A Polly Cracker system based on Satisfiability

Françoise Levy-Dit-Vehel, Ludovic Perret

A Polly Cracker system based on Satisfiability

Françoise Levy-dit-Vehel — Ludovic Perret

N° 4698

Janvier 2003

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4698--FR+ENG

A Polly Cracker system based on Satisfiability

Françoise Levy-dit-Vehel * , Ludovic Perret*

Thème 2 — Génie logiciel
et calcul symbolique
Projet Codes

Rapport de recherche n° 4698 — Janvier 2003 — 15 pages

Abstract: This paper presents a public-key cryptosystem based on a subclass of the well-known *satisfiability* problem from propositional logic, namely the *doubly-balanced* 3-SAT problem. We first describe the construction of an instance of our system starting from such a 3-SAT formula. Then we discuss security issues: this is achieved on the one hand by exploring best methods to date for solving this particular problem, and on the other hand by studying (systems of multivariate) polynomial equation solving algorithms in this particular setting. The result of our investigations is that both types of method fail to break our instances. We end the paper with some complexity considerations and implementation results.

Key-words: combinatorial-algebraic cryptosystems, 3-SAT, hard instances generation, systems of polynomial equations.

* ENSTA, Laboratoire de Mathématiques appliquées, 32 bd Victor, 75739 Paris cedex 15.

Un système de type Polly Cracker basé sur le problème de la satisfaisabilité

Résumé : Nous présentons ici un système de chiffrement à clef publique construit à partir d'une sous-classe du problème de satisfaisabilité de la logique propositionnelle, appelé problème 3-SAT *doublement balancé.* Nous décrivons d'abord la construction d'une instance de notre système à partir d'une telle formule 3-satisfaisable. Puis nous étudions la sécurité d'un tel système, d'une part en explorant les meilleures méthodes actuelles pour résoudre ce problème sous-jacent, d'autre part par l'étude d'algorithmes de résolution de systèmes d'équations multivariées dans ce contexte. Nous concluons que notre système est résistant à ces deux types de méthodes. Des considérations pratiques de complexité et d'implantation terminent l'article.

Mots-clés : cryptosystèmes d'essence combinatoire, 3-SAT, génération d'instances difficiles, systèmes d'équations polynomiales.

1 Introduction

Since the failure of knapsack-based cryptosystems [Sh, Od], a widely accepted opinion was that NP-complete problems were not suited for the construction of secure trapdoor one-way functions. In 1993, M. Fellows and N. Koblitz [FK] proposed to further investigate the use of those problems for designing public-key cryptosystems, and proposed a general framework, called CA-systems[1], the main illustration of which was the Polly Cracker cryptosystem. In this system, the public-key is a set $S = \{p_1, \ldots, p_\ell\}$ of multivariate polynomials over a finite field $\mathbb{F}_q$, and the secret-key is a zero α of S. To encrypt a message $M \in \mathbb{F}_q$, Bob chooses an element $\psi = \sum_{i=1}^{\ell} h_i p_i$ of the ideal generated by S, and sends $c = \psi + M$. Knowledge of α then allows Alice to decrypt the ciphertext just by evaluating it on α.
The (public-key, secret-key) pair is derived from an instance of an NP-complete combinatorial[2] problem, in such a way that knowing the public-key is equivalent to knowing the considered instance, and that finding a secret-key from the public-key is equivalent to finding a solution for this particular instance.
M. Fellows and N. Koblitz suggest several NP-complete problems for use in this context, mainly based on graph theory (e.g. 3-colorability, perfect codes in graphs,...). The public-keys arising from such problems suffer from an intrinsic weakness - namely, some public-key polynomials are linear - as pointed out by R. Steinwandt et al. [GS]. Besides, algorithms that are fast on average have been found for several NP-complete graph problems, such as k-colorability, under commonly used distributions on graphs [Wa].

Here, we follow the CA-systems line of research by proposing a public-key cryptosystem based on the well-known SATISFIABILITY problem from propositional logic. More precisely, we use the 3-SAT problem. One advantage of using this underlying hard problem is that it avoids the presence of degree one polynomials in the public-key.

not seem

The 3-SAT problem has been extensively studied, see e.g. [CMi, Mo, CMo, Ro], mainly due to the fact that it is of interest in other research areas, such as planning or scheduling. Although proven to be NP-complete, this problem admits many "easy" instances, where deterministic algorithms (such as the recursive DPLL[DL]) perform quite well in practice. Indeed, let n (resp. m) denote the number of variables (resp. clauses) of the problem, and set $m = cn$ with $c \in \mathbb{R}^{*+}$. Then, as c increases, it has been shown experimentally that the probability of an instance of chosen for 3-SAT being satisfiable shifts from almost one to almost zero. The range of c over which this transition occurs is[3] $3.003 < c < 4.598$. This is known as the *threshold conjecture.* In this range, there is a value of c corresponding to a complexity peak at which on average half of the instances are satisfiable. The exact value of c yielding this peak can be numerically determined for each instance distribution.
c thus Non-deterministic methods have also been devised, that often give better results on

[1] For "combinatorial-algebraic" cryptosystems.

