initialized%20with,%2Dscheduling%20algorithm%20(KSA). [Accessed: 14-Apr-2021].

[4] Fluhrer S., Mantin I., Shamir A. (2001) Weaknesses in the Key Scheduling Algorithm of RC4. In: Vaudenay S., Youssef A.M. (eds) Selected Areas in Cryptography. SAC 2001. Lecture Notes in Computer Science, vol 2259. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45537-X_1