Asymmetric Cryptography with S-Boxes

Is it easier than expected to design efficient asymmetric cryptosystems ?

- Extended Version -

Jacques Patarin, Louis Goubin
Bull PTS , 68 route de Versailles - BP 45
78431 Louveciennes Cedex - France
e-mail : {J.Patarin,L.Goubin}@frlv.bull.fr

Abstract

In [12], T. Matsumoto and H. Imai designed an asymmetric cryptosystem, called C^*, for authentication, encryption and signature. This C^* scheme was broken in [13] due to unexpected algebraic properties.

In this paper, we study some new "candidate" asymmetric cryptosystems based on the idea of hiding one or two rounds of small S-box computations with secret functions of degree one or two. The public key is given by multivariate polynomials of small degree. The C^* scheme (when its n_i values are small) can be seen as a very special case of these schemes, but in the new schemes, the algebraic properties of [13] generally do not exist, so that completely different cryptanalytic ideas have to be found. We study the efficiency of classical cryptanalysis (such as differential cryptanalysis), and we also present completeley new cryptanalytic tools (such as "gradient cryptanalysis"). With these cryptanalysis, most of the "new" algorithms can be broken and we deduce some very different cryptanalysis of C^*. Moreover, our cryptanalysis and the cryptanalysis of [13] can also be combined in order to faster compute a cleartext from a ciphertext, and to find more informations on the secret key. Thus one of the interests of the paper is to improve the cryptanalysis of C^*.

However, we were not able to find the cryptanalysis of all the new schemes. More precisely, when one round of secret quadratic functions is combined with one round of S-boxes, or when two rounds of S-boxes are carefully hidden by affine functions, the security of these schemes is surprisingly still an open problem. Another interest of the paper lies therefore in the highlighting of these new schemes. The main practical advantage of these schemes is that secret computations are easy and can be performed in low-cost smartcards.

Key words: New asymmetric algorithms, multivariate polynomials, differential cryptanalysis, efficient asymmetric algorithms for smartcards.

Notes:

- This paper is the extended version of the paper with the same title published at ICICS'97.
- In this extended version, we have taken into account the recent results of [1] and [4].

1 Introduction

In [18], Bruce Schneier wrote : "Any algorithm that gets its security from the composition of polynomials over a finite field should be looked upon with skepticism, if not outright suspicion". Moreover, in the same page, he also wrote : "Prospects for creating radically new and different public-key cryptography algorithms seem dim". Maybe Bruce Schneier is right. Nevertheless we will try in this paper to design new public key cryptosystems that get their security... from the composition of polynomials over a finite field ! Moreover, the design of our schemes seems to be amazingly simple... However, we must say that we have no proof of security for these schemes and we will not be shocked if these schemes are looked upon with skepticism, if not outright suspicion.

We will see that, as expected, the easier schemes are not secure but for some more complex schemes ("two-round schemes"), the security is still an open problem.

It should be noticed that the polynomials in the schemes are multivariate polynomials and that Bruce Schneier's declarations were essentially motivated by results on univariate polynomials. The complexity results may be completely different for multivariate polynomials. For example, solving a univariate polynomial equation of small degree d in a finite field is feasible (the complexity is polynomial in d), but solving a multivariate set of polynomial equations of small degree d in a finite field K is NP-hard, even when $K = GF(2)$ and $d = 2$ (cf [8]). Or finding the functional decomposition of a univariate polynomial is often easy (see [19] and [20]) but finding the functional decomposition of multivariate polynomials seems to have a complexity exponential in the number of variables, even with the best known algorithms (see [5] or [6] p. 86). Moreover the general problem of computing a multivariate polynomial decomposition is NP-hard (see [5] or [6] p. 87).

In fact, these hard problems are well known motivations to try to design new asymmetric cryptosystems with multivariate polynomials. In [7] a first design idea was studied by Harriet Fell and Whitfield Diffie but, as they pointed out, their design was not efficient because their function F and its inverse F^{-1} were multivariate polynomials with the same degree d. In [12] Tsutomu Matsumoto and Hideki Imai designed a very efficient scheme (called C^*) with a function F of total degree two, such that the degree of F^{-1} was much larger than two. However this scheme was broken in [13] due to unexpected algebraic properties. In [14] another scheme, called HFE, was designed by Jacques Patarin to avoid these unexpected algebraic properties, but the secret key computations of HFE are not as efficient as in the original Matsumoto-Imai scheme C^*.

The aim of this paper is to introduce and to study "candidate" schemes, whose interest lies in their very simple design, and in the very good efficiency of the secret key computations. More precisely, the secret key computations will be easy in low-cost smartcards because very little RAM is needed and because the number of computations to be performed is very moderate. The main idea of those schemes is to use small S-boxes where some random multivariate functions of small degree are stored, and to combine such a function with some secret multivariate functions of small degree. The public key is sometimes large, but its length is still polynomial in the length of the messages (moreover, we can expect that secret key computations are performed on smartcards, and public key computations are performed on personal computers).

An important part of the paper deals with the efficiency of very different cryptanalytic ideas on these schemes, such as differential cryptanalysis, canonical representation of multivariate polynomials of degree two, or affine multiple attacks. Moreover we introduce some new cryptanalytic ideas, such as for example what we have called "gradient cryptanalysis". A new attack on Matsumoto and Imai's C^* cryptosystem is also described: we will see that it is always possible to "separate" the "branches" of the scheme from each other, whatever their size may be. We will like this be able to recover more of the secret key than in [13], and to faster compute cleartexts from ciphertexts.

Although the easier variations of our schemes can be broken by using those ideas, some of our schemes seem to resist cryptanalysis so far, and may therefore be interesting candidates for new and efficient asymmetric cryptosystems.

Part I
Cryptanalysis of One Round Schemes

2 One round of S-boxes: description of the schemes

Aim of this section

In section 2, we introduce the basic principles of the new schemes we deal with in this paper. As we pointed out above, transformations which are based on algebraic structures can often be used by the cryptanalyst to break the scheme. That is the reason we will use S-boxes, which are supposed to be built with randomly chosen polynomials of small degree, and thus have no algebraic structure. Moreover, the design of the scheme will be such that the encryption function is polynomial with a small degree (typically 2 or 4), but its inverse is polynomial with a high degree, and so cannot be found by Gaussian reductions.
Section 2 is about the simplest way of using these ideas: there will be only one round of S-boxes, hidden by two affine bijective transformations. However, we will see (in section 3) that this first scheme is not secure, whatever the degree of the S-boxes may be.

Description of the schemes

A finite field K, with $q = p^m$ elements is public. For example, we may take $K = GF(2)$, the finite field with two elements.
Each message will be represented by n elements of $GF(q)$, where n is another public integer. n is split into d integers $n_1, ..., n_d$, such that $n = n_1 + ... + n_d$. For each e, $1 \leq e \leq d$, we have a S-box S_e which maps K^{n_e} into K^{n_e}, and is given by quadratic polynomials of the n_e variables.
The secret items will be:

1. Two affine bijections s and t from K^n to K^n.
2. The separation of n into d integers: $n = n_1 + ... + n_d$.
3. The S-boxes: $S_1, ..., S_d$.

Let $x = (x_1, ..., x_n)$ be the cleartext, and $y = (y_1, ..., y_n)$ be the corresponding ciphertext. If we have the secrets, then we can obtain y from x as follows (we will see below how to obtain y from x without the secrets):

1. $x = (x_1, ..., x_n)$ is first transformed with the affine secret permutation s into $s(x) = a = (a_1, ..., a_n)$.
2. Then we compute
$$\begin{cases} (b_1, ..., b_{n_1}) = S_1(a_1, ..., a_{n_1}) \\ \quad \cdots\cdots\cdots\cdots\cdots \\ (b_{n_1+...+n_{d-1}+1}, ..., b_{n_1+...+n_d}) = S_d(a_{n_1+...+n_{d-1}+1}, ..., a_{n_1+...+n_d}) \end{cases}$$
where for each e, $1 \leq e \leq d$, S_e is the secret quadratic S-box which maps K^{n_e} into K^{n_e}.
3. Finally, $b = (b_1, ..., b_n)$ is transformed with another affine secret permutation t, into $t(b) = y = (y_1, ..., y_n)$.

It is easy to see that the composition of all these operations is still a quadratic function of the components of x, so that it can be given by n polynomials of degree two in $x_1, ..., x_n$. Each S-box can be seen as a "branch" of the algorithm.
The public items are:

1. The field K and the length n of the messages.
2. The n polynomials P_1,...,P_n in n variables over K.

So anyone can encipher a message (because from $P_1, ..., P_n$, anyone can obtain y from x).
Moreover, if the secret items are known, a message can be deciphered like this:

1. From $y = (y_1, ..., y_n)$, we compute $b = (b_1, ..., b_n) = t^{-1}(y_1, ..., y_n)$.
2. Then we can find $(a_1, ..., a_n)$ by looking in a table in which all the values of the S-box functions have been precomputed and sorted in lexicographic order.
3. Finally, we compute $x = (x_1, ..., x_n) = s^{-1}(a_1, ..., a_n)$.

Note 1: The parameters must be chosen so that the tables for the S-boxes are not too large. For example, we can take $K = GF(2)$, $n = 64$ and $n_1 = ... = n_8 = 8$, so that each table contains $2^8 = 256$ values.

Note 2: The S-boxes are not necessarily one-to-one maps. So there may be several possible cleartexts for a given ciphertext. One solution to avoid this ambiguity is to put some redundancy in the representation of the messages, by making use of an error correcting code or a hash function (for details, see [14] p. 34, where a similar idea is used in a different scheme).

Note 3: The scheme can also be used in signature. To sign a message M, the basic idea is to compute x from $y = h(R||M)$ (as if we were deciphering a message), where h is a hash function and R is a small pad. If we succeed, (x, R) will be the signature of M. If we do not succeed (because the function is not a bijection), we try another pad R (for variants and details, see [14], where a similar idea is used). It could also be noticed that, with the variation described below (with some quadratic functions q_1, ..., q_d), we will have a larger probability to suceed (*i.e.* we will have to try fewer values R).

Note 4: We said that the S-box can be inverted by looking in a table. More precisely, we always have three different ways to invert a S-box:

1. Looking in a table.
2. Trying all the possible inputs in order to find the right ones ("exhaustive search").
3. Solving the polynomial equations of the S-box. Since the S-box uses only a very small number of variables, these equations can often be inverted by algebraic algorithms (which are not efficient with more variables).

Note 5: The C^* scheme of T. Matsumoto and H. Imai can be seen as a very special case of our algorithm above, when, in C^*, all the n_i variables are small (see [12] for the definition of C^* and of the n_i variables). Therefore, the attacks of the next three sections can be applied against C^* with small n_i parameters, so that we obtain a different cryptanalysis of that scheme. Moreover, this new cryptanalysis is based on the fact that the n_i's are small, and not on algebraic properties of the tranformations, such as in the general cryptanalysis of [13].

