An new hybrid cryptosystem based on the satisfiability problem

Sadek BOUROUBI * — Louiza REZKALLAH **

* USTHB, Faculty of Mathematics, LAID3 Laboratory,
BP 32 16111 El Alia, Bab-Ezzouar, Algiers Algeria
bouroubis@yahoo.fr

** University of Blida, Faculty of Sciences, LAID3 Laboratory,
B.P 270, Route de Soumâa, Blida, Algeria
rlouiza@yahoo.fr

ABSTRACT. In this paper we propose a simple hybrid cryptosystem whose security is based upon the satisfiability problem well known NP-complete. The execution of this cryptosystem is illustrated at the end of the paper by a detailed example to understand better how it processes.

1. Introduction

With the development of the mathematical methods which ensure safe electronic communication, more sophisticated techniques emerged which allow to attack codes on increasingly powerful computers. Modern cryptosystems are based on number theory, commutative group theory, or algebraic geometry. However, it is useful to have other cryp-
tosystems whose security is based on other problems, as the satisfiability problem well known NP-complete.

Let us consider a finite set $X$ with $l$ Boolean variables $x_1, x_2, \ldots, x_l$. A logical or Boolean formula $F(x_1, x_2, \ldots, x_l)$ is a logical expression formed by the Boolean variables, using three operators, negation operator denoted by the symbol bar, conjunction operator denoted by $\land$ and the disjunction operator denoted by $\lor$. A literal is either a variable or its negation like $x$ or $\bar{x}$. A clause is a disjunction of literals. A logical formula is known as in conjunctive normal form (CNF) if it is the conjunction of one or more clauses, as the formula $F$ below:

$$F = (x_1 \lor x_5) \land (x_1 \lor \bar{x}_2 \lor x_4) \land (\bar{x}_3 \lor x_6) \land (x_3 \lor x_4 \lor \bar{x}_5 \lor x_6).$$

The decision problem for Boolean formulas is to determine whether such a formula $F$ is true or false, i.e., if it is possible to assign a value true or false to each variable in such way that the formula has the value true. For example, the formula $F$ above is satisfiable because it takes the value 1 for the assignment $x_1 = x_2 = 1$ and $x_3 = x_4 = x_5 = x_6 = 0$.

The satisfiability problem $k$-SAT, is a problem of decision for which one must determine whether there is a truth (true or false) value we can assign to each of the $k$-literal clauses of a given CNF formula such that the entire expression is true. It is well-known that the 2-SAT problem is solvable in polynomial time, whereas the $k$-SAT problem is NP-complete, for $k \geq 3$. SAT solvers are a powerful tool to test the hardness of certain problems and have successfully been used to test hardness assumptions. In algebraic cryptanalysis, equations are constructed that express the output bits of a cipher in terms of its inputs. These equations are then solved and reasoned about with either dedicated equation solvers such as the F5 algorithm [3], or standard SAT solvers. The first SAT-based cryptanalysis was by Massacci et al. [5], experimenting with the Data Encryption Standard (DES) using DPLL-based SAT solvers. In this paper, our interest is quite different, it is focused around the creation of a new cryptosystem based on the difficulty of the satisfiability problem. We present first, the BinSat algorithm proposed by Aspvall, Plass and Tarjan in 1979 [1], solving the 2-SAT, on which our cryptosystem is based. In fact, this last problem serves to the generation keys of the cryptosystem.
2. BinSat algorithm to solve the 2-SAT problem

In this section we present the BinSat algorithm, for the evaluation of formulas having two literals per clause, based on the problem of path searches in graphs which runs in linear time.

