Int J Inf Secur (2002) 1: 143–148 / Digital Object Identifier (DOI) 10.1007/s10207-002-0012-2

# Attacking a polynomial-based cryptosystem: Polly Cracker

**Rainer Steinwandt, Willi Geiselmann, Regine Endsuleit**

Institut für Algorithmen und Kognitive Systeme, Arbeitsgruppen Computeralgebra und Systemsicherheit, Prof. Dr. Th. Beth, Universität Karlsruhe, Am Fasanengarten 5, 76131 Karlsruhe, Germany;
E-mail steinwan@ira.uka.de, Fax: +49-721/608-5022

Published online: 9 April 2002 – 

**Abstract.** We describe several attacks on Polly Cracker, a public key cryptosystem proposed by Fellows and Koblitz. The first kind of attack shows that variations in the CPU time needed for evaluating polynomials can leak significant information about the secret key. This kind of attack might also be of interest when dealing with other cryptosystems using polynomial evaluations, like Patarin's hidden fields equations.

Next, we exhibit some "structural" weaknesses in Polly Cracker's encryption procedure. In particular, we demonstrate that with the parameters considered in a book by Koblitz it is often possible to reveal the private key easily.

**Keywords:** Public key cryptography – Multivariate polynomials – Cryptanalysis

## 1 Introduction

In [8, 11] Fellows and Koblitz present a public key cryptosystem called Polly Cracker. In this system, the public key consists of a finite set of multivariate polynomials with coefficients in some finite field, and the secret key consists of a common zero of these polynomials. According to [11] two kinds of attacks against Polly Cracker are known:

- The first (straightforward) one aims at finding a common zero of the polynomials in the public key – for a proper choice of the public key this approach should fail (Fellows and Koblitz suggest an encoding of instances of NP-hard combinatorial problems here).
- The second kind of attack makes use of linear algebra techniques and aims at the (probabilistic) encryption procedure of the system. To prevent this approach, in [11], a fine-tuned version of Polly Cracker is suggested that should not be susceptible to a linear algebra approach.

In this paper we describe several different attacks on Polly Cracker:

1. First, we show how to use an approach similar to Kocher's timing attack [13] for attacking public key cryptosystems, which are based on the evaluation of polynomials. It turns out that for some implementations of Polly Cracker this kind of attack has the potential for revealing the complete private key or significant information about it. Similarly, in hidden fields equations (HFE), information about the Hamming weight of the secret key can be derived.
2. Next, we show how ideas like those applied in [4, 5] against SPIFI (cf. [2, 3]) and ENROOT (cf. [3, 9]) can be adapted to the general instance of Polly Cracker. Using these techniques it is sometimes possible to decrypt individual ciphertexts without knowing the secret key.
3. Finally, we demonstrate that with the parameters considered in [11] for the fine-tuned version, the resulting cryptosystem is often weak. Namely, we demonstrate that by means of a computer algebra system it is often possible to reveal the private key within a few minutes.

## 2 Polly Cracker

We denote by $\mathbb{F}_q$ the finite field with $q$ elements and by $n \in \mathbb{N}$ a fixed natural number. Moreover, we assume that plaintexts are represented as elements of $\mathbb{F}_q$; ciphertexts are represented as polynomials in $n$ indeterminates $x_1, \ldots, x_n$ over $\mathbb{F}_q$. With these conventions the main ingredients of Polly Cracker are as follows (see [11] for more details):

– Alice's secret key is a random vector $\sigma \in \mathbb{F}_q^n$.
– Her public key is a finite set of polynomials $P \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$ such that $p(\sigma) = 0$ for all $p \in P$.
– To encrypt a plaintext $m \in \mathbb{F}_q$, Bob chooses for each $p \in P$ a polynomial $b_p \in \mathbb{F}_q[x_1, \ldots, x_n]$. Then the ciphertext computes to

$$c := m + \sum_{p \in P} b_p \cdot p. \tag{1}$$

– To decrypt a ciphertext $c \in \mathbb{F}_q[x_1, \ldots, x_n]$ Alice computes

$$c(\sigma) = m + \sum_{p \in P} b_p(\sigma) \cdot p(\sigma) = m.$$

Of course, the key problem is to choose the public/secret key in such a way that the system is secure. Moreover, as explained in [11, Chap. 5, §6] it may also be advisable to impose some restrictions on Bob's choice of the polynomials $b_p$. In fact, the attacks described in the next section demonstrate that it seems to be rather involved to derive a secure instance of Polly Cracker.