Variation 1 (General S-boxes scheme): The output of the S_i box ($2 \leq i \leq d$) can also be XORed with a quadratic function q_i of the variables $(a_1, a_2, ..., a_{n_1+...+n_{i-1}})$ before performing the t permutation, as shown in figure 1 below. The public key is still a quadratic function of the x_j variables, $1 \leq j \leq n$, and the decryption will be done first on S_1 (it gives a_1, ..., a_{n-1}), then on S_2, ..., and finally on S_d (*i.e.* from the left to the right S-boxes). As mentioned in note 3, this variation may have practical advantages in signature. However, the cryptanalysis of this generalisation will be exactly the same as if $q_2 = ... = q_d = 0$.

Variation 2 (Triangular system except one S-box at the beginning): In figure 2, we have a "triangular system except one S-box at the beginning". This variation is "almost" a permutation, so it may have some practical advantages. However, this variation can be broken as the general one round scheme.

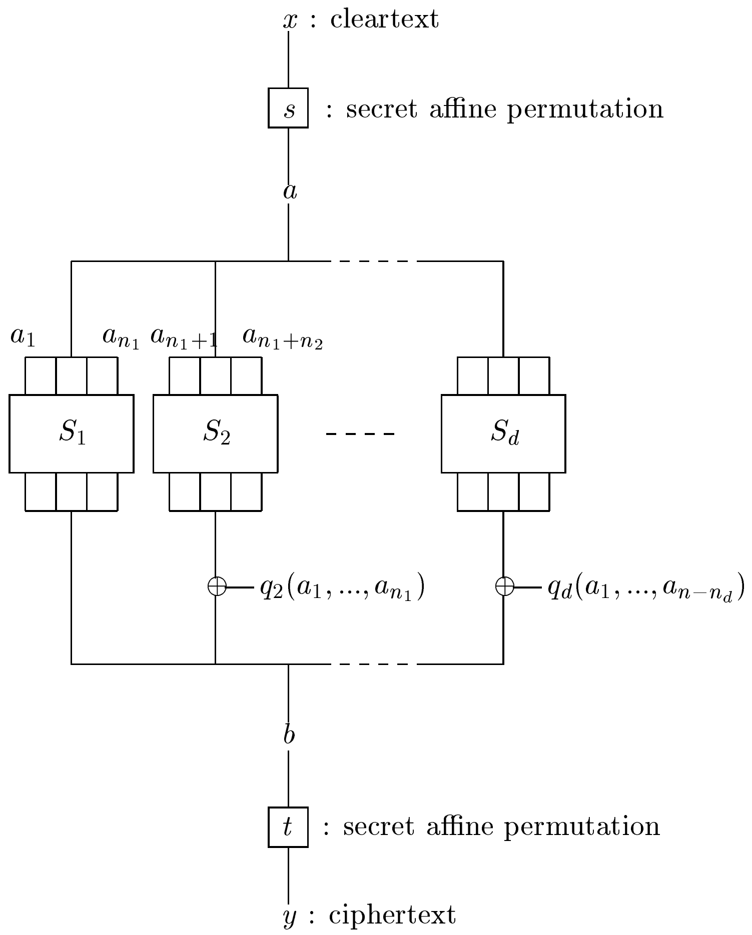

Figure 1: One round of triangular S-boxes (it gives a weak scheme)

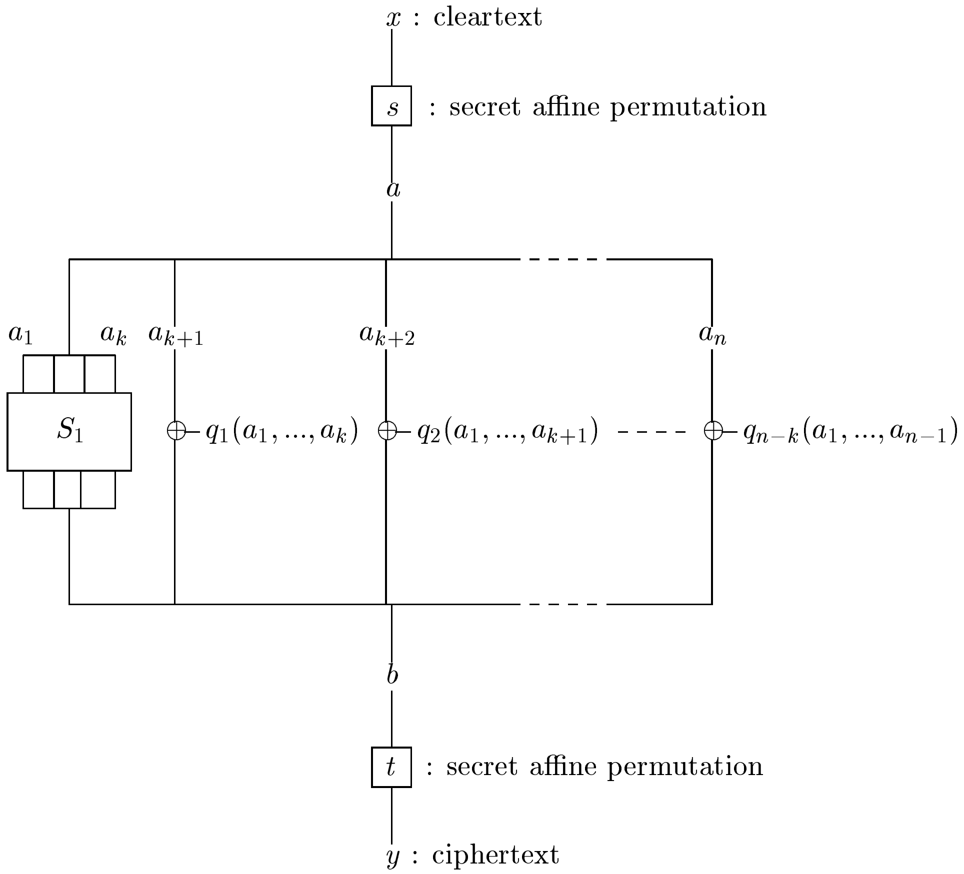

Figure 2: One round of "one S-box followed by a triangular construction" (it gives a weak scheme)

3 One round of S-boxes: Cryptanalysis of the schemes

3.1 The Degeneration Problem (DP)

As we will see below, the cryptanalysis of the one-round schemes is related to the following problem, that we call the "Degeneration Problem" (DP):

Given: A multivariate quadratic equation $y = P(x_1, ..., x_n)$, where P is of degree two, with n variables x_1, ..., x_n.

Problem: Find a change of variables of degree one, from the n variables x_i to (at most) $n-1$ variables x'_1, ..., x'_{n-1} (*i.e.* $\forall i$, $1 \leq i \leq n$, $x_i = P'_i(x'_1, ..., x'_{n-1})$, where P'_i is of total degree one), and find a polynomial Q of total degree two in (at most) $n-1$ variables, such that: $y = Q(x'_1, ..., x'_{n-1})$.

As we will see, there are some algorithms with polynomial complexity for this DP problem, and from them a cryptanalysis of the one-round scheme can be found.

Remark: However, it can be noticed that we did not find an algorithm with polynomial complexity for the following more general problem, that we call the "Linear Combination Degeneration Problem (LCDP)":

Given: A set of k multivariate quadratic equations $y_i = P_i(x_1, ..., x_n)$, where P_i is of degree two.

Problem: Find a linear combination $y = \sum_{i=1}^{k} \alpha_i y_i$ such that, for this linear combination y, the DP problem has a solution.

3.2 First attack: canonical cryptanalysis

In section 3.2, we present an attack based on the existence of a canonical representation of the quadratic polynomials involved in the scheme. Let us recall the classical theorem about the representation of quadratic forms over a field with even characteristic.

Definitions:

1. A *quadratic form* over K is a homogeneous polynomial in $K[X_1, ..., X_n]$ of degree two, or the zero polynomial.
2. Two polynomials f and g of degree ≤ 2 over K, are said to be *equivalent* if f can be transformed into g by means of a nonsingular linear substitution of the indeterminates.
3. A quadratic form f in n indeterminates is called *nondegenerate* if it is not equivalent to a quadratic form in fewer than n indeterminates.

Theorem 3.1 *Let $f \in K[X_1, ..., X_n]$ be a nondegenerate quadratic form over $K = GF(q)$, where q is even. If n is odd, then f is equivalent to:*

$$x_1x_2 + x_3x_4 + ... + x_{n-2}x_{n-1} + x_n^2. \tag{1}$$

If n is even, then f is either equivalent to:

$$x_1x_2 + x_3x_4 + ... + x_{n-1}x_n \tag{2}$$

or to a quadratic form of the type:

$$x_1x_2 + x_3x_4 + ... + x_{n-1}x_n + x_{n-1}^2 + ax_n^2 \tag{3}$$

where $a \in K$.

A proof of this theorem is given in [11]. Moreover, if f is given, then the transformation of f in (1), (2) or (3) is very easy.
The algorithm used in the proof can also be applied to a degenerate quadratic form, in the same way. For such a form f, we can define the "number of independent variables of f" as the smallest integer k such that f is equivalent to a quadratic form in k indeterminates. Following the algorithm given in the proof of the theorem, we can see that this number k is also very easy to compute.
Using these tools, we can now derive a method to "separate" the S-boxes in the scheme. We use the notations of section 2.

Description of the algorithm:

1. Generate randomly n bits $\alpha_1, ..., \alpha_n$, and compute:

$$y' = \sum_{i=1}^{n} \alpha_i P_i(X_1, ..., X_n),$$

which is a polynomial of degree two in $X_1, ..., X_n$.

2. Compute the number k of independent variables of y', and write y' as a polynomial of degree two in $x'_1, ..., x'_k$, where the x'_i have an affine expression in the x_j $(1 \leq j \leq n)$.

The probability that y' has no components in branch number one, *i.e.* the probability that y' has a linear expression in $b_{n_1+1}, ..., b_{n_1+...+n_d}$ is $\frac{1}{q^{n_1}}$.
So by repeating steps 1 and 2 about λq^{n_1} times, we will generate about λ equations $y'_1, ..., y'_\lambda$ such that these equations have no components in branch number one. Moreover, when this occurs, then $k \leq n - n_1$, and when this does not occur, then the probability that $k \leq n - n_1$ is very small, except if there are different branches with about n_1 variables. However, it is very easy to see if two polynomials y'_1 and y'_2 with $k \leq n - n_1$ have no components from the same branch: we write y'_1 in $x'_1, ..., x'_{k_1}$ and y'_2 in $x''_1, ..., x''_{k_2}$, and we see how many terms are linearly independent in $x'_1, ..., x'_{k_1}, x''_1, ..., x''_{k_2}$. If there are more than $n - n_1$ independent terms, then y'_1 or y'_2 has components from the first branch. And if there are less than $n - n_1$ independent terms, then we can write y'_1 and y'_2 as polynomials of degree two in $x'''_1, ..., x'''_{n-n_1}$, where the x'''_i have affine expressions in the x_j $(1 \leq j \leq n)$.

Note: Another idea would be to compute the number k of independent variables of $y'_1 + y'_2$ and test if $k \leq n - n_1$.

Finally, after $O(q^{n_1})$ computations, we will find $n - n_1$ independent equations $y'_1, ..., y'_{n-n_1}$ such that these equations have no components coming from the first S-box, and we will be able to write all these equations as polynomials of degree two in $x'_1, ..., x'_{n-n_1}$, where the x'_i have affine expressions in the x_j $(1 \leq j \leq n)$.
So with these new variables x'_i and y'_i, we have eliminated the variables of the first branch. Applying this algorithm d times, we can "separate" the S-boxes from each other and then the scheme is very easy to break by simple exhaustive search on the cleartext for each S-box.

