Analysis of a Public Key Approach Based on Polynomial Substitution

Harriet Fell
Northeastern University
Boston, Massachusetts

Whitfield Diffie
Bell-Northern Research
Mountain View, California

1 *Introduction*

Ever since the discovery of public key cryptography in 1975[2], the search for public key cryptosystems has been a central theme of cryptographic research. The public key cryptosystems[5,8,7] that have been investigated during this period, however, are slower than conventional systems, require more storage, and, being based on areas of mathematics that were not previously important in cryptography, have not inspired the same degree of trust as conventional systems. It would therefore be desirable to develop new techniques based on principles both different from those employed in current public key cryptosystems and more closely allied with conventional cryptography.

Several years ago, after the development of the public key concept, but before any plausable examples were known, a suggestion along these lines was made to one of the authors by John McCarthy of Stanford University, who said he had gotten the idea from talking with an algebraic geometer about birational transformations. The idea was to build inverse pairs of multivariate polynomial transformations by a procedure commonly employed in algebraic geometry to construct inverse pairs of rational transformations.

2 *The Fundamental Scheme*

Our approach is to regard the plaintext as an n-vector of elements selected from a suitable ring R and build an invertible polynomial transformation $\mathbf{P}$ of several variables from R^n to R^n. The coefficients of this transformation will be the public key and the inverse transformation $\mathbf{Q}$ the secret key. Thus:

$$\begin{array}{ccc} \text{Plaintext} & \mathbf{P} & \text{Ciphertext} \\ \overline{x} = (x_1, \dots, x_n) & \mapsto & (P_1(\overline{x}), \dots, P_n(\overline{x})) \end{array}$$

where $x_1, \dots, x_n \in R$ and $P_1, \dots, P_n$ are multivariate polynomials with coefficients in R.

Assume, for example, that the plaintext is a vector of three components, x, y and z, from a ring R and that p_0 and p_1 are polynomials each in one variable over R. We can now build up a polynomial transformation of three variables by acting on the variables one at a time.

In the first round, (x, y, z) is carried to:

$$(x_1, y_1, z_1) = (x, y, z + p(x, y)).$$

where p can be either p_0 or p_1. In the second, (x_1, y_1, z_1) goes to:

$$(x_2, y_2, z_2) = (x_1 + p(y_1, z_1), y_1, z_1).$$

H.C. Williams (Ed.): Advances in Cryptology - CRYPTO '85, LNCS 218, pp. 340-349, 1986.

The process continues:

$$(x_3, y_3, z_3) = (x_2, y_2 + p(x_2, z_2), z_2)$$

until after a number of rounds:

$$P(x, y, z) = (x_{k-1} + p(y_{k-1}, z_{k-1}), y_{k-1}, z_{k-1})$$

is a nonlinear, invertible, polynomial transformation on a module, M, of dimension 3 over R. The secret key is the sequence of choices of p_0 or p_1 and the order in which they are applied to x, y, and z. For example:

$$\begin{aligned} (x_1, y_1, z_1) &= (x + p_1(y, z), y, z) \\ (x_2, y_2, z_2) &= (x_1, y_1, z_1 + p_0(x_1, y_1)) \\ (x_3, y_3, z_3) &= (x_2, y_2 + p_0(x_2, z_2), z_2) \\ (x_4, y_4, z_4) &= (x_3, y_3, z_3 + +p_1(x_3, y_3)) \\ P(x, y, z) = (x_5, y_5, z_5) &= (x_4 + p_0(y_4, z_4), y_4, z_4) \end{aligned}$$

The secret key is $((x, 1), (z, 0), (y, 0), (z, 1), (x, 0))$. The inverse transformation can be found by using the key in reverse: $((x, 0), (z, 1), (y, 0), (z, 0), (x, 1))$, i.e.:

$$\begin{aligned} (x_4, y_4, z_4) &= (x_5 - p_0(y_5, z_5), y_5, z_5) \\ (x_3, y_3, z_3) &= (x_4, y_4, z_4 - p_1(x_4, y_4)) \end{aligned}$$

and so on.

Naturally the number of polynomials need not be limited to 2 ncr the dimension, d, of M over R to 3, but there is probably no virtue in using polynomials of degree other than $d - 1$.