## 3 Attacking the system

### *3.1 Timing polynomial evaluations*

For decrypting a ciphertext $c$ in the Polly Cracker system, Alice evaluates $c$ at her secret key $\sigma$. The choice of the algorithm used here can influence the security of the resulting system significantly. For example, when evaluating dense polynomials by means of Horner's rule, the Hamming weight of the evaluation point influences the number of additions and multiplications that have to be performed during the evaluation. So one can expect the weight of the secret key $\sigma$ to influence the amount of CPU time required for evaluating a ciphertext.

Of course, in a practical implementation, the occurring multivariate polynomials are usually not dense, and alternate evaluation strategies can be used. Moreover, the complexity of the computations in $\mathbb{F}_q$ or caching phenomena might influence the required execution time.

To demonstrate that in a careless implementation a *timing-side channel* can indeed leak critical information, subsequently we give an example based on the evaluation procedure of the commercial computer algebra system MAGMA (see [6]). In an experimental example, the timing information turned out to be sufficient for revealing the complete secret key.

Now assume that Alice has fixed her secret key $\sigma \in \mathbb{F}_q^n$ and a corresponding public key $P \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$. In a first step we want to identify the Hamming weight of $\sigma$, that is, the number of non-zero entries in $\sigma$. For this we proceed through the following steps:

1. Ideally, in the first (preparative) step we have access to the same kind of platform that is used by Alice for decryption. We produce some ciphertext polynomial $c$, and for each possible weight $0 \le w \le n$ of a secret key we create several random elements from $\mathbb{F}_q^n$ of Hamming weight $w$ and determine the CPU time required for evaluating $c$ at these points.
2. Next, for each $0 \le w \le n$ we use the timings from Step 1 to compute an estimation of the average time $t_w$ required for evaluating $c$ at a point of weight $w$. One may think of different approaches to derive the estimation $t_w$ – in our experiments the following simple procedure worked quite satisfactory: after computing the mean value $\overline{t}_w$ and the standard deviation $s_w$ of all timings obtained for messages of weight $w$, we abandon all timings $t$ with $|t - \overline{t}_w| > 6s_w$. Averaging over the remaining samples yields our estimated encryption time $t_w$.
3. When Alice decrypts the ciphertext $c$, we measure the CPU time $t$ she needs for the evaluation of $c$ at her secret key $\sigma$. Based on $t$ and the reference timings $t_w$ we can derive an estimation for $\sigma$'s Hamming weight. Namely, if there is a unique weight $w$ with $|t - t_w|$ being minimal, we estimate $\sigma$'s weight to be $w$; if $|t - t_w| = |t - t_{w'}|$ for weights $w \neq w'$ in our experiments we estimated $\sigma$'s weight to be $\max(w, w')$. This crude method worked quite satisfactory.

In our experiments with MAGMA it turned out that it is superfluous to perform measurements for each Hamming weight – a much more efficient procedure worked very well: we encrypt the zero message and messages of weight $n$ only. This yields two reference timings $t_0$ and $t_n$, and to interpolate $t_w$ for $0 < w < n$ we suppose the map $\{0, \ldots, n\} \to \mathbb{R}$, $w \mapsto t_w$ to be affine.

Of course, the above idea for estimating the weight of an evaluation point can, in principle, also be applied to the HFE cryptosystem where multivariate polynomials are evaluated at plaintext messages $(m_1, \ldots, m_n) \in \mathbb{F}_q^n$ (cf. [14, 15]). While knowing the weight of an evaluation point alone might not be critical yet in Polly Cracker we can do much better. Instead of estimating only $\sigma$'s weight $w$, we can check whether the $k$th component of $\sigma$ is zero or non-zero:

1. Compute a ciphertext $c := m + \sum_{p \in P} b_p \cdot p$ such that each (non-constant) term[1] contains the variable $x_k$ (e.g., select each $b_p \in \mathbb{F}_q[x_k] \setminus \{0\}$ with $b_p(0) = 0$). So we can expect that Alice's time for decrypting $c$ depends significantly on whether $x_k$ is specialized to zero or to a non-zero element.
2. Next, we need access to the same kind of platform Alice uses for decryption, so that we can evaluate $c$ at several randomly chosen points $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{F}_q^n$ of weight $w$ with $\beta_k = 0$ and at several randomly chosen points $\gamma = (\gamma_1, \ldots, \gamma_n) \in \mathbb{F}_q^n$ of weight $w$ with $\gamma_k \neq 0$.