3.3 Second attack: differential cryptanalysis

We will now describe another very different attack, that also works very well against schemes with small branches. Let $F = (P_1, ..., P_n)$ the function described in section 2. F is seen as a function from K^n to K^n. Let ψ be an element of K^n. Let $X_1, ..., X_k$ be k random plaintexts ($k = n$ typically), and $Y_1, ..., Y_k$ be the corresponding ciphertexts, *i.e.* $Y_i = F(X_i)$ for $i = 1, ..., k$. Let Y'_i be the ciphertext of the plaintext $X_i + \psi$, *i.e.* $Y'_i = F(X_i + \psi)$ for $i = 1, ..., k$. Finally, let $\psi'_i = Y'_i - Y_i$ (where the addition $+$ and the subtraction $-$ are of course done component by component in K).

Definition of test (A): We will say that "ψ satisfied the test (A)" if the vector space generated by all the values ψ'_i is not K^n itself by an affine subspace (the dimension of this subspace will be $n - \lambda$, where λ is one of the n_i values).

It is very easy to see if a value ψ satisfies this test (A) or not, since k is very small ($k = n$ typically). Moreover, we have this theorem:

Theorem 3.2 *The probability that ψ satisfies the test (A), when ψ is randomly chosen, is $\geq \frac{1}{q^{\max(n_i)}}$.*

Proof: The probability that the value $A_1 = (a_1, ..., a_{n_1+1})$ is the same for X_i and $X_i + \psi$ is $\frac{1}{q^{n_1}}$ (because we have q^{n_1} possible values for A_1). We write this relation as $A_1(X_i) = A_1(X_i + \psi)$.
Moreover, since s is an affine function, if the A_1 value is the same for one pair $(X, X + \psi)$ for a specific value X, then for all values X', we will have $A_1(X') = A_1(X' + \psi)$. So, with a probability $\frac{1}{q^{n_1}}$, we have:

$$\forall X_i \in K^n,\ B_1(X_i) = B_1(X_i + \psi).$$

Now, since the transformation t is affine, we see that all the values $\psi'_i = Y'_i - Y_i = F(X_i + \psi) - F(X_i)$ belong to an affine subspace of dimension $n - n_1$. We can now describe our new attack. We proceed in five steps.

1. We find about n values ψ that satisfy the test (A), by randomly trying some ψ values.
2. We say that two such values ψ_1 and ψ_2 "belong to the same branch" if $\psi_1 + \psi_2$ (and more generally $\lambda_1\psi_1 + \lambda_2\psi_2$, $\lambda_1 \in K$, $\lambda_2 \in K$) also satisfies the test (A). We detect and group together the values ψ_i that belong to the same branches. In this way, we separate the ψ values in d different groups, that we call "branches". We will find about n_1 independent values ψ that come from the first branch, n_2 that come from the second branch, ..., n_d that come from the branch number d.
3. By an affine change of variables, we change the variables x_i into x'_i such that, with these new variables, the subspace generated by the values ψ of the first (resp. second,..., d^{th}) branch is characterized by: $x'_1 = ... = x'_{n_1} = 0$ (resp. $x'_{n_1+1} = ... = x'_{n_1+n_2} = 0$,..., $x'_{n_1+...+n_{d-1}+1} = ... = x'_{n_1+...+n_d} = 0$).
4. Similarly, by an affine change of variables, we change the variables y_i into y'_i such that, with these new variables, the subspace generated by the values ψ' of the first (resp. second,..., d^{th}) branch is characterized by: $y'_1 = ... = y'_{n_1} = 0$ (resp. $y'_{n_1+1} = ... = y'_{n_1+n_2} = 0$,..., $y'_{n_1+...+n_{d-1}+1} = ... = y'_{n_1+...+n_d} = 0$).
5. At this point, all the branches have been detected and isolated. To complete the attack, we can now perform an exhaustive search on each input of each of the d branches.

Note: In the second attack, we did not use the fact that the S-boxes are polynomial functions of degree two in a basis. So the attack will work in the same way whatever the functions in the S-boxes may be.

3.4 Third attack: gradient cryptanalysis

We now introduce a new tool to attack the scheme with S-boxes. The general principle of the algorithm is exactly the same as in the first attack that we called "canonical cryptanalysis". It is based on the fact that, with a rather high probability, a linear combination $Q = \sum_{i=1}^{n} \alpha_i P_i(x_1, ..., x_n)$ of the public polynomials has a "number of independent variables" less than n.
In the case of quadratic polynomials, we saw that this "lack" of variables can be detected because a canonical form exists, which enables us to read the "real" number of variables of such a polynomial.
But as soon as the degree is greater or equal than three, we don't have such a canonical representation.
The new idea is to use the so-called *gradient* of Q. For $x = (x_1, ..., x_n)$, it is defined by:

$$\textbf{grad}\ Q(x) = \Big(\frac{\partial Q}{\partial x_1}(x_1, ..., x_n), ..., \frac{\partial Q}{\partial x_n}(x_1, ..., x_n)\Big).$$

The following theorem shows the link between the gradients of Q and the number of independent variables of Q:

Theorem 3.3 *Let Q be a quadratic form in n variables over a field K, then:*
1) If the characteristic of K is > 2, then the number k of independent variables of Q is equal to the dimension of the subspace A generated by all the **grad** *$Q(x)$ ($x \in K^n$).*

2) If the characteristic of K is 2, then it is easy to find a non singular linear substitution of the indeterminates that transforms Q into a quadratic form:

$$R(x'_1, ..., x'_n) = \Phi(x'_1, ..., x'_{\dim A}) + \sum_{i=\dim A+1}^{n} \lambda_i x_i'^2,$$

where Φ is a quadratic form over K. Moreover the number k of independent variables of Q is:

$$k = \dim A + \begin{cases} 0 & \text{if } \forall i,\ \lambda_i = 0. \\ 1 & \text{if } \exists i,\ \lambda_i \neq 0. \end{cases}$$

The rest of the attack is similar to that described in the canonical cryptanalysis. The great difference here is that we can derive the same kind of theorem for cubic forms over a field K, and more generally for forms of any degree over K.

3.5 Fourth attack: linearity in some directions

This attack is probably the simplest of all our attacks.
As above, the attack is based on the fact that, with a rather high probability, a linear combination $Q = \sum_{i=1}^{n} \alpha_i P_i(x_1, ..., x_n)$ of the public polynomials has a "number of independent variables" less than n.
For such a Q, there are some values $d = (d_1, ..., d_n)$, $d \neq 0$, such that:

$$\forall x \in K^n,\ Q(x+d) = Q(x). \qquad (\#)$$

From (#) we will show how to find these values d.
If $x_i x_j$ is a term of $Q(x)$, then in (#) this $x_i x_j$ gives the term: $(x_i + d_i)(x_j + d_j) - x_i x_j$, *i.e.* it gives: $x_i d_j + x_j d_i + d_i d_j$.
Therefore, by writing in (#) that all the terms in x_k, $1 \leq k \leq n$, are zero, we will have n equations of degree one in n variables (there variables are the d_λ variables). So by gaussian reductions, we will find the set of the solutions d, as wanted.
As a result, we have found a break-up into the spaces generated by the S-boxes. By iterating this, we will find the break-up into each of these spaces, and the scheme is very easy to break by simple exhaustive search on the cleartext from each S-box.

3.6 Linear approximations of the S-boxes

A natural idea is of course to also try linear cryptanalysis, by linear approximations of the S-boxes. However, we did not see how to use this idea for an efficient cryptanalysis, even when the S-boxes are public. The problem comes from the fact that s and t are very general affine bijections (much more general than the P transformation.of DES for example), and that they are secret.

3.7 Conclusion

In conclusion, one round of small S-boxes does not give secure algorithms. So, for security, we must have either a transformation that manipulates large values (such a transformation is thus not exhaustively storable in a S-box), or more than one round. We will specifically study the case of small S-boxes with two rounds later in this paper.

4 How to separate large branches in the C^* scheme

As we mentioned in section 2 (see note 5), each of the attacks described in section 3 gives a way to break the C^* scheme of Matsumoto and Imai, when the n_i parameters (which measure the size of the branches) are small. However, this cryptanalysis is deeply based on the fact that the branches are small, and does not give any result in the case of large branches.
Nevertheless, we have found another attack to separate large branches in the Matsumoto-Imai scheme, even if they are large. This attack is described in the extended version of [13] (available from the author).

Part II

Design of Two Round Schemes

5 Complexity of functional decomposition

As we saw in section 3, the scheme which uses only one round of S-boxes is insecure, and we can use several methods to "separate" the branches from each other.
Then a natural question arises: is it possible to design a more secure cryptosystem by using *two* rounds of transformations, each of which is given by polynomials of degree two ? This leads to the following problem:

Decomposition problem: Let g and h be two functions which map K^n into K^n and which are given by polynomials of total degree two in n variables over K. Then $f = g \circ h$ is also a function from K^n to K^n, and it is given by n polynomials of degree *four* in n variables over K. Suppose that f is given. Is it computationally feasible to recover g and h ?

A positive answer to that problem would imply that the two rounds can be easily separated from each other. So, to break the scheme, we would only have to break two independent schemes given by quadratic polynomials in n variables.
That would of course make the idea of using two rounds uninteresting. However, the decomposition of multivariate polynomials was studied by Matthew Dickerson, who gave an algorithm for the following problem:

Multivariate left decomposition: Given polynomials f and h_1,...,h_n in $K[X_1, ..., X_n]$, and an integer r, decide if there exists a polynomial $g(x_1, ..., x_n)$ of total degree at most r that composes with the h_i's to give f. That is, does there exist a polynomial $g(x_1, ..., x_n)$ such that

$$f(x_1, ..., x_n) = g(h_1(x_1, ..., x_n), ..., h_n(x_1, ..., x_n))$$

and deg(g)$\leq r$? If so, determine the coefficients of g.

In [5], Dickerson presents the best-known algorithm for this problem, which is polynomial in the degree of $f, h_1, ..., h_n$, but *exponential* in the number n of variables (note that our decomposition problem is even harder, because the h_i's are not known).
He also shows that the *general* problem of decomposition of multivariate polynomials is difficult, because the following one is NP-hard:

s-1-decomposition problem: Given a monic univariate polynomial $f(x)$ and an integer s, decide if there exists an s-1-decomposition of f, *i.e.* a monic univariate polynomial h of degree s, and a bivariate polynomial $g(y, x) \in K[Y, X]$ of the form $g(y, x) = \prod_{i=1}^{r}(y + \alpha_i x + \beta_i)$ with $\alpha_i, \beta_i \in \hat{K}$, an algebraic extension of K, such that $f(x) = g(h(x), x)$. If so, determine the coefficients of g and h.

There are some reasons to think that the following problem is also NP-hard (cf [5], problem 14, p. 74):

Multivariate decomposition of given degree: Given a polynomial f in $K[X_1, ..., X_n]$ and some subset of the following: integers $k, r, s_1, ..., s_k$, a polynomial $g(x_1, ..., x_k)$$\in K[X_1, ..., X_k]$, and polynomials $h_1(x_1, .., x_n)$,..., $h_k(x_1, ..., x_n)$, decide if there exists a functional decomposition $g, h_1, ..., h_k$ of f (*i.e.* $f = g(h_1, ..., h_k)$) such that deg$g = r$, and deg$h_i = s_i$ for $1 \leq i \leq n$. If so, compute those coefficients of g and the h_i's which were not given.