[2] In a broad sense, i.e. this includes graph theory, boolean logic,...

[3] For 3-SAT ; For k-SAT with higher values of k, this range is shifted. Also, the higher n is, the sharper the range becomes.

satisfiable instances (e.g. Walksat, [SK]), especially near the threshold region. They are known as *local search methods.*
For small values of n - say up to several thousands - all instances are within reach of current algorithms, i.e. those will almost always find a solution if any, or prove there are no (the search tree of DPLL, in case of unsatisfiable instances, gives a proof thereof). Thus, the hardness of this problem is tightly located in the critical range for c, and for (very) large values of n. Having this in mind, and also that the parameter sizes and generation times of our system have to be polynomial[4], we chose to restrict ourselves to a particular class of the 3-SAT instances, namely the class of so-called *doubly-balanced* 3-SAT [DB], a.k.a. *literal-regular* 3-SAT [BS]. Formulae in this class have the particularity that every variable appears (almost) equally often, and (almost) as often negated as unnegated. Instances from this class are much more difficult to solve in general than random 3-SAT instances, as they are designed to have structural regularities, thus confusing variable selection heuristics that are used by most solvers (for example, DPLL-like algorithms treat the variables with a small number of occurences first.)

Note that for random 3-SAT the complexity peak occurs for $c \approx 4.25$, while for doubly-balanced 3-SAT , it has been shown to be $c \approx 3.5$ (both values experimentally determined).

The paper is organized as follows: in the next section, we begin by providing the necessary background to understand the basics of the 3-SAT problem, as well as methods for generating random instances, and doubly-balanced ones. Then we show how to translate this problem into a system of polynomial equations, in order to use it in our cryptographic setting. We exhibit the correspondence existing between the models of 3-SAT and the solutions of the system. In section 3, we describe the cryptographic scheme we propose. Section 4 is devoted to its security: we investigate carefully the single break attacks found by R. Steinwandt et al. [GS], and show that they cannot be conducted in our context. We also investigate total break methods on the system. They are of two types: the first type is the use of 3-SAT solvers to break the considered instances, from which we protected ourselves by carefully choosing the instances. The second type is to run algorithms computing (an element of) the variety of the set of polynomials involved. The best algorithm known to us - namely F_4 [Fa] - does in fact more: it computes a Gröbner basis of the set of polynomials. For the considered sizes, it appears that such an algorithm is of no help. We end the paper by a section concerning complexity of parameter generation, as well as implementation aspects: parameter sizes, run times... We would like to mention that, when investigating Polly Cracker-type systems, our intention was not to design a scheme that was likely to compete with the public-key systems in use. What we were interested in was mainly to show that it was possible to achieve a reasonable level of security while using this type of systems; we also believe that our approach of the SATISFIABILITY problem from a cryptographic standpoint is quite original.

[4]In the size of the input of 3-SAT, namely $n\lg(n)$, denoting by lg() the base-two logarithm.

2 CNF formulae and systems of polynomial equations

2.1 3-SAT and instance generation methods

We begin by recalling what the 3-SAT problem is. Let $X = \{x_1, \ldots, x_n\}$ be a set of boolean variables, and let $\wedge$, $\vee$, $^-$ denote logical *and, or, not* respectively. A *truth assignment* for X is a function $t : X \mapsto \{True, False\}$. For a variable $x_j \in X$, if $t(x_j) = True$, we say that x_j is true under t. (If $t(x_j) = False$, then $\bar{x_j}$ is true under t). For $1 \leq j \leq n$, a *literal* u_j is either x_j or $\bar{x_j}$. A *clause* over X is the disjunction of a set of literals over X. It is satisfied by a truth assignment if, and only if at least one of its members is true under that assignment. A clause containing only three literals will be called a *3-clause*. For instance, $c = x_{j_1} \vee \bar{x_{j_2}} \vee x_{j_3}$, $1 \leq j_1, j_2, j_3 \leq n$, is a 3-clause, and is satisfied unless $t(x_{j_1}) = False$, $t(x_{j_2}) = True$, $t(x_{j_3}) = False$. A *CNF-formula*[5] $\mathcal{C}$ is the conjunction of arbitrarily many clauses $c_1, \ldots, c_m$, $m \in \mathbb{N}^*$. It is satisfiable if, and only if there exists some truth assignment for X that simultaneously satisfies all the clauses in $\mathcal{C}$. Such a truth assignment is called a *satisfying truth assignment*, or a *model* for the formula $\mathcal{C}$. If $\mathcal{C}$ contains only 3-clauses, then we say that $\mathcal{C}$ is a *3-CNF formula*. For instance, $\mathcal{C} = \wedge_{j=1}^{m} c_j$ where $c_j = u_{j_1} \vee u_{j_2} \vee u_{j_3}$, $m \in \mathbb{N}^*$, is such a formula.
In the sequel, we shall denote a CNF-formula either as a conjunction of clauses as above, or equivalently as a collection of clauses, the conjunction then being implicit.