[1] We adopt the convention that a *term* is a *monic monomial*.

3. As in Step 2 of the previous algorithm (when estimating $\sigma$'s weight) we can use the obtained timings to derive an estimation $t_z$ and $t_{nz}$ for the CPU time required on average for evaluating $c$ at a point of weight $w$ with the $k$th component being zero and non-zero, respectively.
4. Finally, we send the ciphertext $c$ to Alice and measure the CPU time $t$ she requires for evaluating $c$ at her secret key $\sigma$. If $|t-t_z| < |t-t_{nz}|$ then we assume the $k$th component of the secret key to be zero, and in case of $|t-t_z| > |t-t_{nz}|$ we assume the $k$th component of the secret key to be non-zero (in case of $|t-t_z| = |t-t_{nz}|$ we do not know).

As knowing both the number $w$ and the positions of the non-zero entries of $\sigma$ restrains the number of possible secret keys from $q^n$ to $(q-1)^w$, for $q=2$ the above attack should allow us to derive the secret key $\sigma$ without further computations. To verify the latter statement experimentally, we used a SUN SPARCstation-10 with 33 MHz and 128 MB of RAM and MAGMA V2.5-1's built-in `Evaluation` function for evaluating polynomials. The secret key $\sigma$ was chosen from $\mathbb{F}_2^{128}$, and the public key consisted of 128 polynomials from $\mathbb{F}_2[x_1,\ldots,x_{128}]$. Measuring the required CPU time by means of MAGMA's `Cputime` function, the above attack revealed the complete secret key $\sigma$ successfully. For example, for the last bit we had $t_z = 2.12\,\mathrm{s}$, $t_{nz} = 2.1673\,\mathrm{s}$, and evaluating the ciphertext at $\sigma$ took 2.121 s – in accordance with $\sigma$'s last bit being reset.

So for practical implementations of cryptosystems like Polly Cracker, appropriate care for hiding timing information should be taken. An obvious way to avoid our timing attack is to choose $\sigma \in \mathbb{F}_q \setminus \{0\}^n$, i.e., zero entries are not allowed. For obvious reasons the latter approach cannot be applied over the field $\mathbb{F}_2$ which, in other respects, is quite attractive for actual implementations.

Also, it should be pointed out that the availability of CPU timings is just one example of a side channel attack, and the question of protecting polynomial-based schemes against other "physical" attacks like differential power analysis (cf. [12]) arises. For example, when using simple delay loops for masking the "real" CPU time, a simple power analysis might be sufficient to reveal the delay loops. The cryptanalytic relevance of differential power analysis-based attacks for polynomial-based schemes is illustrated by the attacks in [16] on two signature schemes.

We do not want to discuss side channel attacks in more detail here, rather we will continue with more "structural" problems of Polly Cracker instead.

### *3.2 Exploiting sparseness for revealing individual messages*

The idea of the linear algebra attacks mentioned in [8, 11] is to reconstruct the polynomials $b_p$ that Bob has used to derive the given ciphertext (1). However, as the plaintext $m \in \mathbb{F}_q$ is a constant, we have

$$m = c(0) - \sum_{p\in P} b_p(0)\cdot p(0), \tag{2}$$

that is, to reveal the plaintext $m$ it is sufficient to determine the constant terms $b_p(0)$ of all those $b_p$'s where $p(0) \neq 0$. If Bob does not pay enough attention to the encryption process, these constants can often be derived very easily from the ciphertext and Alice's public key.

As there are $\binom{n+d}{d}$ terms of total degree $\leq d$ in $n$ indeterminates (e.g., [7, Chap. 9, §2, Lemma 4]), for a reasonable key size we can assume that Alice's public polynomials $P$ are not dense, and with some luck each $p \in P$ contains a term $t_p$ that does not occur in any element from $P \setminus \{p\}$ (the situation where no such "characteristic term" exists will be discussed later). Denote by $\mathrm{T}(\cdot)$ the function that yields the set of terms occurring in a polynomial with non-zero coefficient. Now, if the terms $t_p$ and Bob's polynomials $b_p$ satisfy the condition[2]

$$t_p \notin \bigcup_{p'\in P} \mathrm{T}(p')\cdot\mathrm{T}(b_{p'} - b_{p'}(0)) \quad (p \in P), \tag{3}$$

then the coefficient of $t_p$ in the ciphertext $c$ is just the coefficient of $t_p$ in the public polynomial $p$ multiplied by $b_p(0)$. In other words, we can immediately read off the desired constant terms $b_p(0)$ from the ciphertext and recover the plaintext via equation (2).

