UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION

JUNIOR ACADEMY OF SCIENCES OF UKRAINE UNDER THE AUSPICES OF UNESCO

KAROLINA SUSOL

EVOLUTION OF CRYPTOGRAPHY. SYMMETRIC AND ASYMMETRIC ENCRYPTION

UKRAINE

Introduction and Abstract

Cryptology - is the science about secure communication.
For a long time, the cryptography mainly served diplomatic and military purposes. But nowadays it is cryptography's time to shine! Every single action on the Internet includes encryption. You can't google anything or message anybody without hundreds of crypto operations running at the time.
Cryptosystems are in charge of wireless networking; phone calls; banking systems; messages; authenticity, digital signatures, etc. This restless science will never stop changing! And you have to stay on top all the time in order to use secure and suitable cryptosystems.
This research will definitely help you because we:
1) gradually introduce the reader to the science;
2) classify algorithms and find out their pros and cons;
3) see how cryptography's been evolving;
4) find out why some ciphers have been compromised;
5) analyze the algorithms' base in number theory;
create some tricks to transform sequences.

It is extremely important to provide safe communication and protect people's personal information from intruders. In order to do that, cryptography has to change all the time and improve its methods. And my research has a few algorithms that can be used either in a one-time pad (when the safety is crucial) or in hash-functions (the last three of them).

Conclusion

We classified the famous ciphers and found their weaknesses and strengths. Most of them aren't used nowadays, but they tell a reader a lot about cryptology throughout History. Also, we proved the maths base of some attacks and algorithms using Chinese Remainder Theorem, The Fermat–Euler theorem and the Euclidean algorithm.
In the last chapter, I described one algorithm that can be used for a one-time pad (OTP) and the other three that can be used for both the cipher and hash functions.
The great advantage of this addition to OTP is a low risk of decrypting the messages even in case of stealing the pad. Of course, such conspiration is not for daily purposes. And for national secrets or diplomacy, it would be perfect.
In addition, the last three algorithms would be suitable for using in hash-functions.

Having studied it all I can make a comparison between symmetric and asymmetric encryptions.
Symmetric:
fast;
easier to implement (due to simple operations);
more studied (because it has existed since ancient times).
Asymmetric encryption:
unlike symmetric, can easily generate, keep many keys on the net.
It is far more convenient for key exchange
And which is better? Without symmetric encryption operations with big data would take too much time and without asymmetric encryption it would be impossible to exchange keys! As I have already mentioned, most of the time these types of encryption are combined.

Materials and Methods

In the first chapter, no complicated computations were conducted. Just some conclusions according to the number of ways to encrypt a message.
The second one contained proofs to RSA, Hastad's attack, digital signature (because it is still an asymmetric algorithm) and Diffie-Hellman key exchange. Here will b RSA and my own algorithms.
RSA
1) Choose two big primes p and q (keep them secret) and compute their product N
2) Compute Euler's totient function $\varphi(N)=(p-1)(q-1)$
3) Choose an open exponent e. The most commonly chosen value for e is $2^{16}+1$.
4) Compute the secret value
$d=e^{-1}(\text{mod } \varphi(N))$
The pair {e, N} is the public key
{d, N} is the private key
Here is how Alice will encrypt her message using Bob's private key:
m - Alice's message expressed in a number; compute C - ciphertext as follows:
$C=m^{eB}$ mod NB, where {eB, NB} is Bob's public key
Then Bob can decrypt the message this way:
$m=C^{dB}$ mod NB, where {dB, NB}is Bob's private key
Here is why $m=C^{dB}$ mod NB
$C^{dB}=(m^{eB})^{dB}$ mod NB
Since $dB=eB^{-1}$ mod $\varphi(N)$, we can express dB*eB as $\varphi(N)k+1$, where k is a positive integer.
So $C^{dB}=m^{(eB*dB)}=m^{(\varphi(N)k+1)}$ mod N
We know that for any relatively prime positive integers m and N the following equality holds:
$a^{\varphi(N)}=1(\text{mod } N)$ where a and N are relatively prime and $\varphi(N)$ - Euler's totient function (Euler's theorem)
Eventually,
$C^{dB}=m^{(eB*dB)}=m^{(\varphi(N)*k+1)}=m \text{ mod } N$
Often the Carmichael function is used instead of Euler's totient function.

The first algorithm:
We partition the sequence so that it is increasing while each term has as few digits as possible.
32492340984309137383648 → 3 24 92 340 984 3091 3738 3648
And now we have to add up the digits in each term modulo b (in example 10):
A= 36171311
This algorithm could be used in a one-time pad. But it would be unsuitable for hashes since it is neither collision-resistant nor one-way. We can think of many sequences that result in A. Here are some of them:
3 15 56 223 362 1408 9002 12017
3 60 92 836 1424 7727 8346 9183
Optimization:
Since the first number never changes, Alice and Bob could determine which digit will be the starting point.
In order to confuse the attacker, I recommend starting the transformation from the end.