This plan offers on its face, not only plausible hope for constructing inverse pairs of transformations, but one with very close ties to the shift register mathematics of conventional cryptography. The alternate transformation of variables x and y is closely analogous to alternate operation on the left and right halves in DES[3]. In general, the notion of modifying some components of a vector by adding to them functions of other components underlies all shift registers both linear and nonlinear[1,4].

A key difference between the construction of conventional cryptosystems and public key cryptosystems lies in the way the systems are presented to the user. A shift register cryptosystem is a description of the way in which the plaintext is modified incrementally throught a number of interations to become the ciphertext. Such a description precludes public key use because it can equally readily be read in the other direction as a description of how to derive the plaintext from the ciphertext by incremental modifications. In order to develop a public key system along these lines, it is necessary to simplify the equations that arise from the incremental substitution process in such a way as to conceal the substitutions.

On first glance, it seems sufficient to carry out the substitutions as the process goes on. On second, it becomes obvious that the number of coefficients in the polynomials will grow to astronomical proportions after only a few interations. In order to prevent the equations from exploding into unusable bulk, some device must be found for eliminating most of the terms; the most obvious such devices are nilpotence and J-rings and these will be examined in the remainder of this paper.

3 *Reducing the Number of Coefficients*

3.1 *Nilpotence*

A ring R is nilpotent if there is an integer $k \geq 1$ such that $(R)^k = \{0\}$. That means that for any elements $r_1, \ldots, r_k \in R$ the product $r_1 r_2 \cdots r_k$ is zero, so a multivariate polynomial with coefficients in R can only have meaningful terms whose total degree does not exceed k.

There is a hitch, however, in applying nilpotence in our fundamental scheme. We expect the scheme, described in section 2, to yield a transformation of the form:

$$\begin{array}{ccc} \text{plaintext} & \mathbf{P} & \text{ciphertext} \\ \bar{x} = (x_1, \ldots, x_n) & \mapsto & (P_1(\bar{x}), \ldots, P_n(\bar{x})) \end{array} \qquad (3.1)$$

where $x_1, \ldots, x_n \in R$ and $P_1, \ldots, P_n$ have coefficients in R. Note that no transformation of this form can be invertible. Suppose that each P_i has constant term C_i. Let $T_i(\bar{x}) = P_i(\bar{x}) - C_i$. The system $\mathbf{P}$ is invertible if and only if the system $T = (T_i, \ldots, T_n)$ is invertible but T can never be invertible as $T_i(\bar{x}) \in (R)^2 \subsetneq R$.

What went wrong is that although the iteration in section 2 always produces an invertible transformation, if we apply this iteration scheme in a ring without a unit, we do not get a transformation of the form (3.1) but rather one of the form:

$$\begin{array}{ccc} \text{plaintext} & \mathbf{P} & \text{ciphertext} \\ \bar{x} = (x_1, \ldots, x_n) & \mapsto & \bar{x} + (P_1(\bar{x}), \ldots, P_n(\bar{x})) \end{array} \qquad (3.2)$$

Where $x_1, \ldots, x_n \in R$ and $P_1, \ldots, P_n$ have coefficients in R. The transformation (3.1) is a polynomial transformation but its coefficients are not all in R as R is nilpotent and cannot contain a unit element.

We can still make use of nilpotence, however. We take R to be a finite local ring, that is a finite commutative ring with 1 that has a unique maximal ideal N of nilpotents. (Note R/N is a finite field.) The general form of the encryption transformation again becomes:

$$\begin{array}{ccc} \text{plaintext} & \mathbf{P} & \text{ciphertext} \\ \bar{x} = (x_1, \ldots, x_n) & \mapsto & (P_1(\bar{x}), \ldots, P_n(\bar{x})) \end{array} \qquad (3.3)$$

where $x_1, \ldots, x_n$ and $P_1(\bar{x}), \ldots, P_n(\bar{x})$ all lie in N. That is, $\mathbf{P}$ is a multivariate polynomial transformation from R^n to R^n that is invariant and invertible on N^n. The number of coefficients in the polynomials is restricted by the nilpotence because terms of high total degree are identically zero on N^n and we do not care how they behave on the rest of R^n. The transformations generated by the iteration scheme of section 2 will be of this type if we choose the polynomials $p_1, p_2, \ldots$ to have coefficients in N or to have coefficients in R but constant terms in N.

3.2 *J-Rings*

Let R be a finite commutative ring. R is a J-ring if there is an integer $d \geq 1$ such that $a^d = a$ for any $a \in R$. A multivariate polynomial over a J-ring can be reduced to a polynomial all of whose terms have individual degrees $\leq d$ (or total degrees $< dn$ where n is the number of variables). This time the transformation will be of the form of equation (3.1).