Having in mind a somehow realistic encryption procedure, it seems sensible to assume that Bob's polynomials $b_p$ are sparse. So in particular if the terms of these polynomials are chosen at random, we have quite a good chance for condition (3) to hold. A drastic example is the original version of ENROOT (cf. [9]), which can be regarded as a special case of Polly Cracker. Here all polynomials involved are required to be "highly" sparse and the polynomials used for encryption are chosen at random. Consequently, the plaintext can often be revealed easily (cf. [5]). But also the general Polly Cracker cryptosystem is not immune against this kind of attack.

*Example 1.* In the *Graph 3-Coloring* instance of Polly Cracker (see [8, Example 1] and [11, Chap. 5, §3, Example 3.1]) the public polynomials split into two subsets: the first of these subsets consists of the linear polynomials

$$p_v := x_{v1} + x_{v2} + x_{v3} + 1 \in \mathbb{F}_2\,[\cup_{v\in V}\{x_{v1}, x_{v2}, x_{v3}\}] \quad (v \in V) \tag{4}$$

where $V$ is the vertex set of some graph, and the remaining public polynomials involve only terms of total degree two and their constant term is zero. So it is sufficient to recover the constant terms of those of Bob's

[2] As usual, for $A, B \subseteq \mathbb{F}_q[x_1,\ldots,x_n]$ we write $A\cdot B$ for $\{a\cdot b : a \in A, b \in B\}$.

polynomials that are multiplied with a polynomial of the form (4). For this, each of the variables $x_{v1}$, $x_{v2}$, $x_{v3}$ can serve as a "characteristic term" of $p_v$. The only way to "eliminate" these characteristic terms is to include linear terms in Bob's polynomials $b_{p_v}$ ($v \in V$). In particular, selecting the terms of the $b_{p_v}$'s at random results in a ciphertext which can immediately be decrypted with the method described above with very high probability. It should be mentioned here that this attack does not apply to Koblitz's fine-tuned version where the polynomials $b_{p_v}$ are chosen through "directed randomness" (see [11]). We will address this fine-tuned version in Sect. 3.3.

Now let $Q \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$ be an arbitrary set of polynomials generating a proper ideal $\langle Q\rangle \subsetneq \mathbb{F}_q[x_1, \ldots, x_n]$ which contains Alice's public polynomials $P$, i.e., for all $p \in P$ we have $p \in \langle Q\rangle$. Then for a given ciphertext (1) there exist polynomials $\tilde{b}_{\tilde{q}} \in \mathbb{F}_q[x_1, \ldots, x_n]$ such that

$$c = m + \sum_{\tilde{q}\in Q} \tilde{b}_{\tilde{q}} \cdot \tilde{q}.$$

In other words, if we know the generating set $Q$ and those $\tilde{b}_{\tilde{q}}(0)$ with $\tilde{q}(0) \neq 0$, we are able to recover the plaintext message $m$ analogously as in equation (2). Consequently, a "good" encryption procedure for Polly Cracker has to ensure the following: no "characteristic terms" in the public key can be used for recovering the plaintext, and the attacker cannot find a suitable subset $Q \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$ for mounting such an attack.

In fact, the situation is even more involved: assume that there is a small set of terms $X \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$ (say less than a few thousand) such that each of Alice's public polynomials $p$ is of the form

$$p = \alpha_{p0} + \sum_{x^\nu \in X} \alpha_{p\nu} \cdot x^\nu \tag{5}$$

with $\alpha_{p\nu} \in \mathbb{F}_q$. Although we cannot assume to find "characteristic terms" in this situation, it is possible to adapt the above attack: for this, we assume w.l.o.g. the public polynomials $P$ to be $\mathbb{F}_q$-linearly independent. If this assumption does not hold, we can pass from $P$ to a maximal $\mathbb{F}_q$-linearly independent subset of $P$, as such a subset generates the same ideal in $\mathbb{F}_q[x_1, \ldots, x_n]$ as $P$ does.