The 3-satisfiability problem can then be stated as follows:

INSTANCE: a collection $\mathcal{C} = \{c_1, \ldots, c_m\}$ of 3-clauses on X.
QUESTION: is there a satisfying truth assignment for $\mathcal{C}$?

The random 3-SAT problem which we referred to in the introduction is the 3-SAT problem in which instances are generated according to the following procedure[6]:
The number of variables n and the number of clauses m being fixed, randomly select three distinct variables out of n, then negate each variable with probability 1/2. Combine these literals in a 3-clause. Repeat this process until the desired number m of clauses is reached. Conjoined them to form a CNF-formula.

The restriction of 3-SAT to balanced formulae is the one in which a formula $\mathcal{C}$ is such that, for all i, $1 \leq i \leq n$, each variable x_i appears equally often[7] i.e. in $\lfloor 3m/n \rfloor$ clauses (there are $3m$ positions to fill, corresponding to the m 3-clauses). But then, it can be that some variables appear more often negated than unnegated (or the converse). The doubly-balanced 3-SAT subclass is precisely the class of formulae that do not present this type of irregularity; namely, a formula in this class is such that each literal appears (almost) $3m/(2n)$ times (there are $2n$ possible literals). Such instances can be generated with the following algorithm:
The number of variables n and the number of clauses m is being fixed. Place $\lfloor 3m/(2n) \rfloor$

[5]Conjunctive Normal Form.
[6]Fixed Clause Length generation.
[7]Almost: occurences of some variables must be added if $3m/n$ is not an integer.

occurences of each of the $2n$ literals in a bag. To reach exactly $3m$ literals in the bag, add randomly some literals, not twice the same. To construct each clause, remove three literals on distinct variables from the bag. At some point, if the literals remaining in the bag concern only one or two distinct variables, then randomly add distinct variables in the bag, negating each of them with probability 1/2. Keep on the construction of the clauses until the desired number is reached.

Note that to generate a (doubly-balanced) formula admitting a particular model y, one simply modifies the above procedure by throwing away the 3-clauses that are not satisfied by y.

2.2 Constructing a system of polynomial equations from 3-SAT

We shall now explain how to translate an instance of the 3-SAT problem into a system of polynomial equations. A similar description already appeared in [Ba].

We shall denote by $K[X]$, the polynomial ring $K[x_1,\ldots,x_n]$ over the field K. We choose two field values $T, F \in K$, representing True and False respectively. To a 3-clause c involving the three literals u_j, u_k, u_ℓ, $1 \le j,\ k,\ \ell \le n$, one can associate a total degree 3 polynomial in $K[X]$ as follows: if $u_j = x_j$, then we replace u_j by $(x_j - T)$; if $u_j = \bar{x_j}$, then we replace it by $(x_j - F)$. Replace $\vee$ by multiplication. For instance, the polynomial[8] $p_c(X) \in K[X]$ correponding to the clause $c = x_j \vee \bar{x_k} \vee x_\ell$ is $p_c(X) = (x_j - T)(x_k - F)(x_\ell - T)$. Is is then clear that a satifiying truth assignment of X for c corresponds to a zero of the polynomial $p_c(X)$. Moreover:

Theorem 1. *A 3-CNF $\mathcal{C} = \{c_1,\ldots,c_m\}$ admits a model if, and only if the corresponding system of polynomial equations $\{p_1(X) = 0,\ldots,p_m(X) = 0\}$ has a solution over the algebraic closure of K.*