Dickerson (see [5], p. 75) notices that: "The s-1-decomposition problem seems intuitively easier than problem 14. In problem 14, f, g and h are general multivariate polynomials of arbitrary dimension. Furthermore polynomial g takes the polynomial h_i as arguments, and we know nothing about the form of g other than its degree. In the s-1-decomposition problem, on the other hand, f and h are both univariate polynomials and g is only bivariate. Furthermore, g takes x and not another polynomial as its second argument. We also know a great deal about the structure of the polynomial g, namely

that it factors as: $g(y,x) = \prod_{i=1}^{r}(y + \alpha_i x + \beta_i)$. However, we have tried without success to reduce the s-1-decomposition problem to problem 14."
So, it is still an open problem...

That is the reason why we propose to base a cryptosystem on two rounds of quadratic transformations.

Remark 1: The decomposition problem of this paper consists in finding the decomposition of a multivariate polynomial f (of total degree 4 in n variables x_1, ..., x_n) into two multivariate polynomials g and h, of total degree 2, such that $f = g \circ h$. In a basis, g (and h) are written with about $n \times \frac{n^2}{2}$ coefficients. When we write this equation $f = g \circ h$ in a basis, we obtain about $n \times \frac{n^4}{4!}$ equations in the $2 \times n \times \frac{n^2}{2}$ unknown coefficients. So we have $\mathcal{O}(n^5)$ equations of degree total two in the $\mathcal{O}(n^3)$ unknowns. There is maybe no polynomial time algorithm for this problem (unlike if we had $\mathcal{O}(n^6)$ equations), but since the number of quadratic equations is much larger than the number of unknowns, one can be dubitious about the chances to rely this problem to an NP-complete problem. Moreover, this also shows that, most of the time, there exists "essentially" one decomposition of f (by "essentially", we do not take into account all the obvious solutions than can be deduced from one solution, *i.e.* all the $(g \circ \varphi) \circ (\varphi^{-1} \circ h)$, where φ is an affine permutation of the variables.

Remark 2: In 1999, in the paper [4], an algorithm to find the decomposition of two multivariate polynomials of degree two has been published. However, this algorithm needs all the description of the composition. So, despite the results of this paper [4], we will still be able to design schemes by giving only part of the composition, or by "perturbating" the composition as we will see below.

6 Two rounds: A general description of the schemes ("2R" and "2R$^-$" schemes)

The general scheme will be the following one:
As in section 2, a finite field K, with $q = p^m$ elements, is public, and we want to transform a cleartext $x = (x_1, ..., x_n) \in K^n$ into a ciphertext $y = (y_1, ..., y_n) \in K^n$.
The secret items will be:

1. Three affine bijections r, s and t from K^n to K^n.
2. An application $\phi : K^n \to K^n$ given by n quadratic equations over K.
3. An application $\psi : K^n \to K^n$ given by n quadratic equations over K.

If we have the secrets, then we can obtain y from x as follows:

1. $x = (x_1, ..., x_n)$ is first transformed with the affine secret permutation r into $r(x) = a = (a_1, ..., a_n)$.
2. Then we compute $b = \phi(a)$.
3. $b = (b_1, ..., b_n)$ is then transformed with the affine secret permutation s into $s(b) = c = (c_1, ..., c_n)$.
4. Then we compute $d = \psi(c)$.
5. Finally, $d = (d_1, ..., d_n)$ is transformed with another permutation t, into $t(d) = y = (y_1, ..., y_n)$.

The public items are:

1. The field K and the length n of the messages.
2. Some of the n polynomials $P_1, ..., P_n$ of degree four in n variables over K, such that $y_i = P_i(x_1, ..., x_n)$ $(1 \le i \le n)$. When all these polynomials are given, we call the scheme a "2R scheme". When only some of these polynomials are given, we call the scheme a "2R$^-$ scheme".

So anyone can encipher a message.
Moreover, if the secret items are known, we must be able to decipher a message. For that purpose, we need to invert the functions ϕ and ψ. In practice, ϕ and ψ can be chosen among the following classes of functions:

1. "C^*-functions": monomials over an extension of degree n over K, such as $a \mapsto a^{1+q^\theta}$ (they are the basic transformations in the C^* scheme of Matsumoto and Imai, cf [12]).

2. "Triangular-functions": $(a_1, ..., a_n) \mapsto (a_1, a_2 + q_1(a_1), a_3 + q_2(a_1, a_2), ..., a_n + q_{n-1}(a_1, ..., a_{n-1}))$, where each q_i is a quadratic polynomial in i variables over K (in [7] Fell and Diffie used a particular case of these transformations).

   Note: If $|K|$ is even, then $x \mapsto x^2$ is a bijection on K. So in this case, we can define as "triangular-functions" all the functions $(a_1, ..., a_n) \mapsto (a_1^{\alpha_1}, a_2^{\alpha_2} + q_1(a_1), ..., a_n^{\alpha_n} + q_{n-1}(a_1, ..., a_{n-1}))$, where each q_i is a quadratic polynomial, and where each α_i is either 1 or 2.

3. "S-boxes-functions": they have already been defined in section 2, in the case of a one-round scheme.

4. "One S-box followed by a triangular construction": $(a_1, ..., a_n) \mapsto (b_1, ..., b_n)$, with $(b_1, ..., b_\lambda) = S(a_1, ..., a_\lambda)$ and $b_i = a_i + q_{i-1}(a_1, ..., a_{i-1})$ for $\lambda + 1 \leq i \leq n$. Here, S, q_λ, ..., q_n are quadratic functions, and λ can be seen as the size of the S-box.

5. "Triangular with S-boxes functions", or "General S-boxes functions": $(a_1, ..., a_n) \mapsto (b_1, ..., b_n)$, with:
$$\begin{cases} (b_1, ..., b_{n_1}) = S_1(a_1, ..., a_{n_1}) \\ (b_{n_1+1}, ..., b_{n_1+n_2}) = S_2(a_{n_1+1}, ..., a_{n_1+n_2}) + (q_1(a_1, ..., a_{n_1}), ..., q_{n_2}(a_1, ..., a_{n_1})) \\ \quad \dots\dots\dots\dots\dots\dots \\ (b_{n_1+...+n_{d-1}+1}, ..., b_{n_1+...+n_d}) = S_d(a_{n_1+...n_{d-1}+1}, ..., a_{n_1+...+n_d}) \\ \qquad + (q_{n_2+...+n_{d-1}+1}(a_1, ..., a_{n_1+...+n_{d-1}}), ..., q_{n_2+...+n_d}(a_1, ..., a_{n_1+...+n_{d-1}})). \end{cases}$$
   where the functions S_i and q_i are all quadratic over K.

6. "D^* functions": the definitions and analysis of these D^* functions are the main thema of [16].

Note: It is easy to see that classes number 2, 3 and 4 are just particular cases of class number 5.

The first two have the advantage of being bijections (D^* functions will also be bijective). But, as we will see in sections 7 and 8, all the schemes where ψ is one of the functions of classes 1 and 2, are insecure, whatever we may choose for ϕ. On the contrary, if the second round function ψ is chosen in classes 3, 4, 5 or 6, and the first round function ϕ in classes 1, 2, 3, 4, 5 or 6, the corresponding schemes seem to resist cryptanalysis, so far. We call these schemes the "2R" schemes ("2R" stands fro "two rounds").

7 Cryptanalysis of a second round with C^*

In this section, we study the scheme described in section 6, when ψ is a C^*-function.
A typical example of this situation is the case where ϕ and ψ are both C^*-functions. As was pointed out by Jacques Patarin (cf [15]), this scheme is insecure. Moreover, the cryptanalysis of [15] can be applied even if ϕ is not a C^*-function. Whatever the function ϕ of the first round may be, the basic idea of the attack is to compute all the equations of the form:
$$\sum_{i,j,k} \mu_{ijk} y_i x_j x_k + \sum_{i,j} \nu_{ij} x_i x_j + \sum_{i,j} \xi_{ij} y_i x_j + \sum_i \omega_i x_i + \sum_i \theta_i y_i + \pi_0 = 0$$
and then to introduce some "transition" variables p_i such that these equations can be written:
$$\sum_{i,j} \gamma_{ij} p_i y_j + \sum_i \alpha_i p_i + \sum_i \beta_i y_i + \delta_0 = 0.$$

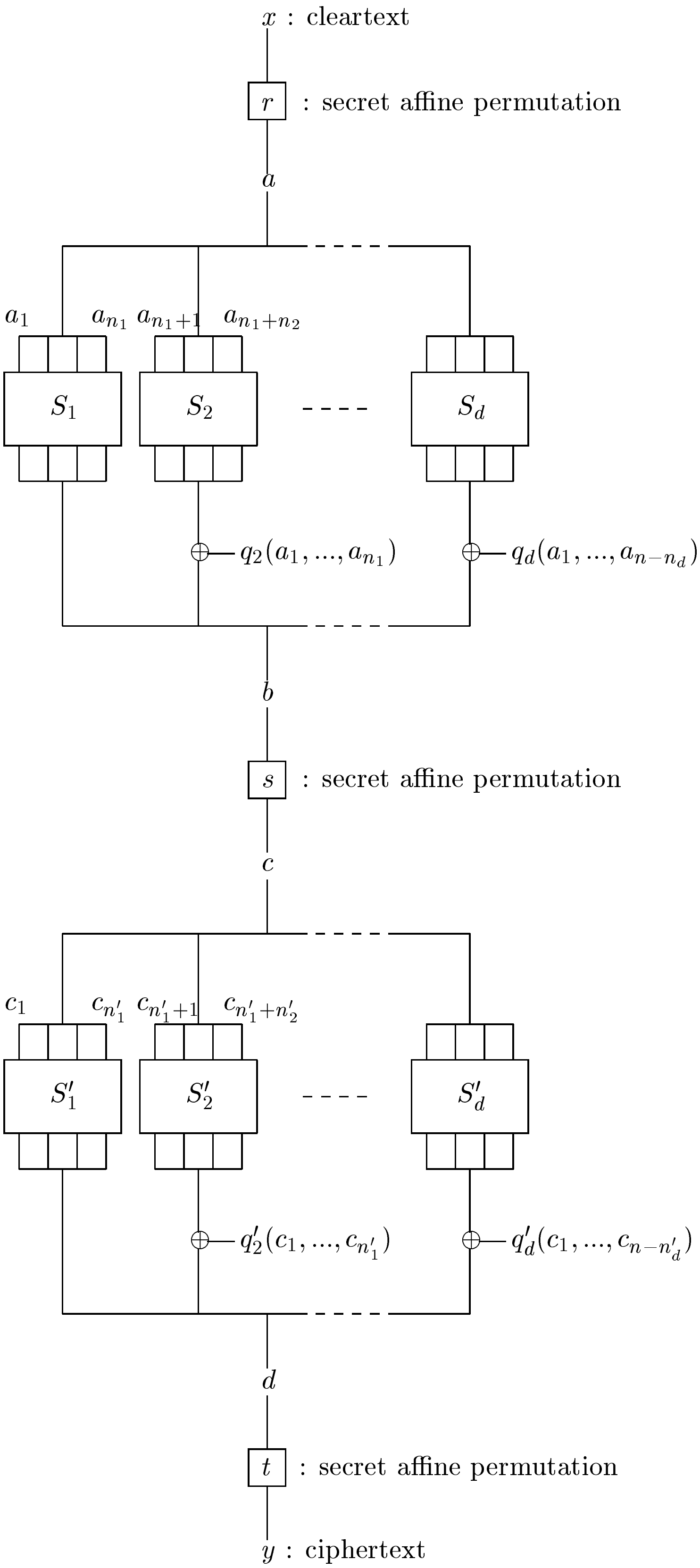

Figure 3: Two rounds of triangular S-boxes (such a scheme may be secure...)