2.1. Associated graph to the 2-SAT

Let $F$ be a 2-SAT instance over variables $x_1, x_2, \ldots, x_l$. We construct the directed graph $G_{2-SAT}$ with vertex set \{ $x_1, x_2, \ldots, x_l, \overline{x_1}, \overline{x_2}, \ldots, \overline{x_l}$ \} and for each clause $(x_i \lor x_j)$ correspond two edges in $G_{2-SAT}$, one from $\overline{x_i}$ to $x_j$ and the other from $\overline{x_j}$ to $x_i$ representing the implications $\overline{x_i} \implies x_j$ and $\overline{x_j} \implies x_i$ respectively. Thus, if $F$ has $m$ clauses there are exactly $2m$ edges in $G_{2-SAT}$.

**Example 2.1** The $G_{2-SAT}$ graph associated to the formula

$$F = (\overline{x_1} \lor x_2) \land (\overline{x_2} \lor x_3) \land (\overline{x_1} \lor \overline{x_3})$$

has a vertex set \{ $x_1, x_2, x_3, \overline{x_1}, \overline{x_2}, \overline{x_3}$ \} and,

- for the clause $(\overline{x_1} \lor x_2)$ correspond the edges $x_1 \implies x_2$ and $\overline{x_2} \implies \overline{x_1}$,
- for the clause $(\overline{x_2} \lor x_3)$ correspond the edges $x_2 \implies x_3$ and $\overline{x_3} \implies \overline{x_2}$,
- for the clause $(\overline{x_1} \lor \overline{x_3})$ correspond the edges $x_1 \implies \overline{x_3}$ and $x_3 \implies \overline{x_1}$.

Then, the graph is given by the Figure 1 below:

![Figure 1: The $G_{2-SAT}$ graph](image-url)
Lemma 2.1 [2] A CNF formula is not satisfiable if and only if there is a variable \( x \) such as there is a paths from \( x \) to \( \overline{x} \) and from \( \overline{x} \) to \( x \) in its associated graph \( G_{2-SAT} \).

The question is then to check if there is a variable \( x_i \) for witch there is a path from \( x_i \) to \( \overline{x}_i \) and from \( \overline{x}_i \) to \( x_i \). If it is the case, the CNF formula is not satisfiable. If no paths of these types are found simultaneously for any variable, the formula is satisfiable. To find an evaluation solution of the 2-SAT, we use the BinSat algorithm, presented as follows:

Algorithm 1: BinSat;
Input: \( G_{2-SAT} \)
Output: S: solution of the 2 - SAT
Begin
repeat
  choose a vertex not yet marked, let be \( x_i \) this vertex
  if there exist a path from \( x_i \) to \( \overline{x}_i \) then
    mark \( x_i \) true and \( \overline{x}_i \) false
  else
    mark \( x_i \) true
    propagate this truth value by connexity in the graph
    for all the vertices thus marked, mark false their negations.
  end
until All the vertices are marked;
End

This algorithm requires \( O(l + 2m) \) time, where \( m \) is the number of clauses in \( F \) and \( l \) the number of variables.

Exemple 2.2 In the \( G_{2-SAT} \) graph of the formula \( F = (\overline{x}_1 \lor x_2) \land (\overline{x}_2 \lor x_3) \land (\overline{x}_1 \lor \overline{x}_3) \), given in the first example, there is no cycle from \( x_i \) to \( \overline{x}_i \) for \( i \in \{1, 2, 3\} \) then \( F \) is satisfiable. At the beginning, no vertex is marked. Choose for example \( x_3 \). There is no paths from \( x_3 \) to \( \overline{x}_3 \), so mark true \( x_3 \) and propagate this value to \( \overline{x}_1 \). At this step, mark \( \overline{x}_3 \) and \( x_1 \) false. The only vertex not yet marked is \( x_2 \). Since there is no paths from \( x_2 \) to \( \overline{x}_2 \), mark \( x_2 \) true, and it’s negation \( \overline{x}_2 \) false. Hence,
the solution of the formula $F$ given by Algorithm 1 corresponds to the assignment of true values of the vertices $\bar{x}_1$, $x_2$ and $x_3$, that is to say: $S = 011$.

### 2.2. The BinSat* algorithm

As the 2-SAT problem can have several solutions, we propose for our cryptosystem to use Algorithm 1, but by marking each time the vertex with the smallest index, the not yet marked (vertices are marked in their natural order). This new version of Algorithm 1 is denoted BinSat*.