Proof
We shall indeed prove more: first, we exhibit a one-to-one correspondence between the models of $\mathcal{C}$ and a variety we call $\mathcal{V}$ defined below:
For an ideal $J \subset K[X]$ generated by a set S, we shall here denote by $V_K(J)$, the variety of S, that is:

$$V_K(J) = \{z \in K^n,\ f(z) = 0,\ \forall f \in S\}.$$

Now denote by I, the ideal generated by the polynomials p_i, $1 \le i \le m$, and let y be an element of the variety

$$V_K(I) = \{z \in K^n, p_i(z) = 0,\ \forall 1 \le i \le m\}.$$

Then we claim that the models of $\mathcal{C}$ are in one-to-one correspondence with the elements of $V_K(I)$ with coordinates in $\{T,F\} \subset K$: first, note that those elements indeed form the variety

$$V_K(I) \cap (\cap_{j=1}^{m} V_K(<(x_j - T)(x_j - F)>) = V_K(<\{p_i\}_{1\le i\le m}, \{(x_j - T)(x_j - F)\}_{1\le j\le n}>).$$

[8]Letting X stand for $x_1,\ldots,x_n$.

Denote this variety by $\mathcal{V}$. Then

$$y \in \mathcal{V} \Leftrightarrow (p_i(y) = 0 \ \forall 1 \leq i \leq m \text{ and } y_j \in \{T, F\} \ \forall 1 \leq j \leq n)$$

$$\Leftrightarrow \forall 1 \leq i \leq m,\ c_i \text{ is satisfied by } t_y : X \to \{True, False\} \text{ given by:}$$

$$\text{for } 1 \leq j \leq n, \text{ if } y_j = T \text{ then } t_y(x_j) = True, \text{ if } y_j = F \text{ then } t_y(x_j) = False.$$

This exactly states that t_y is a model of $\mathcal{C}$.

Next, letting $\overline{K}$ stand for the algebraic closure of K, we prove that every element of $V_{\overline{K}}(I)$ can be mapped to an element of $\mathcal{V}$:
Let $z \in V_{\overline{K}}(I)$. If $z_j \in \{T, F\}$, $\forall 1 \leq j \leq n$, nothing has to be proven. Otherwise, there exists (at least one) j, $1 \leq j \leq n$ with $z_j \in \overline{K} \setminus \{T, F\}$. Call j_0 one such js. Due to the form of the polynomials p_i, as we have $p_i(z) = 0 \ \forall 1 \leq i \leq m$, it must be that the value of the variable x_{j_0} has no influence on the cancellation of the p_is: in other words, for each p_i, we have $p_i(z) = 0$ thanks to some other value z_{j_i} with $j_i \neq j_0$ (variable x_{j_0} can be considered as a "free" variable in this system); and we must evidently have $z_{j_i} \in \{T, F\}$. Thus z can be mapped to the element $\tilde{z} \in \mathcal{V}$ defined by $\tilde{z_j} = z_j$ for all js such that $z_j \in \{T, F\}$, and $\tilde{z_j} = T$ otherwise[9]. □

We shall here use for K a finite field $\mathbb{F}_q$. We ask that T and F be two non-zero field elements, so we set $q \geq 3$.

3 The system

Selecting the Public-key/Secret-key pair

Alice chooses a finite field $\mathbb{F}_q$ with $q \geq 3$, and positive integers m and n. She also takes a vector y of $\{T, F\}^n$ at random. This is her secret-key.

She then generates an instance $\mathcal{C}$ of doubly-balanced 3-SAT admitting y as model. now.[...] For this, she uses a generation method due to E. Hirsch [Hi] and called *hgen2*. This method follows the one described in section 2.1, but with some other constraints, that aim to generate formulae with as independent clauses as possible. For instance, if a clause involves literals u_j, u_k and u_ℓ, then his algorithm is designed such that no other clause of $\mathcal{C}$ involve any two of them.

Having done this, Alice publishes the set $\mathcal{C}$, together with m, n and q (values T and F are also publicly known). In section 5, we shall explain how we represent $\mathcal{C}$. Indeed, as shown in section 2.2, it would have been equivalent - from an information theoretic viewpoint - to publish the m polynomials correponding to these m clauses, but the "clause-representation" allows for a more compact form.

[9]We can equivalently set it to F.

Encryption

The encryption phase follows a regular Polly Cracker scheme. Let $M \in \mathbb{F}_q$ be the plaintext to be transmitted. To encrypt M, Bob first chooses at random $m' = \beta m$ clauses[10] from the total set of m clauses of the public-key $\mathcal{C}$, with $0 < \beta < 1$. We recommend $1/4 \leq \beta \leq 1/2$. Then, he transforms the chosen clauses $c_{j_1}, \ldots, c_{j_{m'}}$ into polynomials $p_{j_\ell}(x)$, as explained above. Next, Bob chooses m' random polynomials $h_\ell(X) \in \mathbb{F}_q[x_1, \ldots, x_n]$. To keep a reasonable complexity in terms of the size of the polynomials, and of the time needed for the generation, the h_ℓ's are chosen to be quadratic polynomials involving a moderate number of terms, e.g. $\lg n$ or possibly $\sqrt{n}$ terms. The ciphertext he sends to Alice is then the total degree 5 polynomial of $\mathbb{F}_q[x_1, \ldots, x_n]$ defined by :

