The Use of Fractions in Public-Key Cryptosystems

Hartmut Isselhorst

Gesellschaft für elektronische Informationsverarbeitung
Oxfordstr. 12-16, D-5300 Bonn 1
West Germany

Abstract. This paper discusses an asymmetric cryptosystem based on fractions, the R^k-system, which can be implemented fast using only additions and multiplications. Also it is very simple to initialize the system and to generate new keys. The R^k-system makes use of the difficulty to compute the numerator and the denumerator of a fraction only knowing the rounded floating point representation. It is also based on the difficulty of a simultaneous diophantine approximation with many parameters and only a little error bound.

INTRODUCTION

Many known public-key cryptosystems deal with integer problems like factorization, discrete logarithms or knapsacks. Searching for another foundation of security we allow the use of real numbers, especially fractions.

Everyone knows that it is easy to choose two primes p and q and to compute the product $n = p \cdot q$. But up to now it is difficult to calculate the factors p and q only knowing n, if n is greater than 10^{200}. But knowing n one has enough information to compute p and q, because factorization is deterministic. To avoid this one can try the following: Allowing real numbers it is possible to replace the multiplication by the division. To be more precisely, we pose the

J.J. Quisquater and J. Vandewalle (Eds.): Advances in Cryptology - EUROCRYPT '89, LNCS 434, pp. 47-55, 1990.

Problem

Let a and p be integers with $1 < a < p < 10^{1000}$ and $\gcd(a, p) = 1$. Denote

$$x_n = 10^{-n} \cdot \lfloor 10^n \cdot a/p \rfloor, \; n \in \mathbb{N}.$$

1. Is it possible to compute a and p from x_n with a suitable parameter n?
2. Is it possible to choose the parameter n in a way such that it is impossible to calculate a and p from x_n?

The following theorem solves the problem.

Theorem 1

Let a, p, k be integers with $10^{k-1} < p < 10^k$, $1 < a < p$, $\gcd(a, p) = 1$.
1. Only knowing x_{2k} it is easy to compute a and p.
2. One cannot calculate a and p from x_n, if $0 < n < 2k-50$ and p is a prime.

Proof:

1. Let $0 < s < t < 1$ and $s = /s_1,...,s_r/$, $t = /t_1,...,t_m/$ (the continued fractions of s and t). Put formally $s_i = \infty$ for all $i > r$ and $t_i = \infty$ for all $i > m$. Then find j with $s_i = t_i$ for all $i \in [1:j-1]$ and $s_j \neq t_j$. Define

$$q = \begin{cases} s_j+1 & j \in 2\mathbb{N},\ j \geq r \\ s_j & j \in 2\mathbb{N},\ j \geq r \\ t_j+1 & j \in 2\mathbb{N}+1,\ j < m \\ t_j & j \in 2\mathbb{N}+1,\ j \geq m. \end{cases}$$

Then $v = /s_1,...,s_{j-1},q/$ is the irreducible fraction in $[s,t]$ with the lowest denominator ([Knuth 81, p.606]).

If a/b and c/d are consecutive fractions in the Farey-sequence F_n, $n \geq 2$, it holds

$$\left| a/b - c/d \right| = \frac{1}{bd} \geq \frac{1}{n(n-1)}$$

([Niven and Zuckerman 76, p.186])

Hence $a/p \in [x_{2k}, x_{2k} + 10^{-2k}] \cap F_{10^k} = \{ a/p \}$.

So the algorithm above computes a and p from the input $s = x_{2k}$, $t = x_{2k} + 10^{-2k}$.