**Algorithm 2 BinSat*:**

**Input:** $G_{2-SAT}$

**Output:** $S$: solution of the $2-SAT$

**Begin**

repeat

choose a vertex of smallest index not yet marked, let be $x_i$ this vertex

if there exist a path from $x_i$ to $\bar{x}_i$ then

mark $x_i$ false and $\bar{x}_i$ true

else

mark $x_i$ true propagate this value by connexity in the graph

for all the vertices thus marked, mark false their negations.

end

end

until All the vertices are marked;

**End**

### 3. Description of the cryptosystem

The proposed cryptosystem is described in three steps, keys generation, encryption and decryption. The cryptosystem proposed here is a hybrid cryptosystem based on the 3-SAT problem for encryption and the best known public key RSA algorithm [8], for sharing the key. This cryptosystem is a hybrid cipher block, where the message is divided into blocks of size $l$. We generate a random vector of size $l$ and note it $SS$, then we generate a 2-SAT having $SS$ as solution using Algorithm 3,
which is solved again by Algorithm 2, its solution $S$, will be the secret key. After that, we mask the 2-SAT in a 3-SAT using Algorithm 4, that we publish. Encryption proceeds in one step. From the inputs $M$ (the plaintext) and $S$ produce the output $C$ such that $C = S \oplus M$.

The Figure 2 illustrates the general scheme of the approach.

![Flowchart](image)

**Figure 2: The illustration scheme**
Traditionally, in cryptography the sender is called Alice and the receiver Bob. Our cryptosystem for Alice and Bob consists of the following steps.

3.1. Keys generation

1) Alice generates randomly a binary vector of size \( l \) and denotes it \( S_S \).

2) Alice generates randomly a 2-SAT with \( m \) clauses having \( S_S \) as solution by using Algorithm 3, called C-2-SAT, presented below. This step ensures that the 2-SAT created admits at least a solution \( S_S \).

3) Alice solves the 2-SAT problem obtained with Algorithm 2. Let \( S = (s_1, s_2, \ldots, s_l) \) be its solution, which will represent the secret key.

4) After creating the 2-SAT, Alice masks it as a 3-SAT problem, which is difficult, using Algorithm 4, called C-3-SAT. For each clause, it proceeds as follows: it adds to each clause one literal at the appropriate position and saves the position of the added literal \( v_k \) in a vector \( V \) and creates, finally, the integer \( a = v_1v_2\ldots v_m \).

5) Alice publishes the obtained 3-SAT problem which forms its public key.
Algorithm 3  C-2-SAT