The next one:
Number the digits and replace ai with $(a_i+j)\%b$, where j is the first number to satisfy aj=ai.
3249234098430938345771 → (3+6) (2+5) (4+7) (9+9) (2) (3+12) ...
9 7 1 8 5 5 ...
Note. Imagine that the sequence is located on a circle. Then for each digit, the "j" will exist (in the worst case j will go through the whole circle and then j=i)
This is how this algorithm looks like on c++:

```
int main(int argc, char** argv) {
string a;
cin >> a;
int len=a.length();

for(int i=0; i<len; i++){
int curr=a[i]-'0', j;
if(i<len-1) j=i+1;
else j=0;
while(a[j]!=a[i]){
j++;
if(j==len) j=0;
}
cout<<(j+curr)%10;
}}
```

And here's how it transforms any string.
The third one lies in finding the remainder of the i-th element and the number (j) of the first such a_j that $a_j\%b==i\%b$. And again, imagine the round sequence.
3249234098430943834384231 → (3+30) (2+5) (4+6) (9+7) (2) ...
3 7 0 6 2 ...
This one is quite similar to the previous algorithm.

And the fourth is:
1. Imagine the sequence written infinitely many times not this way:
1231426731469|1231426731469|...
but this:
.....1426731469|1231426731469|12 31426731469|1231426731469|
2. Take a_i last digits, write down their sum modulo b and delete them.
3. Continue till it is the last digit of the sequence.
The sequence in the example will result in:
9 (4+6) (7+3+1) 6 (3+1+4+2) (1+2) (7+3+1+4+6+9) (1+2+3+1+4+2+6) (4+6+9) (1) (2+6+7+3) (9+1+2+3+1+4) (1+4+2+6+7+3+1+4+6)
9 0 1 6 0 3 0 9 9 1 8 0 4

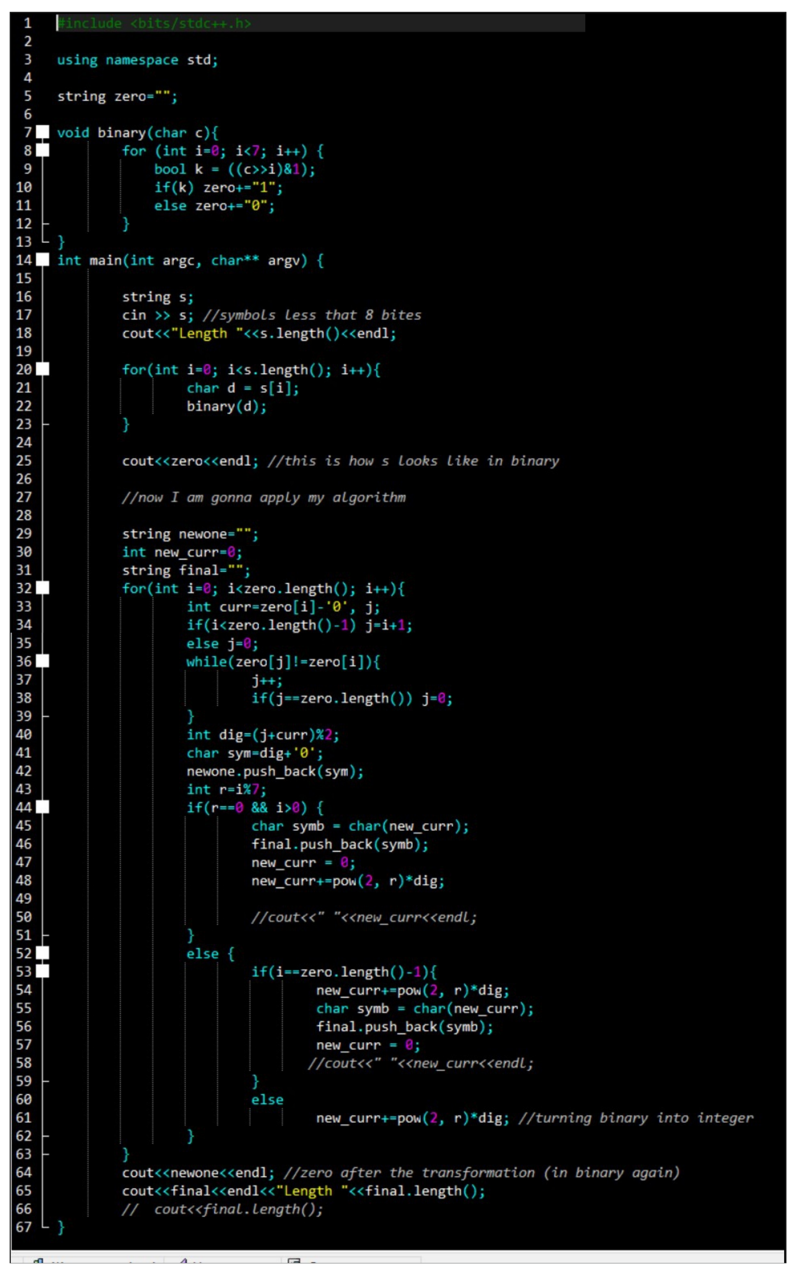

Photo made by Karolina Susol on the 28th of January 2021.
C++ code to my second algorithm