Project 1 documentation

Cryptography course at FIT BUT

Martin Krajňák, xkrajn02

Introduction

The goal of this work is the explanation of the steps which lead to the revelation of the secret. Initially, the work describes revelation of the first part of the encryption algorithm. The next part is dedicated to manual solution and total decryption of the secret used to encrypt files followed by the part explaining the solution achieved with SAT solver.

Known plaintext attack

Since both plain-text (*bis.txt*) and cipher-test (*bis.enc.txt*) are available, then application of the XOR logical function reveals 512 B of keystream. Furthermore, by applying the keysteam to the encrypted file super_cipher.py.enc, a step function used for encryption and SUB array are revealed. A function able to revert the step function needs to be implemented to decrypt files and reveal the secret.

Manual solution

Inspection of the revealed part of the super_cipher.py is showing two substeps. The first step includes simple addition of 1 and other bit operations which are shifting bits. The second step is again using bitwise shift operator to calculate an index to re-assign a new value from the revealed SUB array (substitution) for every bit in the keystream. Therefore a reverse function must implement steps that are inverse to these steps and implement them in reverse order. To reveal the original string used for keystream init we have to apply the reverse fuction 128 times.

The SUB array contains 7 items in total so we have 7 posibilities how to address the items in the array as described in the Table 1. The reverse process has to be done for every bit, so after the value of the first bit is determined we are left with four possibilities as displayed in the table for both 0 and 1. At this state it is imposible to check which option will be the right one, therefore it's necessary to track all four of them. Once the iteration is done for all bits on all four options, one of those four is the right reversation of the substituion. To find which one, the shift operations from the original step function have to be reversed – the add one and shift operation which is simply done by shifting the bits in the reverse order.

A	B	C	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Table 1: Truth table

After application of the reverse shift to the option, the step function can be called with the result and if it matches the original keystream from the input, the step reversal process was has been successful and the result is returned as a total result.

SAT solution

This solution is implemented via satispy[1] library that is using a Minitsat[2] solver binary. The work with the library is for our use case is to create a *Variable* object for each bit of the keystream. The variable object contains two properties: *name* and *reversed*. Those variables are used to simulate boolean values.

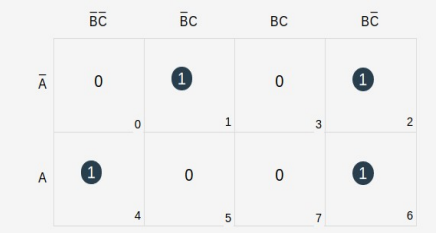

Figure 1: Karnaugh Map

To get a solution, it is required to provide a formula for each bit of the keystream. Therefore a boolean expression in a DNF[3] need to be found for our problem. To achieve this, the Table 1 is converted to the Karnaugh Map displayed in Figure 1. From there there, the formula for True bits is *(-A * -B * C) + (B * -C) + (A * -C)* and for Flase bits *(-A * -B * -C) + (B * C) + (A * C)*.

The library then converts the composed satispy object to Minisat solver binary which tries to provide a solution. If the solution has been found, the result object has two boolean properties *error* and *success*, so we can evaluate the solution. If the solver has successfully found the solution, the object property *varmap* contains the evalution for each bit (*True* or *False*). From those values we are able to reconstruct our solution bit after bit and thus step operation has been reversed and the key has been retrieved.

Conclusion

The goal of this work was to find the initial secret which was used to encrypt the files with initially unknown algorithm. The secret *KRY{xkrajn02-8dd19e02c8a4e45}* has been reveiled with both manual and SAT solver implementation. The manual solution implementation was more time consuming but it is faster than SAT solver.

1 https://github.com/netom/satispy
2 http://minisat.se/
3 https://en.wikipedia.org/wiki/Disjunctive_normal_form