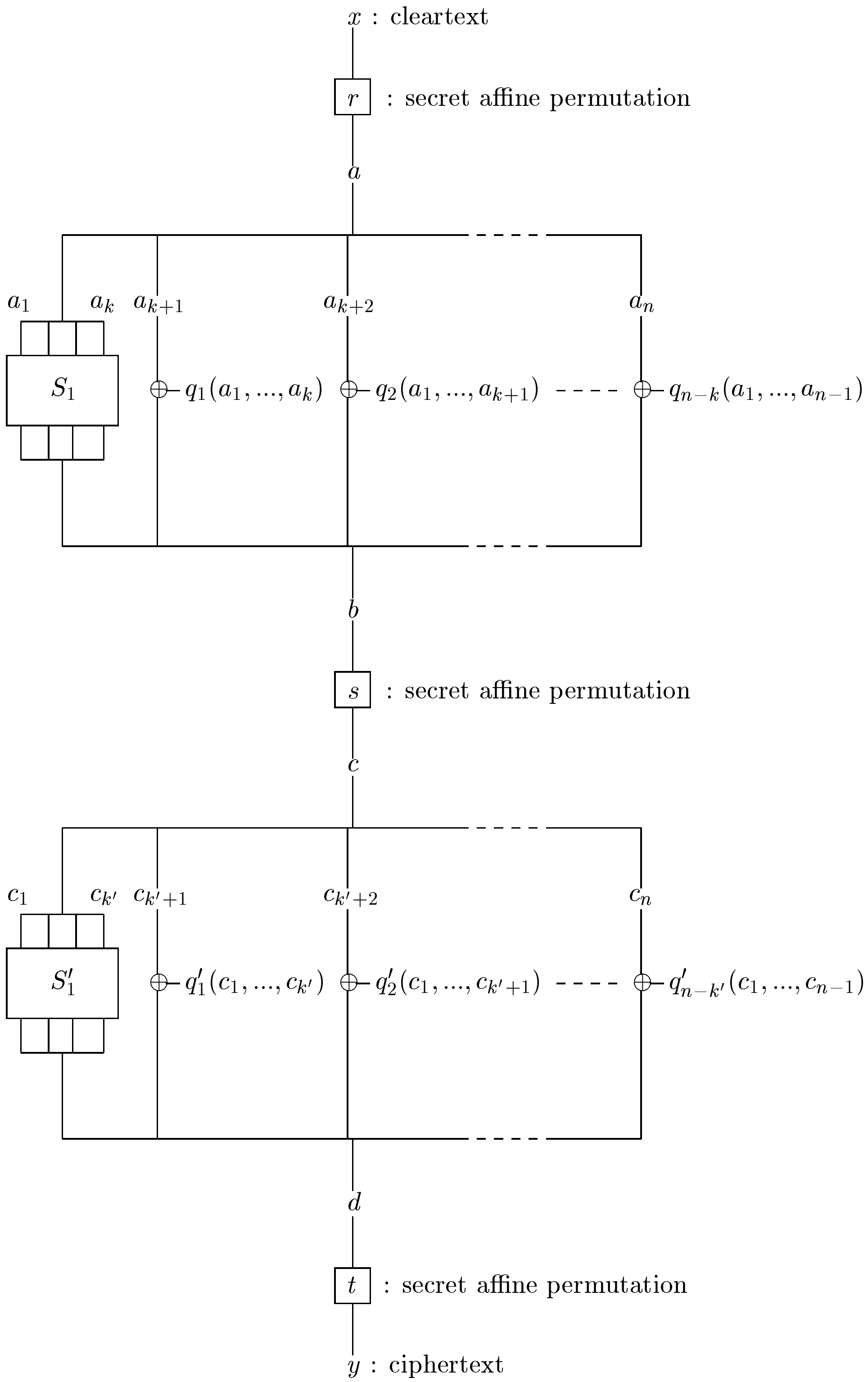

Figure 4: Two rounds of "One S-box followed by a triangular construction" (such a scheme may be secure...)

When the y_i variables are given, then from these equations we can compute the p_j variables by Gaussian reductions. Finally, we can deduce x from the p_j by using:

1. The cryptanalysis of the C^* scheme (cf [13]) if ϕ is a "C^*-function" (this is exactly the situation of [15]).
2. A cryptanalysis similar to the one described in section 8, if ϕ is a "triangular-function".
3. The cryptanalysis of one round of S-boxes (see section 3), if ϕ is a "S-boxes-function".

8 Cryptanalysis of a second round with a triangular-function

Here, we consider the scheme of section 6, when ψ is a "triangular-function". More precisely, we have:

$$(d_1, ..., d_n) = \psi(c_1, ..., c_n) = (c_1, c_2 + q_1(c_1), ..., c_n + q_{n-1}(c_1, ..., c_{n-1})),$$

where $q_1, ..., q_{n-1}$ are quadratic polynomials.
Once more, that scheme is insecure. The key idea is to use the fact that the equation $d_1 = c_1$ gives an equation of degree only 2 in the x_i variables, and that, for each i, $1 \leq i \leq n$, c_i can be obtained from d_i and $c_1, ..., c_{i-1}$. An attack can be derived as follows:

1. By Gaussian reductions on the α_i, β_{ij}, γ_i and δ variables below, find the equations of the form:

   $$\sum_i \alpha_i y_i = \sum_i \sum_j \beta_{ij} x_i x_j + \sum_i \gamma_i x_i + \delta.$$

   The vector space of the solutions has dimension one (because the solutions come from $d_1 = c_1$), so that we obtain $d_1 = \sum_i \alpha_i y_i$ and $c_1 = \sum_i \sum_j \beta_{ij} x_i x_j + \sum_i \gamma_i x_i + \delta$.

2. By Gaussian reductions on the α'_i, β'_{ij}, γ'_i and δ' and on the coefficients of q_1, find the equations of the form:

   $$\sum_i \alpha'_i y_i = \sum_i \sum_j \beta'_{ij} x_i x_j + \sum_i \gamma'_i x_i + \delta' + q_1\Big(\sum_i \sum_j \beta_{ij} x_i x_j + \sum_i \gamma_i x_i + \delta\Big).$$

   The vector space of the solutions has again dimension one (because the solutions come from $d_2 = c_2 + q_1(c_1)$). We thus obtain $d_2 = \sum_i \alpha'_i y_i$ and $c_2 = \sum_i \sum_j \beta'_{ij} x_i x_j + \sum_i \gamma'_i x_i + \delta'$. We also obtain q_1.

By repeating this argument n times, we have finally found $c_1, ..., c_n$ as quadratic polynomials in $x_1, ..., x_n$. What remains is to cryptanalyse the first round, *i.e.* the function $c = s \circ \phi \circ r(x)$, and that is feasible for each of the possible choices for ϕ (see section 3).

9 Second round with S-boxes: description of the schemes

In this section, we present our candidate algorithms. They are all based on the general scheme described in section 6. As we said in section 6, four possibilities exist for the second round function ψ. We choose ψ as a "S-boxes-function", or as "one S-box followed by a triangular construction" or as "S-boxes combined with a triangular construction", or as a "D^* function" (D^* functions are treated in [16] and not in this paper). The algorithms are also differentiated from each other by the choice of the first round function ϕ.
As we said before, six possibilities exist for ϕ:

1. ϕ is a "C^*-function".
2. ϕ is a "triangular-function".
3. ϕ is a "S-boxes-function".

4. ϕ is "one S-box followed by a triangular construction".

5. ϕ is a "Triangular with S-boxes function".

6. ϕ is a "D^* function" (D^* functions are treated in [16] and not in this paper).

Figure 3 illustrates the example with two rounds of "S-boxes combined with a trangular construction". Similarly, figure 4 illustrates the example with two rounds of "one S-box followed by a triangular construction".

In the design of these schemes, the main idea is to hide the quadratic function ϕ (which is weak when used alone, as we saw in section 3) with the second round ψ. For instance, in the case of "C^*-S-box", the algebraic structure of the C^*-functions (which is not completely hidden in the scheme of Matsumoto and Imai, which led to the cryptanalysis of their scheme) seems difficult to recover if it has been mixed with S-boxes, which are supposed to have no special structure. This point will be detailed in section 10.3.
Moreover, as we pointed out, the problem of decomposition of multivariate polynomials seems to be difficult, so probably no general attack exists to "separate" the two rounds from each other.
In conclusion, the security of our schemes is an open problem...

Note: When the first round is a "C^*-function" and the second round is a "S-boxes-function", the scheme may be secure, but it becomes insecure if the two quadratic functions are put the other way round, as we saw in section 7. This shows that the analysis of the security closely depends on the order of the quadratic functions used in the schemes.

10 Remarks about the security of these 2R ("two round") schemes

10.1 The "Quadratic Degeneration Problem" (QDP)

As we have seen in section 3, the security of the one round schemes is related to the DP problem. Similarly, the security of the two round schemes seems to be related to the following problem, that we call the "Quadratic Degeneration Problem" (QDP).

Given: A multivariate equation of total degree four, $y = P(x_1, ..., x_n)$, where P is of degree four, with n variables x_1, ..., x_n.

Problem: Find (at most) $n-1$ polynomials of total degree two, P'_1, ..., P'_{n-1}, and a polynomial Q of total degree two, such that:

$$\begin{cases} \forall i,\ 1 \le i \le n-1,\ x'_i = P'_i(x_1, ..., x_n) \\ y = Q(x'_1, ..., x'_n) \end{cases}$$

We do not know if an algorithm with polynomial complexity exists for this problem (when a solution exists). So we do not know how to cryptanalize the general two-round schemes.

Remark: In the problem above, we have to find $n-k$ polynomials P'_1, ..., P'_{n-k}, with $k=1$. When $k=0$, this is the decomposition problem (of a multivariate polynomial of total degree 4). If the problem is easy for $k=1$, or for $k=2$, or for $k=3$ for example, then it would give a cryptanalysis of two-round schemes, and this problem may indeed be easier for $k=1$, or $k=3$, than for $k=0$.

10.2 Effect of the gradient cryptanalysis

The method we described in section 3.4, in order to cryptanalyse one round of S-boxes, can be extended to the case of two rounds. But, fortunately or unfortunately, it does not seem to lead to a practical attack against our schemes. Let us describe the idea.
We use the notations of sections 6 and 10.1. Let us consider $h = s \circ \phi \circ r$, which is given by n quadratic polynomials $h_1, ..., h_n$ in n variables over K. Let $f = \sum_i \alpha_i P_i$, where the α_i's are randomly chosen in K, and let $g = \sum_i \alpha_i (t \circ \psi)_i$, so that $f = g \circ h$.