2. If a/p, c/d are consecutive fractions in F_p we have

$$[x_n, x_n + 10^{-n}] \cap F_p = \{a/p\} \Leftrightarrow 10^{-n} \le \left|\frac{a}{p} - \frac{c}{d}\right| = \frac{1}{dp} \Leftrightarrow n \ge \log_{10}(p \cdot d)$$

From [Horster and Isselhorst 89, p.101] we have

$$\frac{1}{|F_p|} \cdot \sum_{\substack{\frac{a}{b}, \frac{c}{d} \in F_p \\ \text{consecutiv}}} \log_{10}(b \cdot d) \approx 2 \cdot \log_{10}(p+1) - \frac{1}{\log_e(10)}.$$

so to compute a and p one needs to know x_n with $n = 2 \cdot \log_{10}(p)$ almost everytime. Knowing only $2 \cdot \log_{10}(p) - 50$ digits, one has to guess 50 sequential digits following x_n or approximately 50 partial quotients of a/p ([Isselhorst 88, p.104]).
Here one should observe, that the probability of a/p having a short period is nearly zero, because there are only a few primes q having a short period in 1/q ([Horster and Isselhorst 89, p. 89]).

Remark

Now it is possible to use the fraction a/p with $2 \cdot \log_{10}(p)$-50 digits as a public key, because it is impossible to compute a and p having not enough information about a/p.

THE PROPOSED PUBLIC-KEY CRYPTOSYSTEM

Knowing the results about fractions we look for a way to use them for building a public-key cryptosystem. One possibility to do this is based on the computation with a real modulus:

$$a \equiv b \pmod{c} \Leftrightarrow (a-b)/c \in \mathbb{N},\ a,b,c \in \mathbb{R}^+$$

$$a \text{ MOD } b := a - \lfloor a/b \rfloor \cdot b$$

The following lemma combines the results about fractions with a real modulus.

Lemma

Let p be a prime, $t > 0$, a, $a^* \in [1:p-1]$ with $a \cdot a^* \equiv 1 \pmod{p}$ and denote

$$c := t \cdot a/p$$

$$E(x) := (c \cdot x) \text{ MOD } t,\ x \in [0:p-1]$$

$$D(y) := (y/t \cdot p \cdot a^*) \text{ MOD } p$$

then $D(E(x)) = x$ for all $x \in [0{:}p-1]$.

Proof:

Since $a/r \text{ MOD } b/r = a/r - \lfloor (a/r)/(b/r) \rfloor \cdot b/r = (a \text{ MOD } b)/r$ for all $r > 0$ we get

$$\begin{aligned} D(E(x)) &= ((\, t \cdot a/p \cdot x \text{ MOD } t\,)\, /t \cdot p \cdot a^*\,) \text{ MOD } p \\ &= ((\, t/p \cdot (a \cdot x \text{ MOD } p))\, / t \cdot p \cdot a^*) \text{ MOD } p \\ &= (a \cdot x \text{ MOD } p) \cdot a^* \text{ MOD } p = x. \; \square \end{aligned}$$

This can be interpreted as a model of a cryptosystem, which uses the fraction a/p in the encryption function $E(x)$, but uses the integer $p \cdot a^*$ in the decryption function $D(y)$. Here it is important to see, that the integer $p \cdot a^*$ is not the same as the fraction p/a.

However it works only if one uses exact arithmetic, it is possible to get a and p knowing c. But when the system is made fault tolerant with rounded numbers, it can be secured and implemented using the results above.

So the lemma can be improved to a public-key cryptosystem, which will be discussed here:

The R^k-System

Assumptions: Let
- p be a prime, $p > 10^{250}$, $k \in \mathbb{N}+2$
- $A = (a_{i,j}) \in \mathbb{Z}_p^{k \times k}$, $\det(A) \not\equiv 0 \pmod p$
- $A^* = (a^*_{i,j}) \in \mathbb{Z}_p^{k \times k}$ with $A \cdot A^* \equiv I \pmod p$
- $t \in (0,p)$ (for example $t = 1$)
- $z = \lceil \log_{10}(4 \cdot p/t) \rceil$
- $c^n_{i,j} = 10^{-n} \lfloor 10^n \cdot t \cdot a_{i,j} / p \rfloor$, $C_n = (c^n_{i,j}) \in \mathbb{R}^{k \times k}$ with $n = \lceil 2 \cdot \log_{10}(p) - 50 - \log_{10}(t) \rceil$

Plaintext: - $X \in \mathbb{Z}_m^k$ with $m = \left\lfloor \dfrac{p}{10^{50} \cdot 4k} \right\rfloor$

Encryption function: $E(X) = 10^{-z} \lfloor 10^z \cdot ((C_n \cdot X) \text{ MOD } t) \rfloor$

Public keys: $\quad - C_n, t, z$

Decryption function: $\quad D(Y) = (A^{*} \cdot \lfloor Y \cdot p/t + 1/2 \rfloor) \text{ MOD } p$

Secret keys: $\quad - p, A^{*}$

Theorem 2

The R^k-system holds $D(E(X)) = X$ for all $X \in \mathbb{Z}_m^k$.