Input: $SS$ a binary vector and $m$ an integer

Output: 2-SAT with $m$ clauses

Begin

$E := \emptyset$

$k := 1$

$c := true$

while $k \leq m$ do

while $c = true$ do

repeat

$i := \text{trunc} (\text{random}(l))$

$j := \text{trunc} (\text{random}(l))$

until $i \neq j$

$b := \text{random}(1)$ if $b \geq 0.5$ then

| add $x_j$ to $C_k$

else

| add $\bar{x}_j$ to $C_k$

end

if $SS[i] = 1$ then

| add $x_i$ to $C_k$

else

| add $\bar{x}_i$ to $C_k$

end

if $C_k \notin E$ then

| $E := E \cup C_k$

| $c := false$

end

$k := k + 1$

end

$c := true$

End

Note that Algorithm 3 runs in time $O(m^2)$. 
Algorithm 4  C-3-SAT:
Input: 2-SAT with ℓ variables, m clauses and S its solution
Output: 3-SAT
Begin /*i and j design through this algorithm the indexes of variables which form the clause Ck*/
k := 1
while k ≤ m do
  repeat
    p := trunc(random(ℓ))
    b := random(1)
    if p < i then
      V[k] := 1
      if b ≥ 0.5 then
        add x_p to C_k at the first position,
      else
        add \(\bar{x}_p\) to C_k at the first position,
      end
    end
    if (p > i) \& (p < j) then
      V[k] := 2
      if b ≥ 0.5 then
        add x_p to C_k at the second position
      else
        add \(\bar{x}_p\) to C_k at the second position
      end
    end
    if p > j then
      V[k] := 3
      if b ≥ 0.5 then
        add x_p to C_k at the third position
      else
        add \(\bar{x}_p\) to C_k at the third position
      end
    end
  until (p < i) \lor ((p > i) \& (p < j)) \lor (p > j);
  k := k + 1
end
End
This algorithm runs in time $O(m)$. Notice that Algorithm 3 and Algorithm 4 have almost the same idea so, we apply only Algorithm 4 for the previous example.

**Exemple 3.1** Let $F = (\bar{x}_1 \lor x_2) \land (\bar{x}_2 \lor x_3) \land (\bar{x}_1 \lor \bar{x}_3)$ be a 2-SAT problem.

- $k = 1$. We consider the first clause $C_1 = (\bar{x}_1 \lor x_2)$ then $i = 1$ and $j = 2$. Suppose that $b = 0.25$ and $p = 3$, then $V[1] = 3$ and add $\bar{x}_3$ to the first clause at the third position, so we get $C_1 = (\bar{x}_1 \lor x_2 \lor \bar{x}_3)$.

- $k = 2$. We consider the second clause $(\bar{x}_2 \lor x_3)$, then $i = 2$ and $j = 3$. Suppose that $p = 1$, $b = 0.76$, then $V[2] = 1$ and add $x_1$ to the second clause at the first position, so we get $C_2 = (\bar{x}_2 \lor x_3 \lor x_1)$.

- $k = 3$. We consider the last clause $(\bar{x}_1 \lor \bar{x}_3)$, then $i = 1$ and $j = 3$. Suppose that $p = 2$, $b = 0.12$, then $V[3] = 2$ and add $\bar{x}_2$ to this clause at the second position, so we get $C_3 = (\bar{x}_1 \lor \bar{x}_3 \lor \bar{x}_2)$.

The resulted 3-SAT is then:

$$F = (\bar{x}_1 \lor x_2 \lor \bar{x}_3) \land (\bar{x}_2 \lor x_3 \lor x_1) \land (\bar{x}_1 \lor \bar{x}_3 \lor \bar{x}_2).$$

### 3.2. Encryption

When Alice wants to send a binary message $M = m_1 m_2 \ldots m_l$ to Bob, she reads for example the value $(n, e)$ Bob's RSA public key and calculates $\hat{a} = a^e \pmod{n}$. The cipher text $C$ is determined by $C = S \oplus M$, where $S$ is the solution of the 2-SAT resulted from Algorithm 3 and $\oplus$ denotes the sum in $\mathbb{Z}/2\mathbb{Z}$, then transmits $(C, \hat{a})$.

### 3.3. Decryption

To decrypt the message $C' = c_1 c_2 \ldots c_l$, Bob, using its RSA secret value $d$, calculates $a = \hat{a}^d \pmod{n}$, and then deduces the vector $V$, therefore; the position of added variables. He removes the variables added to the published 3-SAT of Alice, and then obtains the corresponding 2-SAT, thereafter its solution $S = (s_1, s_2, \ldots, s_l)$ by using Algorithm 3, and deduces the message of Alice, by $S \oplus C' = M$. 
3.4. Explicit example

Let us note that the parameters used in this example are artificially small. In practice it is recommended to use a large parameters.

1) Keys generation

Alice chooses $SS = 01101$, $m = 3$ and executes Algorithm 3:

$k = 1 : i = 1, j = 2, b = 0.22, SS[1] = 0 \Rightarrow C_1 = (\bar{x}_2 \lor \bar{x}_1)$,
$k = 2 : i = 3, j = 4, b = 0.55, SS[3] = 1 \Rightarrow C_2 = (x_4 \lor x_3)$,
$k = 3 : i = 4, j = 5, b = 0.63, SS[4] = 0 \Rightarrow C_3 = (x_5 \lor \bar{x}_4)$.