We suppose for simplicity that the sizes of the S-boxes of ψ are $n_1 = ... = n_8 = 8$. Then, with a probability $\frac{1}{2^8} = \frac{1}{256}$, g has no components from the first S-box. An easy calculation then gives:

$$\mathbf{grad}\ f(x) = \sum_{r=1}^{n} \frac{\partial g}{\partial c_r}(h(x))\ \mathbf{grad}\ h_r(x) = \sum_{r=9}^{n} \frac{\partial g}{\partial c_r}(h(x))\ \mathbf{grad}\ h_r(x).$$

Therefore, $\mathbf{grad}\ f(x)$ lies in the subspace generated by $(\mathbf{grad}\ h_9(x), ..., \mathbf{grad}\ h_n(x))$, and that is true for any $x = (x_1, ..., x_n)$ in K^n. So we may deduce the following equalities:

$$\begin{cases} det(\mathbf{grad}\ f(x), \mathbf{grad}\ h_2(x), ..., \mathbf{grad}\ h_n(x)) = 0 \\ det(\mathbf{grad}\ h_1(x), \mathbf{grad}\ f(x), \mathbf{grad}\ h_3(x), ..., \mathbf{grad}\ h_n(x)) = 0 \\ ... \\ det(\mathbf{grad}\ h_1(x), ..., \mathbf{grad}\ h_7(x), \mathbf{grad}\ f(x), \mathbf{grad}\ h_9(x), ..., \mathbf{grad}\ h_n(x)) = 0 \end{cases}$$

Let $(-1)^{i_0+j_0}\Delta_{i_0 j_0}(x)$ be the value of the determinant of the matrix $\left(\frac{\partial h_i}{\partial x_j}(x)\right)_{\substack{1\le i\ne i_0\le n \\ 1\le j\ne j_0\le n}}$. Then the equations become the eight following:

$$\sum_{j=1}^{n} \Delta_{ij}(x) \frac{\partial f}{\partial x_j}(x) = 0 \qquad (1 \le i \le 8),$$

where $\Delta_{ij}(x)$ is unknown and is polynomial of total degree $n-1$ in x. We may imagine using a lot of different values of x to obtain relations between the coefficients of the polynomials $\Delta_{ij}(x)$, but this task seems to be impractical, due to the very high degree of these polynomials.

Note 1: In the case of *one* round of S-boxes, the attack works, because the corresponding Δ_{ij} are constant polynomials. Thus we obtain about $8n$ equations about the $8n$ unknown Δ_{ij}, by trying about n different values of x. The Δ_{ij} can then be found by Gaussian reductions.

Note 2: Once again, the scheme seems to resist this attack because the *inverse* of the quadratic functions that we use, has a high total degree in x.

10.3 Effect of the affine multiple attack

Another attack, which is very general, was described in [14]. It can be used against schemes based on a univariate polynomial transformation hidden by secret affine bijective transformations.
This attack is based on the following fact: if f is a univariate polynomial over a finite field K, then by using a general algorithm (see for example [3]), we can compute an "affine multiple" of the polynomial $f(x) - y$, *i.e.* a polynomial $A(x, y) \in K[X, Y]$ such that:

1. Each solution of $f(x) = y$ is also a solution of $A(x, y) = 0$.
2. $A(x, y)$ is an affine function of x.

We will now see that, because of the affine multiple attack, we cannot choose $n_1 = ... = n_d = 1$ in our schemes, *i.e.* each S-box must have at least two elements of K as input and output.

- Suppose first that K has $q = 2^m$ elements and that $n_1 = ... = n_d = 1$ (for example $m = 8$, $n = 8$ and $d = 8$). Each S-box S_e is given by a univariate quadratic polynomial over $K = GF(2^m)$.

  If $m = 1$, then the cryptanalysis is obvious: the quadratic polynomial is in fact an affine function (because $x^2 = x$ if $GF(2)$), and thus $t \circ \psi \circ s$ is itself a secret affine function, so that the scheme can be broken as a one-round scheme, and so can be easily broken.

  If $m > 1$, then the same attack can also work as follows. The public equations are given over $K = GF(2^m)$, but the cryptanalyst can rewrite them over $GF(2)$, so that the previous attack applies, with mn public equations of degree 2 over $GF(2)$, instead of n public equations of degree 4 over $GF(2^m)$.

- Suppose now that K has $q = p^m$ elements, where p is a small prime, $p \neq 2$. Since the characteristic is not 2, the S-boxes cannot be seen as affine functions any more. However, we can use here the affine multiple principle to attack the scheme.

  Let $f_e(x) = \alpha_e x^2 + \beta_e x + \gamma_e$ be the univariate quadratic polynomial stored in S_e $(1 \leq e \leq d)$, and let $A_e(x, y)$ be an affine multiple of $f_e(x) - y$. If all the exponents in y are $\leq k$, then there will be an attack with a Gaussian reductions on $O(n^{1+k})$ terms. Moreover, since p is small, k is also small, so that this attack is efficient.

  Example: We take $p = 3$. An easy calculation gives $A_e(x, y) = \alpha_e^2 x^3 - (\alpha_e y + \beta_e^2 - \alpha_e \gamma_e)x + \beta_e(y - \gamma_e)$. Then, by Gaussian reductions, we compute all the equations of the form:

$$\sum_{i,j,k} \mu_{ijk} y_i x_j x_k + \sum_{i,j} \nu_{ij} x_i x_j + \sum_{i,j} \xi_{ij} y_i x_j + \sum_i \omega_i x_i + \sum_i \theta_i y_i + \pi_0 = 0$$

  and we can then proceed as in section 7.

In conclusion, we do not recommend using S-boxes given by a univariate polynomial of degree 2 over K.

Note If we have for example $K = GF(2)$, $n = 64$ and $n_1 = ... = n_8 = 8$, we can imagine a similar attack, if we consider that each S-box S_e can be given as a univariate polynomial over $GF(2^8)$. But it is impractical, because this polynomial f_e is generally of very high degree (about 256), and most of the time, the degree of $A_e(x, y)$ in y becomes very high too. Therefore, the Gaussian reductions is not feasible any more.

11 How to choose the parameters and smartcard implementations

Let $K = GF(2^m)$ (K is not necessarily of characteristic 2, but for simplicity we will assume that it is so; moreover the computations are a little easier in characteristic 2). Let n be, as usual, the length of the cleartext x or of the ciphertext y (*i.e.* $x \in K^n$ and $y \in K^n$). Let F be one of our candidate algorithms (with public polynomials of degree four).
We recommend choosing m and n such that:

$$\begin{cases} mn \geq 128 & \text{(C1)} \\ n \geq 12 & \text{(C2)} \end{cases}$$

We also recommend to not publish in the public key all the equations of the composition (*i.e.* to have a "$2R^-$" scheme). (C3)

Condition (C1): To avoid exhaustive search on the cleartext x, we need $mn \geq 64$. Moreover, Eli Biham found an attack, based on the "birthday paradox" that shows that we need in fact $mn \geq 128$. This attack is described in Appendix 1.

Condition (C2): When n is very small ($n < 8$ typically), then to solve a set of n polynomial equations of small total degree d ($d \leq 4$ for example) with n variables in a finite field K is feasible with *ad hoc* techniques (for example with GCD of polynomials, Gröbner bases, or by exhaustive search on some of the variables). Moreover, this is often easy even when K is large (because here n is very small). So, in order to avoid these attacks, we must have $n \geq 8$. Moreover, for polynomial equations of total degree $d = 2$, we recommend for security to have $n \geq 16$, even if it is not yet clear if these attacks are efficient when $8 < n < 16$. Similarly, for polynomial equations of total degree $d = 4$, we recommend for security to have $n \geq 12$, even if it is not clear if these attacks are efficient when $8 < n < 12$. (This gives the condition (C2)).

Condition (C3): Condition (C3) is here to avoid a nice idea from [4] that may create an efficient decomposition algorithm. This idea is described in Appendix 2.

Note: Instead of (C3), another way to avoid the results of paper [4] is to introduce a "perturbation" in the originally public equations, as it was suggested in [17] for the C^* scheme. These "perturbations" can consist in introducing extra variables (it will give a 2RV scheme), fixing some variables (it will give a 2RF scheme), mixing the equations with truly random ones (it will five a 2R$^+$ scheme), etc. Moreover, all these "perturbations" can be combined (it gives a 2R^{-+}VF scheme). See [17] for more details. (In [17], these ideas are used and studied in the case of a C^* scheme).

Speed: Our schemes are very fast in secret key computations (more than 100 times faster than a 512 bits RSA for example). However, our schemes may be slower in public key computations compared with a 512 bits RSA with a small public exponent e.

Length of the public key:

- If $m = 1$ (*i.e.* $K = GF(2)$), then the length of the public key is huge: it is 162 Mbytes with $n = 128$.
- If $m = 4$ (*i.e.* $K = GF(16)$), then the length of the public key is more realistic: it is 920 Kbytes if $n = 32$.
- Moreover, if r, s and t are linear (not only affine), and if the S-boxes are homogeneous, then the public polynomials will be homogeneous. If $m = 4$, the length of the public key is then 818 Kbytes if $n = 32$.
- If $K = \mathbf{F}_{257}$ or $K = \mathbf{F}_{256}$ and $n = 16$, then the length of the public key is only $\simeq 20$ Kbytes.
- It might also be possible to choose r, s, t and the S-boxes as polynomials with values in a subfield K' of K. For example, if $K = GF(16)$ and $K' = GF(2)$, then the public key is divided by 4: its length is now 205 Kbytes if $n = 32$. Moreover, if $K = \mathbf{F}_{256}$, $K' = GF(2)$ and $n = 16$, then the length of the public key is only $\simeq 2.5$ Kbytes. It is not yet clear if this decreases the security of the scheme or not.

So, if $m \neq 1$, the schemes can have a reasonable length for the public key, despite the degree four of the public polynomials.

Smartcard implementations, secret key computations: The secret computations are very easy and very fast in a smartcard. The RAM needed when $mn \leq 128$ is about 32 bytes. The ROM needed for the program which computes F is also very moderate. The secret functions r, s and t can be stored in EEPROM or be computed from a secret seed of 64 bits stored in EEPROM. If the S-boxes have very small inputs, they can be stored in EEPROM, or even in ROM if all the cards have the same S-boxes. For example, if a S-box takes 8 bits in input and gives 8 bits in output, it will be stored in 256 bytes. We may also have the S-boxes all the same, or such that each S-box S_i can easily be computed from S_1 and/or a small secret seed (if the S-boxes have larger inputs, for example 16 bits in input and 16 bits in output, then the S-boxes might be recomputed and the inversion might require some small polynomial resolutions). As we can see, the parameters can be chosen in order to have very efficient secret key computations in smartcards.