$$c(X) = \sum_{\ell=1}^{m'} h_\ell(X) p_{j_\ell}(X) + M.$$

Decryption

Upon receiving $c(X)$, Alice evaluates it at y to recover the plaintext:

$$c(y) = \sum_{\ell=1}^{m'} h_\ell(y) p_{j_\ell}(y) + M = M.$$

4 Security issues

Total break

Given a system of m polynomial equations of total degree 3 in n variables over a finite field $\mathbb{F}_q$, the security of our scheme relies on the difficulty of finding a solution to this system in the algebraic closure of $\mathbb{F}_q$. finite In other words, if I denotes the ideal generated by these polynomials, the problem is to find an element of $V_{\overline{\mathbb{F}_q}}(I)$. This problem can be solved by means of computing a Gröbner basis of I. In this case, this gives in fact all the elements of $V_{\overline{\mathbb{F}_q}}(I)$. The complexity of computing a Gröbner basis of a system of polynomials - although theoretically doubly exponential in the number of variables - depends in practice very much on the nature of the system, and of the algorithm used. To our knowledge, the best current algorithm for computing Gröbner bases is F_4, and is due to J.C. Faugère [Fa]. systems, Faugère HFE's variables memory.

We have run the F_4 algorithm on instances of our scheme via the web interface[11] of Fgb. Practically, yet for polynomial systems corresponding to $n = 100$ variables and $m = 350$ clauses, such an algorithm fails computing a Gröbner basis: indeed, we have noticed that, after some iterations, the algorithm cannot terminate, due to the handling of huge matrices (typically square matrices of a hundred thousand entries). Thus, the sizes we consider are far out of reach of this type of algorithms.

[10]This choice was made for the sake of efficiency.
[11] http://calfor.lip6.fr/ jcf/Software/Fgb/index.html

Another attempt to break our system would be to run a 3-SAT solver on it. It is clear that the crucial point in using the 3-SAT problem in a Polly Cracker system lies in the method chosen for generating hard satisfiable instances. While it remains an open problem to generate hard solved instances [IL], the doubly-balanced 3-SAT formulae are among the hardest 3-SAT instances to solve by currently known methods: this is due to the fact that they are not completely random, as instances from the random 3-SAT problem can be, nor completely "structured" (this terminology refers to 3-SAT instances arising from the modelisation of real-life phenomena occurring in e.g. planning or scheduling). Thus, efficient algorithms on random formulae such as UnitWalk or OKsolver [Sa] will be defeated by the regularity of those formulae, whereas algorithms that perform well on structured instances - like Zchaff or Sato [Sa] - will then behave poorly, those formulae being too "random" to handle. The ones chosen by us for the construction of our public-keys come from the *hgen2* generator of E. Hirsch. Formulae generated by this algorithm have proven to be the ones that best resist to known solvers. Besides, instances from this generation method have won the smallest (in terms of n) satisfiable unsolved instance challenge of the SAT'2002 competition[12]: the smallest such instance had parameters $n = 500$ and $m = 1750$.

Single break

We recall that this attack, as opposed to the total break one, consists of recovering the cleartext from a ciphertext, but does not recover a secret key, thus in principle not compromising other uses of the system.

R. Steinwandt et al. [GS] have pointed out two attacks allowing to retrieve the cleartext from a ciphertext, both based on the fact that, as the cleartext is an element of $\mathbb{F}_q$, it is sufficient to compute the coefficients $h_\ell(0)$ to obtain it. The one based on the assumption that the terms of the public polynomials all belong to a small[13] subset of $\mathbb{F}_q[X]$ is not applicable here. The other one assumes the presence of so-called *characteristic terms* in the public-key. Such a term X^{ν_i} in p_i is such that it does not appear as a term of p_j, for $j \neq i$, $1 \leq i, j \leq m$. Then, if one has the additional property that the term X^{ν_i} cannot be obtained as a product of the form $X^\mu X^\theta$, X^μ being a term of p_j and X^θ a term of h_j, with $j \neq i$ and $\theta \neq 0$, then one can recover the coefficient $h_i(0)$ simply by the equality $a_i X^{\nu_i} = \alpha_i X^{\nu_i} \times h_i(0)$, a_i being the coefficient of X^{ν_i} in the ciphertext, and α_i the coefficient of X^{ν_i} in p_i.
While it is the case in our system that, for all $1 \leq i \leq m$, p_i possesses at least one such characteristic term, the attack does not work, as the following method shows: retrieve for a fixed instance of balanced 3-SAT (n, m), we have first exhibited the characteristic monomials of each p_i. For fixed i, $1 \leq i \leq m$, let us call those monomials $\alpha_{i_1} X^{\nu_{i_1}}, \ldots, \alpha_{i_k} X^{\nu_{i_{k_i}}}$. For each of them - say $\alpha_{i_j} X^{\nu_{i_j}}$ - we have computed $V_{i_j} = a_{i_j} / \alpha_{i_j}$, where a_{i_j} is the coefficient of $X^{\nu_{i_j}}$ in the ciphertext. If the values V_{i_j} coincinde for all $1 \leq j \leq k_i$, that means that one can determine $h_i(0)$ uniquely (as being equal to V_{i_j}). If not, that means that for some monomials $\alpha_{i_j} X^{\nu_{i_j}}$, the second condition in Steinwandt's attack does not hold. We are then