3.3 *Upper-Triangular Matrices*

We represent the plaintext by a pair of upper-triangular matrices with entries from a finite ring R. The encryption transformation will be of the form:

$$\begin{array}{ccc} \text{plaintext} & \mathbf{P} & \text{ciphertext} \\ (X_1, X_2) & \mapsto & (P_1(X_1, X_2), P_2(X_1, X_2)) \end{array} \qquad (3.4)$$

where X_1, X_2 are upper-triangular $k \times k$ matrices over R and P_1, P_2 have coefficients in R. If $M_1, \ldots, M_k$ are upper triangular $k \times k$ matrices over R then the product $M_1 \cdots M_k$ is zero. This means that the polynomials will have terms of total degree at most $k - 1$. As the matrices do not commute, there are more terms to deal with than in the commutative nilpotent case (3.1) but there is also hope that non-commutativity will make lower degree polynomial systems more difficult to invert.

	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$	$k=14$	$k=15$	$k=16$	$k=17$	$k=18$
$n=2$	6 192 384	10 320 640	15 480 960	21 672 1344	28 896 1792	36 1152 2304	45 1440 2880	55 1760 3520	66 2112 4224	78 2496 4992	91 2912 5824	105 3360 6720	120 3840 7680	136 4352 8704	153 4896 9792	171 5472 10944	190 6080 12160
$n=3$	10 220 660	20 440 1320	35 770 2310	56 1232 3696	84 1848 5544	120 2640 7920	165 3630 10890										
$n=4$	15 240 960	35 560 2240	70 1120 4480	126 2016 8064	210 3360 13440												
$n=5$	21 273 1365	56 728 3640	126 1638 8190	252 3276 16380													
$n=6$	28 308 1848	84 924 5544	210 2310 13860														
$n=7$	36 360 2520	120 1200 8400	330 3300 23100														
$n=8$	45 360 2880	165 1320 10560															
$n=9$	55 440 3960	220 1760 15840															
$n=10$	66 462 4620	286 2002 20020															
$n=11$	78 462 5082	364 2184 24024															
$n=12$	91 546 6006	455 2730 32760															
$n=13$	105 525 6825	560 2800 36400															
$n=14$	120 600 8400	680 3400 47600															

For multivariate polynomials of degree k in n variables

126	$T(n,k) =$ # of terms possible in one polynomial
2016	$\lceil \frac{64}{n} \rceil \cdot T(n,k) =$ # of bits to represent one polynomial
8064	$n \cdot \lceil \frac{64}{n} \rceil \cdot T(n,k) =$ # of bits in the public key

Figure 3.1 Bits of Key in Commutative Case (Plaintext = 64 bits)

4 *Finding Systems of Practical Size*

The public key in the system we have proposed consists of the coefficients of the polynomials making up the transformation **P**. We do not want a key that is too large and have taken 10,000 bits to be the upper limit on the size of key that we will consider.

4.1 *The Commutative Case*

We must first count the maximum number of terms in a polynomial of total degree k in n variables. This number, $T(n,k)$ can be computed recursively as follows:

$$\begin{aligned} T(1,k) &= k+1 && (\text{i.e.,} 1, X, \ldots, X^k) \\ T(n,1) &= n+1 && (\text{i.e.,} 1, X_1, \ldots, X_n) \\ T(n,k) &= T(n,k-1) + T(n-1,k) && \forall_{k,n>1}. \end{aligned}$$

	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	
$n=2$	6 384 768	10 640 1280	15 960 1920	21 1344 2688	28 1792 3584	36 2304 4608	45 2880 5760	55 3520 7040	66 4224 8448	78 4992 9984	91 5824 11648
$n=3$	10 430 1290	20 860 2580	35 1505 4515	56 2408 7224	84 3612 10836						
$n=4$	15 480 1920	35 1120 4480	70 2240 8960	126 4032 16128							
$n=5$	21 546 2730	56 1456 7280	126 3276 16380								
$n=6$	28 612 3672	84 1848 11088									
$n=7$	36 684 4788	120 2280 15960									
$n=8$	45 720 5760	165 2640 21120									
$n=9$	55 825 7425	220 3300 29700									
$n=10$	66 858 8580	286 3718 37180									