Proof (sketch):

The central step is

$$0 \le \left[\left(\sum_{j=1}^{k} c_{i,j}^{n} \cdot x_j\right) \text{MOD } t\right] \cdot \frac{p}{t} - \left[10^{-z} \lfloor 10^{z} \left(\sum_{j=1}^{k} c_{i,j}^{n} \cdot x_j\right) \text{MOD } t \rfloor\right] \cdot \frac{p}{t} \le \frac{1}{4}.$$

With [Isselhorst 88, p. 131-134] we also have

$$\left| \left[\left(\sum_{j=1}^{k} t \cdot a_{i,j} / p \cdot x_j\right) \text{ MOD } t\right] \cdot p/t - \left[\left(\sum_{j=1}^{k} c_{i,j}^{n} \cdot x_j\right) \text{MOD } t\right] \cdot \frac{p}{t} \right| =$$

$$\left| \left(\sum_{j=1}^{k} a_{i,j} \cdot x_j\right) \text{MOD } p - \left[\left(\sum_{j=1}^{k} c_{i,j}^{n} \cdot x_j\right) \text{MOD } t\right] \cdot \frac{p}{t} \right| < 1/4.$$

Taking both inequalities together implies

$$\left| \left(\sum_{j=1}^{k} a_{i,j} \cdot x_j\right) \text{MOD } p - \left[10^{-z} \lfloor 10^{z} \left(\sum_{j=1}^{k} c_{i,j}^{n} \cdot x_j\right) \text{MOD } t \rfloor\right] \cdot \frac{p}{t} \right| <$$

$$\left| \left(\sum_{j=1}^{k} a_{i,j} \cdot x_j\right) \text{MOD } p - \left[\left(\sum_{j=1}^{k} c_{i,j}^{n} \cdot x_j\right) \text{MOD } t\right] \cdot \frac{p}{t} \right| +$$

$$\left| \left[\left(\sum_{j=1}^{k} c_{i,j}^{n} \cdot x_j\right) \text{MOD } t\right] \cdot \frac{p}{t} - \left[10^{-z} \lfloor 10^{z} \left(\sum_{j=1}^{k} c_{i,j}^{n} \cdot x_j\right) \text{MOD } t \rfloor\right] \cdot \frac{p}{t} \right| < \frac{1}{4} + \frac{1}{4} = \frac{1}{2},$$

and finally $\left(\sum_{j=1}^{k} a_{i,j} \cdot x_j\right) \text{MOD } p = \lfloor \left[10^{-z} \lfloor 10^{z} \left(\sum_{j=1}^{k} c_{i,j}^{n} \cdot x_j\right) \text{MOD } t \rfloor\right] \cdot \frac{p}{t} + \frac{1}{2} \rfloor$. □

DISCUSSION

a) **SECURITY**

i) The R^k-system with parameter k=1 is not secure. It is easy to approximate the number $c_n/t \approx e/f$ by continued fractions. Then one can simulate the original

R^1-system with e/f instead of a/p. So one can break a R^1-system without knowing the secret keys a and p.-

ii) With $k \geq 2$ one can try the same attack: one looks for an approximation of $c_{i,j}^{n}/t \approx e_{i,j}/f$, which is a simultaneous diophantine approximation.

But note the following facts:

- the number of simultaneous diophantine approximations increases quadratically with k
- the error bound is always very small ($\sim 10^{50}/p^2$, [Horster and Isselhorst 89])
- the common denominator f has to be bounded: $f \leq 10 \cdot p$.