[12] http://www.satlive.org/SATCompetition
[13]Namely of size less than a few thousand.

left with more than one possible value for $h_i(0)$, thus not being able to determine it uniquely. Of course, one could set $h_j(0)$ to be the most often taken value of V_{i_j}. But the thing is that there are not many characteristic terms for each p_i (i.e. k_i is small[14], typically 2 or 3), so that this sort of decision is likely to give a false value for $h_j(0)$. There is also a possibility that $a_{i_j} = 0$, i.e. that the term $X^{\nu_{i_j}}$ cancels in the ciphertext. In this case, it can be that, for a particular i, no value can be obtained for $h_i(0)$ (if this is the case for all $1 \leq j \leq k_i$).
In the table below, we have quoted, for $n = 50, 75$ 100 and $m = 4n$, the number of is such that $h_i(0)$ is uniquely determined, ambiguously determined, and not known at all. We have done these experiments for a ciphertext involving polynomials $\{h_i\}_{1 \leq i \leq m}$ with $\lg n$ terms, and with $\sqrt{n}$ terms respectively. It is plain that the more terms there are in the h_is', the less coefficients $h_i(0)$ can be determined. What the results tell is that an unsufficient proportion of these values can effectively be determined (on average 25% for $\sqrt{n}$, and for $\lg n$: around 30% for $n \leq 100$, and about 40% for larger values of n), thus rendering that attack infeasible on these instances. We precise that those tests have been run on 20 instances of each size, and that the values given are respectively the smallest number of is found among the 20 instances, the largest one, and the mean values of the results obtained for each of them. For example for $n = 50$, we had to find 200 values for $h_i(0)$; when the h_is involve $\sqrt{n}$ terms, we could get the right values of $h_i(0)$ for 54 indices i on average, one instance yielding only 47 indices, and one instance yielding 65 such indices. There were on average 131.2 values of $h_i(0)$ that were ambiguously determined (at least 122 and at most 146 such values, depending on the instances tested), and the number of indices i for which the values $h_i(0)$ were unknown is 14.8 on average (with min. equal to 8 and max equal to 26).
Finally, those tests were conducted under the assumption that all the polynomials of the public-key are used to construct the ciphertext (i.e. $m' = m$). In the case when $m' < m$, which we recommend, this attack is completely infeasible, as the adversary does not know which polynomials Bob has chosen.

	$n = 50$ $m = 200$			$n = 75$ $m = 300$			$n = 100$ $m = 400$		
$\sqrt{n}$	47	65	54	77	119	88.7	94	119	104.9
	122	146	131.2	175	209	198.4	258	296	283.7
	8	26	14.8	6	17	13	8	21	11.4
$\lg(n)$	50	75	56.4	77	108	95.8	122	164	148.6
	114	137	131.3	178	202	193.5	215	267	242.8
	8	16	12.4	4	16	10.8	2	14	8.6

5 Practical considerations

In this section, we consider $3.003 < c < 4.598$ and $1/4 \leq \beta \leq 1/2$, those quantities being typically chosen to be $c = 3.5$ and $\beta = 1/3$.

[14]Remember that the number of terms of p_i is at most 8.

The generation of the set $\mathcal{C}$ of clauses has been performed using the algorithm *hgen2*. Apart from that, the complete implementation of instances of our system has been done using the MAGMA symbolic language - which we found best suited for the manipulation of multivariate polynomials (multiplication, evaluation on a vector of $\mathbb{F}_q^n$, computation of the number of terms ...) - with interfaces in C.
in MAGMA in The single break attack has also been conducted with MAGMA.