For multivariate polynomials of degree k in n variables

35	$T(n,k) =$ # of terms possible in one polynomial
1120	$\lceil \frac{128}{n} \rceil \cdot T(n,k) =$ # of bits to represent one polynomial
4480	$n \cdot \lceil \frac{128}{n} \rceil \cdot T(n,k) =$ # of bits in the public key

Figure 4.2 Bits of Key in Commutative Case (Plaintext = 128 bits)

The recursion step follows since $T(n, k-1)$ is the number of terms of total $\leq k$ in which X_n appears. Each such term is of the form X_n times a term in n variables with total degree $\leq k-1$. $T(n-1, k)$ is the number of terms of total degree $\leq k$ in which X_n does not appear.

Now we need the number of bits necessary to represent the coefficients of a polynomial of total degree k in n variables. This clearly depends on the size of the ring but there are restrictions on the ring size if we want the plaintext size to conform to present standards. If the plaintext $(X_1, \ldots, X_n)$ is to have 64 bits then each X_i must represent $\lceil \frac{64}{n} \rceil$ bits. If R is a J-ring so that encryption method (3.1) is used, then R must have cardinality $2^{\lceil \frac{64}{n} \rceil}$. The number of bits needed to represent a single polynomial of total degree k in n variables is given by $\lceil \frac{64}{n} \rceil \cdot T(n,k)$. The number of bits in the public key polynomial transformation is $n \cdot \lceil \frac{64}{n} \rceil \cdot T(n,k)$. These numbers are computed and presented in Figure 4.1. The same computation for a plaintext of 128 bits is presented in Figure 4.2.

If the ring is local and the general encryption method of equation 3.3 is used, then the cardinality of the nilpotent ideal, N, must be $2^{\lceil \frac{64}{n} \rceil}$ or $2^{\lceil \frac{128}{n} \rceil}$. The cardinality of R is at least twice that of N. The number of bits needed to represent the public key is at least double the numbers that appear in Figures 4.1 and 4.2. If the specific iteration described in section 2 is used with $P_1, \ldots, P_n$ having coefficients in N then the number of bits needed to represent the public key is exactly as shown in Figures 4.1 and 4.2.

It is striking that if we are restricted to 10,000 bits of key then the polynomials making up the encryption transformation and also those making up its inverse can have no more than 153 terms ($n = 2, k = 16$, 64-bit plaintext). We shall see in the next section that this is too small for cryptographic security.

p	$\log p$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$	$k=12$	$k=13$	$k=14$	$k=15$	$k=16$	$k=17$
2	1	6	10	15	21	28	$\boxed{36}$	45	55	$\boxed{66}$	78	91	105	120	$\boxed{136}$
4	2	12	20	30	$\boxed{42}$	56	$\boxed{72}$	90	110	$\boxed{132}$					
8	3	18	30	$\boxed{45}$	63	$\boxed{84}$	108	$\boxed{135}$							
16	4	24	$\boxed{40}$	60	$\boxed{84}$	112	$\boxed{144}$								
32	5	30	50	$\boxed{75}$	105	$\boxed{140}$									
64	6	$\boxed{36}$	60	90	126										
128	7	42	$\boxed{70}$	105	$\boxed{147}$										
256	8	48	80	120											
512	9	54	90	$\boxed{135}$											
1024	10	60	100												
2048	11	$\boxed{66}$	110												
4096	12	72	120												
8192	13	78	$\boxed{130}$												
16384	14	84													
32768	15	90													
65536	16	96													
131072	17	102													
262144	18	108													
524288	19	114													
1048576	20	120													
2097152	21	126													
4194304	22	$\boxed{132}$													

Figure 4.3 Bits of Plaintext in one Upper-Triangular Matrix over Z_p

4.2 *The Upper-Triangular Case*

We consider the special case of $k \times k$ upper-triangular matrices over Z_p. We have taken p to be a power of 2 so that there is no bit loss in representing the ring. A single matrix carries $(\log p)\left(\frac{k^2-k}{2}\right)$ bits. In Figure 4.3 we show the number of bits of plaintext in a pair of upper-triangular matrices over Z_p.

Figure 4.3 also provides information in the case of a general commutative ring R. The number of bits shown for a particular p provides lower (upper) bounds when R is any commutative ring with cardinality greater than (less than) p.