Smartcard implementations, public key computations: A smartcard can compute F at least as easily as F^{-1} when F is its own function, *i.e.* when it uses its secret values to compute F and F^{-1}. The public key is then not needed to compute F and F^{-1}. In some applications, the smartcard doesn't have to make public key computations (which are then done in a PC for example), but only secret key computations. In this case, our schemes are very efficient. However, in some applications, the public key is required (for example we may ask for a stored and signed public key). As we have seen above, in some implementations, we may have a public key of 7.5 Kbytes (in 1996, some smartcards can store 8 Kbytes of EEPROM, or more), but for most of the schemes, the public key is larger than that, and it will then not be possible to store it in the smartcard. In this case, and if the public key must be given, we can imagine that the signed public key is stored in another device than the secure chip of the card, for example in an optic storage on the card... Or, if we have a lot of time to compute the public key, the card will compute and give some specific cleartext/ciphertext pairs (it will choose these pairs of course), and by Gaussian reductionss, the public key will be computed outside (however, this might take a long time !). So, as we can see, the public key can be stored only in very few cases, and most of the schemes are very efficient in smartcards, when only secret key computations are required in the smartcard.

12 Concrete examples of S-boxes

Attacks on Matsumoto-Imai like cryptosystems often rely on the existence of general relations between the inputs x_i and the outputs y_j, that we can classify as follows:

Type 1 relations: $\sum \gamma_{ij} x_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0.$

Type 2 relations: $\sum \gamma_{ij} x_i y_j + \sum \mu_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0.$

Type 3 relations: $\sum \xi_{ijk} x_i y_j y_k + \sum \gamma_{ij} x_i y_j + \sum \mu_{ij} y_i y_j + \sum \alpha_i x_i + \sum \beta_i y_i + \delta_0 = 0.$

In order to avoid generalisations of these attacks against our two round schemes, we will select S-boxes such that no type 1, 2 or 3 relations exist (except $0 = 0$ or obvious relations such that $y_i = y_i^2$ in $\mathbf{F}_2$). Nicolas Courtois did for us some simulations in order to see if there are some small S-boxes without any type 1, 2, 3 relations. The results of these simulations are given in Table 1, Table 2 and Table 3.

Note: In table 1, table 2 and table 3, λ is by definition the number such that the S-boxes take λ elements of the field $\mathbf{F}_q$ in input and give also λ elements of $\mathbf{F}_q$ in output.

λ	2	3	4	5	6	7	8	9	10	11
$\mathbf{F}_2$	$\neq 0$	$\neq 0$	$\neq 0$	$\neq 0$	7 9 91	0 0 112	0 0 114	0 0 78	0 0 0	
$\mathbf{F}_4$	$\neq 0$	$\neq 0$	$\neq 0$	0 0 84	0 0 0					
$\mathbf{F}_8$	$\neq 0$	0 6 105	0 0 11	0 0 0						
$\mathbf{F}_{16}$	$\neq 0$	0 8 50	0 0 0							
$\mathbf{F}_{32}$	26 60 250	0 10 40	0 0 0							
$\mathbf{F}_{64}$	$\neq 0$	0 12 48								
$\mathbf{F}_{128}$	$\neq 0$	0 14 56								

Table 1: some results obtained in characteristic 2

Legend:

type 1 type 2 type 3	: means that we found at least one S-box with those numbers of independent equations.
$\neq 0$	: means that we did not find a S-box with 0 equations of type 1.
	: we did no simulations but we expect no equations of type 1, 2, 3 for most of the S-boxes

Examples:

- For $\lambda = 10$ and $\mathbf{F}_2$, we found a S-box with 10 bits in input and 10 bits in output, where no type 1, 2 or 3 equations exist.
- For $\lambda = 4$ and $\mathbf{F}_{16}$, we found a S-box with 4 elements of $\mathbf{F}_{16}$ in input and 4 elements of $\mathbf{F}_{16}$ in output, where no type 1, 2 or 3 equations exist.

λ	2	3	4	5
$\mathbf{F}_3$	2 4 10	0 1 14	0 0 9	0 0 0
$\mathbf{F}_9$	2 4 14	0 0 1	0 0 0	
$\mathbf{F}_{27}$	3 3 21	0 0 0		

Table 2: some results in characteristic 3

λ	1	2	3
$\mathbf{F}_5$	0 1 1	0 0 0	0 0 0
$\mathbf{F}_7$	0 0 0		
$\mathbf{F}_{17}$	0 0 0		
$\mathbf{F}_{251}$	0 0 0		

Table 3: some results in characteristic ≥ 5

Conclusion: Even when λ is very small, we can find some S-boxes with no type 1, 2, 3 equations. This is nice, since large values of λ mean more complex S-boxes, and could imply larger public keys. The example given in section 11, with $n = 16$ and $K = \mathbf{F}_{16}$ (with a public key of 30 Kbytes) can be obtained with 4 S-boxes, each of which has 4 elements of $\mathbf{F}_{16}$ in input and output, and where no type 1, 2, 3 relations exist.

13 Is it possible to have bijective S-boxes ?

A natural question is the following: is it possible to use *bijective* S-boxes in our schemes ?

A first idea is using constructive methods to build bijective S-boxes. Unfortunately, such methods have always given the possibility of an attack of the scheme, so far (for example with C^* or triangular-functions).

If the probability of being bijective were not too small, we could randomly generate functions with a small degree, and thus find some of them which are bijective and have no specific structure (for example, this second idea works for functions of degree 1). When the degree is ≥ 2, obtaining an accurate evaluation of the probability of being bijective is difficult. But, if we suppose that this probability is about the same as for randoms functions, it may be shown (see below) that building bijective S-boxes with this method gives a huge public key for our schemes.

A new method for building efficient and strong bijective S-boxes with small degree may be discovered some day, but at the present, all known methods give not efficient or weak bijective S-boxes.

Rough evaluation of the number of "strong" bijective S-boxes

The problem : We would like to know if some "strong" bijective S-boxes exist, *i.e.* some functions f from $GF(2^n)$ to $GF(2^n)$ $(2 \le n \le 32)$, such that :

1. f is a bijection.
2. When we consider $GF(2^n)$ as a vector space of dimension n over $GF(2)$, then in a basis, f is given by n polynomials of degree $\le d$ (for us, $d = 2$ or $d = 3$).
3. f is "strong", *i.e.* the "construction of f" does not give any obvious weakness for the security of our schemes (we will give more details about this point below).

For example, with a triangular construction, or with a Matsumoto-Imai C^* construction, we can very easily design a function f that satisfies 1 and 2. However, the construction of f gives a way to attack the schemes. This comes from the fact that, if $f(x) = y$, where $x = (x_1, ..., x_n)$ and $y = (y_1, ..., y_n)$, and $y_i = P_i(x_1, ..., x_n)$ $(1 \le i \le n)$, then in these constructions, there are some polynomial equations $Q_i(x_1, ..., x_n, y_1, ..., y_n) = 0$ of small total degree, and of degree one in the x_i variables. More generally, we do not want either to have equations $Q_i(x_1, ..., x_n, y_1, ..., y_n) = 0$ of small total degree, except sums of products of the public equations by polynomials of small total degree (*i.e.* except the polynomials of the ideal of $K[x_1, ..., x_n, y_1, ..., y_n]$ generated by the public equations).
When this occurs, we will say that the "construction of f" gives a weakness, and if this does not occur, we will say that f is a "strong" bijection. So, do strong bijections exist (for $d = 2$ or $d = 3$ for example) ?

First evaluation : Let α_n be the probability for a function from $GF(2^n)$ to $GF(2^n)$ to be a bijection. Let H_d be the number of functions from $GF(2^n)$ to itself, that are given by polynomial equations of degree $\le d$ in a basis over $GF(2)$. Then, in a first evaluation, we may think that the number of "strong" bijections from $GF(2^n)$ to $GF(2^n)$ of degree $\le d$ is about $\alpha_n H_d$. This comes from the fact that, in a first evaluation, we may suppose that most of the bijections are "strong" and that the density of strong bijections in H_d is perhaps about the same as the density of strong bijections in the set of *all* functions from $GF(2^n)$ to $GF(2^n)$.
An easy calculation gives :

$$\alpha_n = \frac{(2^n)!}{2^{n.2^n}}, \qquad H_2 = 2^{n(1+n+\frac{n(n-1)}{2})}, \qquad H_3 = 2^{n(1+n+\frac{n(n-1)}{2}+\frac{n(n-1)(n-2)}{6})}.$$

Let $N = 2^n$. The Stirling formula $N! \sim N^N e^{-N}\sqrt{2\pi N}$ gives :

$$\alpha_n = \frac{N!}{N^N} \simeq e^{-N}\sqrt{2\pi N}.$$

Example 1 : Let $n = 10$. Then $N = 1024$ and $\alpha_{10} \simeq \frac{1}{2^{1471}}$, $H_2 = 2^{560}$, $H_3 = 2^{1760}$. Since $\alpha_{10}H_2$ is much smaller than 1, in first approximation, we expect to have no "strong" bijections of degree two over $GF(2^{10})$. Moreover, since $\alpha_{10}H_3 \simeq 2^{289}$, in first approximation, we expect to have about 2^{289} "strong" bijections over $GF(2^{10})$, of degree three (in a basis).

Example 2 : Let $n = 6$. Then $N = 64$ and $\alpha_6 \simeq \frac{1}{2^{88}}$, $H_2 = 2^{132}$. Then, in this first evaluation, we may expect to have about $\frac{2^{132}}{2^{88}} = 2^{44}$ "strong" bijections over $GF(2^6)$, of degree two in a basis. However, we will see now that this is probably not true.

Second evaluation : If f is a "strong" bijection and if g and h are two affine bijective functions, then $f' = g \circ f \circ h$ is also a "strong" bijection. Moreover, there are exactly

$$q^{n(n+1)}\Big[\Big(1-\frac{1}{q}\Big)\Big(1-\frac{1}{q^2}\Big)...\Big(1-\frac{1}{q^n}\Big)\Big]$$

bijections from $GF(q^n)$ to $GF(q^n)$ that are affine over $GF(q)$. So, since we have exactly

$$C = 2^{2n(n+1)}\Big[\Big(1-\frac{1}{2}\Big)...\Big(1-\frac{1}{2^n}\Big)\Big]^2$$

possibilities for (g,h), we see that if the number B of "strong" bijections (obtained in the first evaluation) is much smaller than C, then is does not mean that there are "about B" strong bijections, but it means, in this second evaluation, that we expect to have *no* such bijection.

Examples :

- In our example 1 above, $C \simeq 2^{217}$ and since $289 > 217$, we expect indeed to have about 2^{289} "strong" bijections of degree three, as claimed. These bijections of degree three are expected to come from about 2^{72} bijections f_i of degree three, and to be of the form $g \circ f_i \circ h$, where g and h are two affine bijections.
- However, in our example 2 above, we have $C \simeq 2^{81}$, and since $44 < 81$, we expect to have no strong bijections of degree two with $n = 6$ (if we had one, then there would exist at least about 2^{81} such bijections).

Results : With our "second evaluation", the results are :

- No "strong" bijective functions of degree $d = 2$ are expected to exist.
- "Strong" bijective functions of degree $d = 3$ are expected to exist if and only if $n \leq 10$.
- "Strong" bijective functions of degree $d = 4$ are expected to exist if and only if $n \leq 13$.
- "Strong" bijective functions of degree $d = 5$ are expected to exist if and only if $n \leq 16$.