5.1 Parameter sizes

public-key, The public-key consists of m 3-clauses in the variables x_i, $1 \leq i \leq n$. It can thus be stored using $3m\lg(n)$ bits, that is $0(n\lg(n))$ bits with $m = cn$.

The secret-key is n bits long, as we can identify T with 1 and F with 0 for its storage.

The size of the ciphertext c can be estimated as follows: recall that c is of the form:

$$c(X) = \sum_{\ell=1}^{m'} h_\ell(X) p_{j_\ell}(X) + M.$$

The number R of monomials in the ciphertext is at most[15] $\sum_{\ell=1}^{m'} t_\ell s_\ell + 1$, where t_ℓ is the number of monomials of p_{j_ℓ}, namely exactly[16] 8 in our case, and s_ℓ is the number of monomials of h_ℓ, which we bound from above by $\lg n$. With $m' = \beta m$ and $m = cn$, this yields $R = O(n\lg n)$. Besides, each ciphertext monomial can be stored using $O(\lg q + \lg n)$ bits: indeed, as each monomial $ax_1^{\alpha_1}...x_n^{\alpha_n}$ is of total degree at most 5 (we recall that the total degree of h_ℓ is two), it can be stored as $(a, (j_1, \alpha_{j_1}), \ldots, (j_5, \alpha_{j_5}))$, with $\sum_{i=1}^{5} \alpha_{j_i} \leq 5$. For each (j_i, α_{j_i}), one needs $\lg n + 3$ bits, monomial. yielding $\lg q + 5(\lg n + 3)$ bits for each monomial.
Finally, this gives a ciphertext of size $O((n\lg n)(\lg q + \lg n))$ bits. Summerizing, we have:

Proposition 1. *The proposed scheme, involving m polynomials in n variables over $\mathbb{F}_q$ ($m = cn$, $m' = \beta m$), has the following features:*

The public-key is of size $O(n\lg n)$ bits.

The secret-key is n bits.

The ciphertext is $O((n\lg n)(\lg q + \lg n))$ bits.

5.2 Running times

We first give the time needed for the generation of the set $\mathcal{C}$ of clauses, for different values of n, and for $m = 3.5n$, according to the *hgen2* implementation:

[15] Some terms possibly cancelling.
[16]Due to the fact that T and F are non-zero.

n	time (seconds)
250	0.78
300	0.79
400	1.42
500	2.23
600	3.03
700	4.39

We can evaluate the complexity of encryption and decryption in terms of n and $\lg q$: for clarity, first set $r = \lg q$. The complexity[17] of an operation (addition or multiplication) over $\mathbb{F}_q$ is known to be $O(\Theta(r))$, with $\Theta(r) = r \lg r \lg \lg r$. Then, we have:

Proposition 2. *For the system described, with $m = cn$ and $m' = \beta m$, the following complexities hold:*

> *The cost of generating the ciphertext is $O(n \lg n \Theta(r))$ bit operations, plus the cost of generating $O(nr \lg n)$ bits.*
>
> *The cost of decryption is $O(n \lg n \Theta(r))$ bit operations.*

Proof
The bit cost of generating the ciphertext c can be divided into three steps:

- the complexity of constructing the m' polynomials p_i starting from the m' clauses $c_{i_1}, \ldots, c_{i'_m}$ chosen by Bob. This can be estimated as $O(m')$, which is $O(n)$ as $m' = \beta m = \beta cn$.
- the complexity of generating the h_is: for each h_i, one has to generate $\lg n$ coefficients in $\mathbb{F}_q$ at random, that is $r \lg n$ bits. And the number of h_is to generate is $m' = \beta cn$, thus yielding $O(nr \lg n)$ bits to generate at random.
- the complexity of computing c: this is the heaviest part of the generation. We need to perform R multiplications of elements of $\mathbb{F}_q$ and at most R additions, where R is the number of monomials in c. This makes $O(R\Theta(r))$ bit operations. With the estimate on R given in the previous paragraph, we get $O(n \lg n \Theta(r))$ bit operations. Comparing to this, the cost of computing the exponents of the monomials (addition of integers less than 5) is negligible.

For decryption, we have to evaluate the complexity of evaluating c on some point of $\mathbb{F}_q^n$. A typical monomial - being of total degree at most five - can be evaluated with $O(1)$ multiplications in $\mathbb{F}_q$, i.e. with bit cost $O(\Theta(r))$. Then, R additions in $\mathbb{F}_q$ have to be performed, as there are R monomials in c. Thus we obtain a bit complexity of $O(R\Theta(r))$, that is $O(n \lg n \Theta(r))$. $\square$

[17]bit cost.