				$2^k - 1 =$		bits of key =
			# bits per	# terms per	# bits per	# bits per 2
p	log p	k	matrix	polynomial	polynomial	polynomials

64 Bit Plaintext

2	1	9	36	511	511	1022
4	2	7	42	127	254	508
8	3	6	45	63	189	378
16	4	5	40	31	124	248
64	6	4	36	15	90	180

128 Bit Plaintext

2	1	12	66	4095	4095	8190
4	2	9	72	511	1022	2044
8	3	8	84	255	765	1530
16	4	7	84	127	508	1016
32	5	6	75	63	315	630
128	7	5	70	31	217	434
2048	11	4	66	15	165	330

Figure 4.4 Bits of Key; Upper-triangular Case

In Figure 4.4 we compute the number of bits in the public key when the plaintext has 64 and 128 bits respectively. To do this computation we must find the number of terms in a polynomial in 2 upper-triangular $k \times k$ matrices. Recall that these polynomials will have total degree at most $k-1$. The number of terms of exactly degree j in 2 non-commuting variables is 2^j. It is the same as making j choices from 2 items with repetition allowed. The total number of terms is, therefore:

$$\sum_{j=0}^{k-1} 2^j = 2^k - 1.$$

There appear to be cases where the key is not terribly big and where the number of terms in each polynomial is large enough that we might have cryptographic security. We will see, however, in the next section that we can solve for the coefficients of the polynomials in the inverse transformation in layers so that we need never face a very large system of equations.

	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
$n=2$	7 224 448	15 480 960	31 992 1984	63 2016 4032	127 4064 8128	255 8160 16320
$n=3$	26 572 1716	81 1782 5346	241 5302 15918			
$n=4$	63 1008 4032	255 4080 16320				
$n=5$	124 1612 8060	624 8112 40560				
$n=6$	215 2365 14190					

For multivariate polynomials of degree k in n variables

81 1782 5346	$n^{k+1}-1 =$ # of terms possible in one polynomial $\lceil \frac{64}{n} \rceil \cdot (n^{k+1}-1) =$ # of bits to represent one polynomial $n \cdot \lceil \frac{64}{n} \rceil \cdot (n^{k+1}-1) =$ # of bits to represent the public key.

Figure 4.5 Bits of Key; Non-commutative Case; 64-Bit Plaintext

4.3 *The General Non-Commutative Case*

As will be shown, in the next section, the commutative and upper-triangular cases are not cryptographically secure so we offer one other suggestion. Let R be a non-commutative finite ring. In figure 4.5 we show the number of bits of key in this case. The number of terms in a polynomial of total degree k in n non-commuting variables is given by $n^{k+1}-1$. The reasoning is the same as that used to compute the number of terms in the polynomials of section 4.1 above. Figure 4.5 shows that there are very few cases to investigate. The number of terms in the polynomials is small but there is some hope that the complications of non-commutative arithmetic will impede cryptanalysis.

5 *Inverting These Systems*

Assume that we know the public key, $\mathbf{P} = (P_1, \ldots, P_n)$, and we want to find a transformation $\mathbf{Q} = (Q_1, \ldots, Q_n)$ such that $Q_i(P_1(\overline{x}), \ldots, P_n(\overline{x})) = X_i(i = 1, \ldots, n)$. We know that such a system $\mathbf{Q}$ of polynomials exists and that the Q_i have the same types of terms as the Pj. That is:

$$Q_i = a_i + b_{i_1} V_1 + \cdots + b_{i_n} V_n + c_{i 1_1} V_1^2 + \cdots.$$

We know which terms are present, we must find the coefficients of $Q_i\,(i = 1, \ldots, n)$.

5.1 *If R is a J-ring*

Pick a vector $\overline{A} = (A_1, \ldots, A_n)$ in R^n. Compute $P_1(\overline{A}), \ldots, P_n(\overline{A})$. Set $Q_1(P_1(\overline{A}), \ldots, P_n(\overline{A})) = A_1$. This gives a linear equation with coefficients in R whose unknowns are the coefficients of Q_1. Let q by the number of coefficients of Q_1. If we can produce vectors $\overline{A}_i = (A_{i_1}, \ldots, A_{i_n})$ in $R^n\,(i = 1, \ldots, q)$ such that the resulting linear equations are independent then, hopefully, we can solve for the coefficients of Q_1. As $\mathbf{Q}$ is invertible, we know that such an independent system exists. Any J-ring R is a direct sum of finite fields[8]. Hence the system can be solved independently over each component field of R using standard techniques of linear algebra over fields. A system of approximately 150 equations can be solved in a reasonable time by existing techniques and in our systems of practical size, Q_i never has more than 153 coeficients.