So, if we want to design a bijective scheme with composition of a quadratic function h and a bijective function g made with bijective S-boxes, g will have to be of degree $d \geq 3$, and therefore $F = g \circ h$ will be of degree ≥ 6. However, a function F of degree 6 from $GF(2^n)$ to $GF(2^n)$ will have a public key of 635 Mbytes if $n = 64$. This is too large for all practical applications (at least at the present !). So, if this second evaluation is valid, we can conclude that we will not have bijective S-boxes in our two-round schemes.

Note : More precisely, we can conclude that this "second evaluation" shows that there are probably no strong bijective S-boxes of small degree for "random reasons". However, there may be some for "structural reasons", *i.e.* the hypothesis that the strong permutations are almost randomly distributed in the subset of multivariate functions of total degree d, may be false. For example, the probability for a linear function to be a permutation is much higher than the probability for a random function to be a permutation. So, some strong bijective S-boxes may exist despite our evaluations. In this appendix, we have just studied two "first evaluations" based on a random distribution hypothesis, but this hypothesis may be wrong, and some constructions for strong permutations of small degree may still exist.

14 Comparison with symmetrical cryptosystems

The cryptosystems we study in this paper have a lot of similarities with classical symmetric cryptosystems (such as DES for instance), since they use for example S-boxes (*i.e.* local transformations of a small number of values), followed by linear transformations, and there are several rounds (two for our schemes). However, DES for instance (cf [2]), or Khufu (cf [9]) are not secure when only very few rounds are used. So, why do our schemes (with only two rounds) resist classical attacks ? This can be explained by the following arguments:

1. In each round that uses S-boxes, *all* the input bits are transformed, and not only half of them, as happens in a Feistel scheme, such as the one used in DES.

2. The affine transformations are *secret*, and that is not the case for the P transformation of DES, which is public.

3. Moreover, the affine transformations are very general, *i.e.* every output bit is a linear combination of *all* the input bits, and not only of one input bit, such as in the P transformation of DES.

4. In our schemes, the S-boxes can be secret or public. In DES, they are public, and in Khufu they are secret.

Note 1: R. Rivest recently proposed XDES, which is a composition of DES, an initial simple affine secret transformation, and a final one (more precisely, these transformations consist in XORing the input with a secret value). Maybe this change does not strengthen DES against differential of linear cryptanalysis, but it seems to prevent other attacks (in particular, against exhaustive search on the key, cf [10]). So, XDES may illustrate the idea that composing an encryption algorithm with initial and final affine secret transformations, may lead to a significant strengthening of this algorithm.

Note 2: In our schemes, it is very important to have only two rounds, because the composition of three rounds, each round being quadratic, would lead to polynomials of degree eight, so that the length of the public key would be much too large for practical applications.

15 Conclusion

In this paper, we have studied new asymmetric algorithms which all rely on the idea of using one or two rounds of very simple quadratic transformations, which are hidden by secret affine tranformations. By "very simple" quadratic transformations, we mean quadratic transformations given by a triangular set of equations, or given by quadratic and small S-boxes. When there is only one round like this, or when the second round is built with functions whose algebraic structure is poorly hidden, we have proven that the corresponding schemes are insecure. From these ideas, we were able to design some new cryptanalysis of the Matsumoto and Imai scheme C^*.

However, when the parameters and the quadratic polynomials involved in each round are carefully chosen, we still have candidate algorithms that seem to resist the attacks. In this paper, the main characteristic of these schemes is that, in the second round, they use S-boxes, which are given by multivariate polynomials of small degree, and are randomly chosen in order to avoid any simple algebraic structure.

Since the publication of these algorithms at ICICS'97, two cryptanalysis papers ([1] and [4]) have been written on these algorithms. Due to [1], it is necessary that the input of the schemes has at least 128 bits. Due to [4], it is recommended to not publish all the originally public equations.

When all this is done, are these algorithms secure ? If they are, it would be a surprising and easy way of designing asymmetric cryptosystems. If they are not, it would strengthen the idea that the messages cannot be split in small branches, but must be transformed in a global way, and therefore that we need algebra to build secure asymmetric cryptosystems.
So the question remains open...

References

[1] Eli Biham, *Cryptanalysis of Patarin's 2-Round Public Key System with S-Boxes*, not yet published.

[2] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the full 16-Round DES*, CRYPTO'92, Springer-Verlag, pp. 487-496.

[3] Ian Blake, XuHong Gao, Ronald Mullin, Scott Vanstone, Tomik Yaghoobian, *Applications of finite Fields*, Kluwer Academic Publishers, p. 25.

[4] Y. Ding-Feng, L. Kwok-Yan, D. Zong-Duo, *Cryptanalysis of the "2R" schemes*, Proceeding of CRYPTO'99, Springer, pp. 315-325.

[5] Matthew Dickerson, *The functional Decomposition of Polynomials*, Ph.D Thesis, TR 89-1023, Department of Computer Science, Cornell University, Ithaca, NY, July 1989.

[6] Matthew Dickerson, *The Inverse of an Automorphism in polynomial Time*, IEEE 30^{th} annual symposium on Foundations of Computer Science (FOCS), 1989, pp. 82-87.

[7] Harriet Fell and Whitfield Diffie, *Analysis of a public Key Approach based on polynomial Substitutions*, CRYPTO'85, Springer-Verlag, pp. 340-349.

[8] Michael Garey, David Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness*, Freeman, p. 251.

[9] Henri Gilbert, Pascal Chauvaud, *A chosen Plaintext Attack of the 16-Round Khufu Cryptosystem*, CRYPTO'94, Springer-Verlag, pp. 359-368.

[10] Joe Kilian, Phillip Rogaway, *How to protect DES against eshaustive Key Search*, CRYPTO'96, Springer-Verlag, pp. 252-267.

[11] Rudolf Lidl, Harald Niederreiter, *Finite Fields*, Encyclopedia of Mathematics and its applications, volume 20, Cambridge University Press.

[12] Tsutomu Matsumoto, Hideki Imai, *Public quadratic polynomial-Tuples for efficient Signature-Verification and Message-Encryption*, EUCROCRYPT'88, Springer-Verlag, pp. 419-453.

[13] Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai public Key Scheme of Eurocrypt'88*, CRYPTO'95, Springer-Verlag, pp. 248-261.

[14] Jacques Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new Families of asymmetric Algorithms*, EUROCRYPT'96, Springer-Verlag, pp. 33-48.

[15] Jacques Patarin, *Asymmetric Cryptography with a Hidden Monomial*, CRYPTO'96, Springer-Verlag, pp. 45-60.

[16] Jacques Patarin, Louis Goubin, *Trapdoor one-way permutations and multivariate polynomials*, ICICS'97, Springer, November 1997, pp. 356-368.

[17] Jacques Patarin, Louis Goubin, Nicolas Courtois, *C^*_{-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai*, Advances in Cryptology, Proceedings of ASIACRYPT'98, Springer, pp. 45-60.

[18] Bruce Schneier, *Applied cryptography*, second edition, John Wiley and Sons, chapter 20, p. 501.

[19] Joachim von zur Gathen, *Functional Decomposition of Polynomials: the tame Case*, J. Symbolic Computation (1990), vol. 9, pp. 281-299.

[20] Joachim von zur Gathen, *Functional Decomposition of Polynomials: the wild Case*, J. Symbolic Computation (1990), vol. 10, pp. 437-452.

Appendix 1

An attack of Eli Biham based on the birthday paradox (cf [1])

Introduction: Eli Biham found an attack of the scheme based on two rounds of S-boxes with a complexity about $\mathcal{O}(\sqrt{q^n})$ (instead of $\mathcal{O}(q^n)$) for exhaustive search of one cleartext). Moreover, after his attack, it is possible to decipher very fast any ciphertext (not only one). At ICICS'97, we gave examples with $q^n \geq 2^{128}$, so that Eli Biham's attack is not very efficient against our published schemes, but however this attack is very interesting because it illustrates (one more time) the efficiency in cryptanalysis of the "birthday paradox" and it shows that for schemes based on two rounds of S-boxes we really need $q^n \geq 2^{128}$. (In the HFE schemes of [14], different ways of having a scheme with $q^n \simeq 2^{64}$ are explained. These ways of having $q^n \simeq 2^{64}$ should not be generalized in the case of two rounds of S-boxes.)

The idea: The idea is to use the fact that two inputs that differ only for one S-box will give the same output (*i.e.* a collision) if they have the same output for this S-box.
The starting point of the attack is to search for a collision $f(a) = f(b)$, and then to randomly choose values c_i's and search for (about 100) collisions $f(c_i \oplus a) = f(c_i \oplus b)$. From these collisions, we will have a way to detect the first secret linear transformation and the S-boxes of the first round.

Appendix 2

An attack (from [4]) for the decomposition problem

In [4], an algorithm is suggested for computing the decomposition of two quadratic multivariate polynomials. The efficiency of this algorithm relies on two hypotheses. No simulations have been done so far on this algorithm, so it is not easy to evaluate the probablity of the algorithm to succeed (*i.e.* when the two hypotheses are valid). However, this algorithm looks sufficiently dangerous to recommend to not publish <u>all</u> the equations of a composing $h = f \circ g$ in the public key of the 2R schemes described in this paper.

Notations: Let $h = f \circ g$, when f and g are two quadratic functions from $(\mathbf{F}_q)^n$ to $(\mathbf{F}_q)^n$. In a basis, g is given as $(g_1, ..., g_n)$, where g_i, $1 \leq i \leq n$ is a function from $(\mathbf{F}_q)^n$ to $\mathbf{F}_q$.

Aim of the algorithm: The aim of the algorithm is to find the vector space G generated by g_1, ..., g_n, *i.e.* $G = \text{Vect}(g_1, ..., g_n)$. From G, it is then easy to find a decomposition of h. Since $h = f \circ g$, h is also equal to $(f \circ A^{-1}) \circ (A \circ g)$, where A is any linear and bijective function from $(\mathbf{F}_q)^n$ to $(\mathbf{F}_q)^n$.

Remark: G is a vector space of dimension about n and G is included in the vector space of dimension about $\frac{n^2}{2}$ of all the quadratic polynomials from $(\mathbf{F}_q)^n$ to $(\mathbf{F}_q)^n$.

How to compute G: Let $V = \left\{ \sum\limits_{i=1}^{n} x_i G \right\}$. (Remark: V is a vector space of dimension about n^2 and V is included in the vector space of dimension about $\frac{n^3}{6}$ of all the cubic polynomials from $(\mathbf{F}_q)^n$ to $(\mathbf{F}_q)^n$.)
To compute G, the algorithm uses two hypotheses:

Hypothesis 1:

$$V = \text{Vect}\Big(\frac{\partial h_i}{\partial x_j}\Big).$$

When this hypothesis 1 is true, then we can compute V, since h is given.

Hypothesis 2:

$$G = \{\text{polynomials } r \text{ of degree 2 such that: } \forall i,\ 1 \leq i \leq n,\ x_i \cdot r \in V\}.$$

When this hypothesis 2 is true, then we can compute G since this hypothesis 2 will give relations of degree one on the coefficients of r.

Remark: To avoid problems such that $x_i^4 = x_i$ or $x_i^3 = x_i$ or $x_i^2 = x_i$, the authors of [4] make the hypothesis that $q \geq 5$.