Of course, these complexities give only the orders of magnitude of the encryption/decryption costs. In particular, the decryption phase is much faster than the encryption one in practice. To have a more precise idea, we have quoted below real-time implementations. The table successively gives, for several values of n and for $m' = m/3 = (3.5n)/3$ and $m/4 = (3.5n)/4$ (when this value was not an integer, we took the integer part of it), the size of the public-key (PK), the encryption time, the number of terms of the ciphertext and the decryption time.

(n, m')	size of PK (KBytes)	encryption time (seconds)	Number of terms in ciphertext	decryption time (seconds)
(500,583)	5.8	32.16	29515	0.93
(500,437)	5.8	18.87	22276	0.71
(550,641)	6.5	47,80	36556	1.3
(550,481)	6.5	26,84	27438	0.97
(600,700)	7.2	58	38978	1.49
(600,525)	7.2	33.34	29824	1.14

6 Concluding remarks

We have presented a cryptographic scheme of Polly-cracker type, the underlying problem of which is based on a subclass of the family of SATISFIABILITY problems. We have examined its security on the one hand by considering single break attacks, and on the other hand by exploring the best known methods to date to attack the hard problem. Of course, the security arguments presented do not constitute a proof of the security of our system in a formal sense. A result about proven security seems hard to reach at the moment, but is within the scope of our current investigations. For the moment, we think that those results are yet quite interesting, first because we achieved some kind of (computational) security while using Polly Cracker-type schemes - which has never been reached so far - and also because we believe that our approach - namely the investigation of sharp methods from propositional logic and the setting of results in a cryptographic context - is quite new. and

References

[Ba] D. Bayer. The division algorithm and the Hilbert scheme. PhD. Thesis, Harvard University, Cambridge, Massachussets, 1982.

[BS] R.J. Bayardo Jr., R. Schrag. Using CSP look-back techniques to solve exceptionnally hard SAT instances. Proceedings of 2nd Int. conference on Principles and Practice of constraint Programming, 1996, pp. 46–60.

[CMo] S. Cocco, R. Monasson. Statistical physics analysis of the computational complexity of solving random satisfiability problems using backtrack algorithms. The European Physical Journal B 22, 2001, pp.505–531.

[CMi] S.A. Cook. D.G. Mitchell. Finding hard instances of the satisfiability problem: a survey. DIMACS Series in discrete mathematics and theoretical computer science, 1997.

[DL] M. Davis, G. Logemann, D. Loveland. A machine program for theorem proving. Communications of the ACM, 5, 1962, pp. 394–397.

[DB] O. Dubois, Y. Boufkhad. From very hard doubly balanced SAT formulae to easy unbalanced SAT formulae, variations of the satisfiability threshold. Proccedings of the DIMACS workshop on the satisfiability problem: theory and applications, March 1996.

[Fa] J.-C. Faugère. A new efficient algorithm for computing Gröbner basis: F_4. Journal of pure and applied algebra, vol. 139, 1999, pp. 61–68.

[FK] M. Fellows, N. Koblitz. Combinatorial cryptosystems galore ! Proceedings of the second international conference on "Finite Fields : theory, applications and algorithms", Las vegas 1993, Contemporary Mathematics, vol. 168, 1994, pp. 51-61.

[Hi] E. Hirsch. Private communication.

[IL] R. Impagliazzo, L. Levin, M. Luby. Pseudo-random number generation from one-way functions. Proc. of 21st STOC, 1989, pp. 12–24.

[Mo] R. Monasson. Threshold phenomena and complexity: a statistical physics analysis of the random satisfiability problem. School on complexity, CIRM, Marseille, 28 Janvier - 1 Fevrier 2002.

[Od] A. Odlyzko. The rise and fall of knapsack cryptosystems. Cryptology and computational number theory, proc. of Symposium on Applied Mathematics 42, AMS 1990, pp. 75–88.

[Ro] R. Rodošek. A new approach on solving 3-Satisfiability. Proc. of International Conference on Artificial Intelligence and Symbolic Mathematical Computing (AISMC), Austria 1996, pp. 197–212.

[Sa] http://www.satlive.org/SATCompetition

[SK] B. Selman, H. Kautz, B. Cohen. Noise strategies for improving local search. Proc. of AAAI-94, 1994, pp. 337–343.

[Sh] A. Shamir. A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. IEEE trans. on Information Theory IT-30, 1984, pp. 699–704.

[GS] W. Geiselmann, R. Steinwandt. Some cracks in Polly Cracker. Europäisches Institut für Systemsicherheit, Université de Karlsruhe, Tech. Report 01/01, 2001.

[Wa] J. Wang. Average-case computational complexity theory. In L. Hemaspaandra and A. Selmen, editors, Complexity Theory Retrospective II, Springer, 1997, pp. 295-328.

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Futurs : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399