Now, if for all of Bob's polynomials $b_p$ the condition

$$\big(\mathrm{T}(b_p - b_p(0)) \cdot X\big) \cap X = \emptyset \quad (p \in P) \tag{6}$$

holds, then the coefficient $c_\nu$ of $x^\nu \in X$ in the ciphertext computes to $\sum_{p\in P} \alpha_{p\nu} \cdot b_p(0)$, and we obtain a linear system of equations for the constant coefficients $b_p(0)$. Namely, for $P = \{p_1, \ldots, p_r\}$, $X = \{x^{\nu_1}, \ldots, x^{\nu_s}\}$ we have

$$\begin{pmatrix} \alpha_{p_1\nu_1} \ldots \alpha_{p_r\nu_1} \\ \vdots \quad\quad \vdots \\ \alpha_{p_1\nu_s} \ldots \alpha_{p_r\nu_s} \end{pmatrix} \cdot \begin{pmatrix} b_{p_1}(0) \\ \vdots \\ b_{p_r}(0) \end{pmatrix} = \begin{pmatrix} c_{\nu_1} \\ \vdots \\ c_{\nu_s} \end{pmatrix}. \tag{7}$$

The coefficient matrix on the left-hand side of equation (7) is of rank $r = \mathrm{card}(P)$, because by evaluating the polynomials in $P$ at Alice's secret key we see from equation (5) that

$$\alpha_{p0} = -\Big(\sum_{x^\nu \in X} \alpha_{p\nu} \cdot x^\nu\Big)(\sigma) \quad (p \in P).$$

Hence, if the columns in the coefficient matrix of equation (7) were linearly dependent we also had a linear dependence among the public polynomials $P$ – a contradiction. Therefore, applying Gauß' algorithm to equation (7) yields a unique solution for the constant terms $b_p(0)$, and we can recover the plaintext via equation (2).

Of course, the question arises what to do if neither "characteristic terms" exist nor condition (6) holds. For the special case of the ENROOT system an approach of this kind has been proposed in [3], but the attacks of Bao et al. in [4] demonstrate that the "tame collisions" occurring there are not sufficient for saving that system: the construction used in [3] did not prevent the attacker from being able to determine a small set of candidates for the terms occurring in Bob's polynomials $b_p$. Knowing such a set, a simple linear algebra-based attack is sufficient to recover the plaintext $m$.

To preclude linear algebra-based attacks, one can try to introduce some "hidden terms" in the encryption procedure as described in the quote of H. W. Lenstra in [11, Chap. 5, §6]. However, the "differential" attack from [10] demonstrates that the use of "hidden terms" still does not guarantee the security of the resulting system. The latter attack takes quotients of monomials in Alice's public key into account and aims at recovering such hidden terms as mentioned above. After preprocessing a ciphertext with such a "differential" attack, a linear algebra-based attack might become feasible again.

In summary, fixing a public key and a "good" encryption procedure for Polly Cracker seems to be quite involved. One attempt for deriving a secure instance of Polly Cracker, including a special encryption procedure, has been proposed in [11, Chap. 5, §7]. Unfortunately, the next section demonstrates that for the parameters considered in this instance, it is often feasible to reveal the secret key within a few minutes.

### *3.3 Revealing the secret key*

In [11, Chap. 5, §7] Koblitz suggests to base Polly Cracker on the so-called *Graph Perfect Code* problem with the public key being derived from a suitable 3-regular graph. For this instance of Polly Cracker, Koblitz describes an accompanying fine-tuned encryption procedure that is designed to withstand the kind of linear algebra attack mentioned in the introduction.