She obtains then the 2-SAT:

$$(\bar{x}_1 \lor \bar{x}_2) \land (x_3 \lor x_4) \land (\bar{x}_4 \lor x_5).$$

Using Algorithm 2, she gets $S = 10111$.

Then, she runs after Algorithm 4:

$k = 1 : p = 4, b = 0.27 \Rightarrow C_1 = (\bar{x}_1 \lor \bar{x}_2 \lor \bar{x}_4), V[1] = 3$,
$k = 2 : p = 2, b = 0.68 \Rightarrow C_2 = (x_3 \lor x_2 \lor x_4), V[2] = 2$,
$k = 3 : p = 1, b = 0.11 \Rightarrow C_3 = (\bar{x}_1 \lor \bar{x}_4 \lor x_5), V[3] = 1$.

Hence, Alice’s public key is the 3-SAT:

$$(\bar{x}_1 \lor \bar{x}_2 \lor \bar{x}_4) \land (x_3 \lor x_2 \lor x_4) \land (\bar{x}_1 \lor x_4 \lor x_5).$$

The vector $V = (3, 2, 1)$, then $a = 321$.

2) Encryption

Let $(n, e) = (36581, 5)$ be Bob’s RSA public key. Alice ciphers $a$, i.e., $\hat{a} = a^e \pmod{n} = 321^5 \pmod{36581} = 2677$ and its message $M = 0110001101$, using $S$, i.e., $C = M \oplus S$.

Alice sends Bob $(C, \hat{a}) = (1101111010, 2677)$. 
3) **Decryption**

To decrypt the message $C$, Bob uses its secret value $d = 14477$ and calculates $a = a^d \pmod{n} = (2677)^{14477} \pmod{36581} = 321$. Then, he removes then the third literal from the first clause of Alice’s 3-SAT, the second one from the second clause, and the first from the third, thus, obtains the 2-SAT of Alice. Using BinSat* algorithm, he determines the solution $S = 10111$. At the end, he calculates the message $M$ by $M = S \oplus C$ and obtains finally $M = 0110001101$.

**4. Security of the cryptosystem**

The 3-SAT problem we have used, is a particularly interesting subclass of satisfiability problem: it is the random 3-SAT. It is well known that there exist some algorithms able to find exact solution for the 3-SAT problem, but the enumeration of all it’s solutions is clearly more difficult. For the example given in section 3.4 above, the 3-SAT problem is satisfiable, and it admits 23 solutions, among them is our solution $S = 10111$. Here, for illustration, we have considered a small instance, in practice we generate instances with large parameters for which finding one solution does not solve the problem.

**5. Conclusion**

In this paper, we propose a hybrid cryptosystem based on a subclass of the well-known satisfiability problem, namely 3-SAT problem, using the BinSat* algorithm for the 2-SAT problem. The cryptosystem was presented in three steps: keys generation, encryption and decryption, followed by an explicit example.

**Acknowledgements**

The authors gratefully acknowledge the referees for their helpful remarks on this paper.
References


Sadek BOUROUBI (Algiers, August 1964) is a full Professor at University of Science and Technology of Houari Boumediene (USTHB). He received the Magister degree in Mathematics from USTHB in 1991 and received his habilitation (Doctorat d’Etat) at USTHB on Operational Research in 2004. He is a research director and is in charge of Laboratory of Decision Aid LAID3 of USTHB. He is the author of several articles and his current interests include combinatorial optimization, combinatorial problems and cryptography.

Louiza REZKALLAH (Bouira, May, 1980) received his Magister in operations research from USTHB University, Algiers, Algeria, in 2007. Currently she is a doctoral student at the same University and a lecturer in Mathematics at Saad Dahlab University Blida, Algeria. His current interests include combinatorial optimization, metaheuristics and cryptography.