There remains the problem of generating q independent equations. We suggest the following simple procedure. 1. choose a $\overline{A} \not\equiv \overline{O}$ in R_n and accept the linear equations it produces. 2. After having found $k-1$ independent equations, choose a new vector $\overline{A} \not\equiv \overline{O}$ at random and accept it if it is independent of the $k-1$ vectors already found. Otherwise, discard it and repeat this step until you succeed. If, at each stage, the system is put, componentwise, into reduced row-echelon form, then checking the new equation and row-reducing the new system are both easy. We cannot prove that this method will produce the necessary q equations in a reasonable amount of time but believe it does for the following reasons:

Assume $R = Z_p$, p a prime. If k vectors are chosen at random from $(Z_p)^q$ then the probability that they are independent is given by $(1-\frac{1}{p^q})(1-\frac{1}{p^{q-1}})\cdots(1-\frac{1}{p^{q-k+1}}) > \frac{p-2}{p-1}$ so the probability that a random $q \times q$ determinant over Zp is non-singular is given by:

$$\prod_{i=1}^{q}(1-\frac{1}{p^i}) > \frac{p-2}{p-1}.$$

Although this gives 0 as a lower bound when $P=2$ the products are actually greater than 1/4 in this case.

Unfortunately, the coefficient vectors in these equations are not generated at random from all possible p^q vectors over Zp; we can only generate p^n vectors but, expect that they will be randomly distributed in the larger set. Given this, the above argument shows that we are likely to generate q independent equations without much difficulty.

5.2 *The Nilpotent Case*

The message is a vector in N^n where N is a nilpotent ring embedded as a maximal nilpotent ideal in a local ring R. The quotient ring R/N is a finite field. N^n is invariant under the public key polynomial map $\mathbf{P}:R^n \mapsto R^n$. That is, $\mathbf{P}:N^n \mapsto N^n$ is one to one and onto. The component polynomials P_i of $\mathbf{P}$ have coefficients in R. To find the coefficients of $Q = P^{-1}$ we first work over the field R/N and then raise the solution to R.

We can assume that P_i $(i = 1 \ldots n)$ has no constant term. Otherwise $\mathbf{P}(\overline{X}) = \mathbf{T}(\overline{X}) + \overline{C}$ where $\overline{C}$ is a constant vector in N^n and $\mathbf{T}(X)$ is a polynomial transformation whose components have no constant terms. $\mathbf{P}$ is invertible on N^n if and only if $\mathbf{T}$ is. If $\mathbf{U} = \mathbf{T}^{-1}$ on N^n then: $\mathbf{Q}(\overline{Y}) = \mathbf{U}(\overline{Y} - \overline{C})$ is $\mathbf{P}^{-1}$ on N.

Let $\mathbf{P}_L$ and $\mathbf{Q}_L$ be the linear parts of $\mathbf{P}$ and $\mathbf{Q}$. Then

$$\overline{x} = \mathbf{Q}(\mathbf{P}(\overline{x})) = \mathbf{Q}_L(\mathbf{P}_L(\overline{x})) + \text{higher order terms.}$$

Let $(\mathbf{P}_L \bmod N)$ mean the polynomial obtained by replacing each coefficient, c, of $\mathbf{P}_L$ by $(C \bmod N)$. We invert $\mathbf{P}_L$ to find $\mathbf{Q}_L$. First find

$$\mathbf{Q}_L = (\mathbf{P}_L \bmod N)^{-1} \qquad \text{over} R/N.$$

Now form $\mathbf{Q}''_L$ with coefficients in R by replacing each coefficient of $\mathbf{Q}'_L$ by a representative in R of its class. Then

$$\mathbf{Q}''_L \circ \mathbf{P}_L = I + B \quad \text{where } B \text{ has entries in } N$$

and

$$(I - B + B^2 - \cdots \pm B^{k-1})\mathbf{Q}''_L\mathbf{P}_L = I \pm B^k = I \quad \text{on } N^n.$$

So set

$$\mathbf{Q}_L = \mathbf{P}_L^{-1} = (I - B + B^2 - \cdots \pm B^{k-1})\mathbf{Q}''_L.$$