Using MAGMA we did some experiments with the suggested parameter sizes. During these experiments it turned out that, besides problems concerning efficiency,

with the specified parameters there are non-negligible problems in the generation of a secure public key-private key pair. To explain these problems we shortly recall how Alice derives a public key-private key pair (for more details see [11, Chap. 5]):

1. Alice starts with a(n undirected) 3-regular graph $G = (V, E)$ containing a *perfect code*. This means there is a subset $\Sigma \subseteq V$ such that for each $v \in V$ there is exactly one $u \in \Sigma$ with $v \in \mathrm{N}(u)$ – here
$$\mathrm{N}(u) := \{w \in V : \ w = u \text{ or } \{u, w\} \in E\}.$$
Koblitz suggests the graph to have $n \approx 500$ vertices (so $\Sigma$ contains $n/4 \approx 125$ elements).
2. Now her secret key is $\sigma := (\chi(v))_{v \in V} \in \mathbb{F}_2^n$ where $\chi(v) = 1$ if $v \in \Sigma$ and $\chi(v) = 0$ otherwise.
3. The public key consists of the following two sets of polynomials (with common zero $\sigma$):
$$B_1 := \Big\{1 - \sum_{u \in \mathrm{N}(v)} x_u : \ v \in V\Big\} \subseteq \mathbb{F}_2[\{x_v : v \in V\}], \quad (8)$$
$$B_2 := \Big\{x_u x_w : \ u, w \in \mathrm{N}(v), u \neq w, v \in V\Big\} \subseteq \mathbb{F}_2[\{x_v : v \in V\}].$$

The key point in the construction of a public key-private key pair is the choice of the graph $G$. Unfortunately, in [11, Chap. 5, §7] neither a concrete algorithm for doing so nor an example is given. An obvious and also a desirable method for deriving a key pair is to use a graph that has been constructed at random by a procedure as described in [11, answer to Ex. 9 of Chap. 5, §3]. However, in our experiments it turned out that a random construction does not yield acceptable key pairs. Of course, this does not show that the *Graph Perfect Code* problem is, in general, easy. For example, the NP-hard problem of actually finding a vertex coloring with three colors of a graph of chromatic number three also turns out to be "easy" for random graphs (cf. e.g., [1] and the references given there). Nevertheless our results give computational evidence that deriving cryptographically useful (hard) instances of Graph Perfect Code is not as easy as one could be tempted to hope:

For our experiments we used MAGMA to derive at random 3-regular (connected) graphs with $n = 500$ vertices containing a perfect code. In all our examples (some 1000) after computing a reduced lexicographic Gröbner basis of the ideal $\langle B_1\rangle \subseteq \mathbb{F}_2[\{x_v : \ v \in V\}]$ spanned by the *linear* polynomials (8) – in other words after computing a reduced row echelon form – no more than five "free parameters" were left to be determined. In other words, to reveal the private key it was sufficient to check through which of the $\leq 2^5 = 32$ possible specializations of these free parameters we obtain a common zero of the public key.

We want to mention a refinement of this astonishingly simple approach: consider the zero-dimensional ideal $\langle B\rangle \subseteq \mathbb{F}_2[\{x_v : \ v \in V\}]$ generated by $B := B_2 \cup \{x_v^2 - x_v : v \in V\} \cup B_3$ where $B_3$ contains those elements of a reduced Gröbner basis of $\langle B_1\rangle$ which involve no more than two monomials (necessarily of total degree $\leq 1$). Then applying Buchberger's algorithm to $B$ produces only monomials and binomials (sums of two monomials), because by definition the S(yzygy)-polynomial of binomials can never involve more than two terms (see, e.g., [7, Chap. 2, §6, Definition 4]).

As also all occurring coefficients are contained in $\mathbb{F}_2$, one can hope to be able to compute a reduced Gröbner basis of $\langle B\rangle$ and to obtain in this way new linear polynomials in the indeterminates $x_v$. Such linear polynomials can then be added to the already known linear conditions (8) with the aim of reducing the number of free parameters. As an example, we applied this technique to a graph where the original linear equations allowed five free parameters: after adding the additional linear polynomials only one free parameter was left. For an instance with three free parameters we could determine the secret key uniquely.

## 4 Conclusion

We described attacks against both the general Polly Cracker cryptosystem and the fine-tuned variant described by Koblitz.

The attacks demonstrate that there are non-negligible problems in deriving secure public keys and encryption procedures for Polly Cracker; the timing attack might also be of interest for other cryptosystems that rely on the evaluation of polynomials, like HFE. For a more detailed analysis it would be helpful to have a concrete public key and some ciphertext available, so that the effectiveness of the proposed attacks can be examined against a concrete instance of Polly Cracker. Unfortunately, neither [8] nor [11] give an example or a specification how to generate such an example. Therefore it remains unclear whether it is possible to derive instances of Polly Cracker which are of practical relevance.

## References

1. Alon N, Kahale N (1997) A spectral technique for coloring random 3-colorable graphs. SIAM J Comput 26:1733–1748
2. Banks WD, Lieman D, Shparlinski IE (2000) An identification scheme based on sparse polynomials. In: Imai H, Zheng Y (eds) Proceedings of Third International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2000. Lecture Notes in Computer Science, vol 1751. Springer, Berlin Heidelberg New York, pp 68–74
3. Banks WD, Lieman D, Shparlinski IE, To VT (2001) Cryptographic applications of sparse polynomials over finite rings. In: Won D (ed) Proceedings of Third International Conference on Information Security and Cryptology – ICISC 2000. Lecture Notes in Computer Science, vol 2015. Springer, Berlin Heidelberg New York, pp 206–220
4. Bao F, Beth T, Deng RH, Geiselmann W, Schnorr C, Steinwandt R, Wu H (2000) Cryptanalysis of SPIFI II and ENROOT II. In: Security through Analysis and Verification,

Dagstuhl Seminar No. 00501, no. 294 in Dagstuhl Seminar Report (abstract), p. 7
5. Bao F, Deng RH, Geiselmann W, Schnorr C, Steinwandt R, Wu H (2001) Cryptanalysis of two sparse polynomial based public key cryptosystems. In: Kim K (ed) Proceedings of 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001. Lecture Notes in Computer Science, vol. 1992. Springer, Berlin Heidelberg New York, pp 153–164
6. Bosma W, Cannon J, Playoust C (1997) The Magma algebra system I: the user language. J Symbolic Comput 24:235–265
7. Cox D, Little J, O'Shea D (1992) Ideals, varieties and algorithms: an introduction to computational algebraic geometry and commutative algebra. Undergraduate Texts in Mathematics. Springer, New York Berlin Heidelberg
8. Fellows M, Koblitz N (1994) Combinatorial cryptosystems galore! In: Mullen GL, Shiue PJ-S (eds) Finite fields: theory, applications, and algorithms. Contemporary Mathematics, vol 168. American Mathematical Society, pp 51–61
9. Grant D, Krastev K, Lieman D, Shparlinski I (2000) A public key cryptosystem based on sparse polynomials. In: Buchmann J, Høholdt T, Stichtenoth H, Tapia-Recillas H (eds) Coding theory, cryptography and related areas. Proceedings of an International Conference, Guanajuato, Mexico, April 1998. Springer, Berlin Heidelberg New York pp 114–121; at the time of writing available from
`http://www.comp.mq.edu.au/~igor/GKLS.ps`
10. Hofheinz D, Steinwandt R (2001) A "differential" attack on Polly Cracker. To appear in 2002 IEEE International Symposium on Information Theory, ISIT 2002 Proceedings (extended abstract), 2002
11. Koblitz N (1998) Algebraic aspects of cryptography. With an appendix on hyperelliptic curves by Alfred J. Menezes, Yi-Hong Wu, and Robert J. Zuccherato. Algorithms and Computation in Mathematics, vol 3. Springer, Berlin Heidelberg New York
12. Kocher P, Jaffe J, Jun B (1999) Differential power analysis. In: Wiener M (ed) Advances in Cryptology – CRYPTO '99. Lecture Notes in Computer Science, vol 1666. Springer, Berlin Heidelberg New York, pp 388–397
13. Kocher PC (1996) Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz N (ed) Advances in Cryptology – CRYPTO 1996. Lecture Notes in Computer Science, vol 1109. Springer, Berlin Heidelberg New York, pp 104–113
14. Patarin J (1996) Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In: Maurer U (ed) Advances in Cryptology – EUROCRYPT 1996. Lecture Notes in Computer Science, vol 1070. Springer, Berlin Heidelberg New York, pp 33–48; for an extended version see [15]
15. Patarin J (1996) Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. At the time of writing available at `http://www.univ-tln.fr/~courtois/hfe.dvi`; extended version of [14]
16. Steinwandt R, Geiselmann W, Beth T (2001) A theoretical DPA-based cryptanalysis of the NESSIE candidates FLASH and SFLASH. In: Davida GI, Frankel Y (eds) Information Security, 4th International Conference, ISC 2001 Proceedings. Lecture Notes in Computer Science, vol 2200. Springer, Berlin Heidelberg New York, pp 280–293