Now go on to the quadratic terms. Let $\mathbf{P}_Q$ and $\mathbf{Q}_Q$ be the quadratic parts and $\mathbf{P}_H$ and $\mathbf{Q}_H$ the higher order parts of $\mathbf{P}$ and $\mathbf{Q}$ respectively. Then

$$\begin{aligned}\bar{x} = Q(P(\bar{x})) &= \mathbf{Q}_L(\mathbf{P}_L(\bar{x}) + \mathbf{P}_Q(\bar{x}) + \mathbf{P}_H(\bar{x})) + \mathbf{Q}_Q(\mathbf{P}_L(\bar{x}) + \mathbf{P}_Q(\bar{x}) + \mathbf{P}_H(\bar{x})) \\ &\quad + \mathbf{Q}_H(\mathbf{P}_L(\bar{x}) + \mathbf{P}_Q(\bar{x}) + \mathbf{P}_H(\bar{x})) \\ &= \mathbf{Q}_L(\mathbf{P}_H(\bar{x})) + \mathbf{Q}_L(\mathbf{P}_Q(\bar{x})) + \mathbf{Q}_Q(\mathbf{P}_L(\bar{x})) + \text{higher order terms.}\end{aligned}$$

This gives a system of equations whose unknowns are the coefficients of $\mathbf{Q}_Q$. We can proceed as with the linear parts, finding the coefficients of $\mathbf{Q}$ one degree at a time.

For rings of practical size, these systems are therefore too easily solved to be secure.

5.3 *The Upper-Triangular Case*

The encrypting transformation is $(X_1, X_2) \mapsto \mathbf{P}(X_1, X_2) = (P_1(X_1, X_2), P_2(X_1, X_2))$ where X_1, X_2 are upper-triangular matrices over a commutative ring R and P_1, P_2 are polynomials with coefficients in R. To decrypt, we must find a polynomial system $\mathbf{Q} = (Q_1, Q_2)$ such that $Q_i(\mathbf{P}(X_1, X_2)) = X_i\,(i = 1, 2)$. As before, we can use $\mathbf{P}$ and $\mathbf{Q}$ to produce pairs $(U, V), Q(U, V)$ and to set up a system of linear equations in the coefficients of $\mathbf{Q}$. This system is particularly easy to solve as only the linear and constant terms show up in the entries just above the diagonal of $\mathbf{Q}(U, V)$. The quadratic terms enter into the entries two levels above the diagonal and so on. We can, therefore, solve for the coefficients of $\mathbf{Q}$ in a layered manner, similar to the nilpotent case.

6 *Conclusions*

We set out to build a public key cryptosystem by repeatedly substituting for variables in multivariate polynomials and simplifying the results to conceal the substitution process. There seems, however, to be no way to build such a system that is both secure and has a public key of practical size when the devices used to limit the number of coefficeints are nilpotence and J-rings. We have only shown, however, that it is impossible to produce such a system if the total degree of the encryption polynomial determines the size of the public key. Perhaps, by properly choosing p_0 and p_1, we can employ the fundamental scheme to produce sparse encrypting polynomials. Then the public key could be kept small while the encrypting polynomial has large total degree and is difficult to invert.

References

[1] Don Coppersmith and Edna Grossman, "Generators for Certain Alternating Groups with Applications to Cryptography," *SIAM J. Appl. Math.*, Vol. 29, No. 4, pp. 624–627, Dec 1975.

[2] Whitfield Diffie and Martin E. Hellman, "New Directions in Cryptography," *IEEE Trans. Info. Thy.*, Vol. IT–22, No. 6, pp. 644–654, November 1976.

[3] Data Encryption Standard, FIPS Pub. No. 46, National Bureau of Standards, 15 January 1977.

[4] Solomon W. Golomb, *Shift Register Sequences*, Holden Day, San Francisco, 1967.

[5] R. McLeice, A Public-Key Cryptosystem Based On Albebraic Coding Theory, DSN Progress Report 42–44, Jet Propulsion Lab, Calif. Inst. of Tech., Pasadina CA, Jan–Feb 1978.

[6] R. C. Merkle and M. E. Hellman, "Hiding Information and Signatures in Trapdoor Knapsacks," *IEEE Transactions on Information Theory*, Vol. IT–24, No. 5, pp. 525–530, September 1978.

[7] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *CACM*, Vol. 21, No. 2, pp. 120–126, February 1978.

[8] Gustavus J. Simmons. Personal Communication.