Furthermore the best algorithm to solve simultaneous diophantine approximation problems of this kind would in my opinion Lagarias' algorithm [Lagarias 85], which uses $O(k^{12} \cdot (k^2 \cdot \log_2(10^n) + \log_2(p))^4)$ bit-operations to find *some* approximation. It is not guaranteed that solutions found by this procedure will work.

b) **ADVANTAGE**

i) The advantage of the R^k-system is, that it works fast. To encrypt and decrypt k integers out of [0:m] ($m \approx p \cdot 10^{-50}$) there are $(2k^2+10k)$ operations like addition, multiplication and reduction. (With k=10, t=1 there are 9 additions, 10 multiplications and 1 reduction to encrypt or decrypt one number). It is possible to choose t=1, so that the reduction mod t is very simple.

ii) It is easy to initialize the R^k-system and to generate new keys, because one needs only one prime p and an invertible matrix A with the $\mathbb{Z}_p$-invers A^*.

iii) To strengthen the system one can select a higher dimension k without the need to use larger numbers as in the RSA-scheme.

iv) The R^k-system provides another way to build a public-key cipher without using the well known arithmetical problems like factorization or knapsacks.

c) **DISADVANTAGE**

i) The security of the R^k-system is not proved.

ii) The size of the public and the secret key might be regarded as a disadvantage. But unlike knapsack-schemes within the R^k-system one encrypts $\log_2(m)/k >> 1$ bits with every component of the key.

FURTHER RESEARCH

i) Look for other attacks for the R^k-system: One is to try to get the prime p with a simultaneous diophantine approximation with only a few components of the key matrix C_n.

ii) Examine if the security of the R^k-system holds when p is an arbitrary integer and not necessarily a prime, and $\det(A) \not\equiv 0 \pmod p$. So the initialization becomes easier.

iii) Examine if one can select a small number $k \in [2:10]$, such that the R^k-system is very fast. This should be used for messages which have to be secret only for a short time (like one hour or one day e. g. in military use).

CONCLUSION

The paper shows how to use fractions in a public-key cryptosystem, which is based on the problem of a simultaneous diophantine approximation with many parameters. The new R^k-system can be implemented in a fast way using only addition and multiplication with only one reduction. Also new keys can be produced very simply, so that one can use a different pair of keys in every communication.

REFERENCES

[Horster and Isselhorst 89]:

P. Horster, H. Isselhorst, Approximative Public-Key-Kryptosysteme, Informatik-Fachberichte 206 (Heidelberg: Springer, 1989)

[Isselhorst 88]:
H. Isselhorst, Ein Beitrag zur Verwendung rationaler Zahlen in Public-Key Kryptosystemen (Heidelberg: Hüthig, 1988)

[Knuth 81]:
D. E. Knuth, The Art of Computer Programming Vol.2. Seminumerical Algorithms, 2ed (Reading: Addison Wesley, 1981)

[Lagarias 85]:
J. C. Lagarias, "The Computational Complexity of Simultaneous Diophantine Approximation Problems," SIAM J. Comput. Vol.14 No1 (1985), p.196-209

[Niven and Zuckerman 76]:
I. Niven, H. S. Zuckerman, Einführung in die Zahlentheorie I (Mannheim: Bibliographisches Institut, 1976)

SYMBOLS

$\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ $\mathbb{N}+i = \{i, i+1, i+2, i+3, \ldots\}$

$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ $\mathbb{Z}_p = \{0, 1, 2, \ldots, p-2, p-1\}$

$\mathbb{R}$ = real numbers

$\lfloor x \rfloor$ = greatest integer less than x $\lceil x \rceil$ = lowest integer greater than x

gcd(a,b) = greatest common divisor of a and b

$[a:b] = \{a, a+1, a+2, \ldots, b-1, b\}$

I = unit matrix

SMALL EXAMPLE

The prime: $p = 64301$

The dimension $k = 2$

The invertible matrix $A = \begin{pmatrix} 5387 & 2993 \\ 7461 & 4001 \end{pmatrix}$

The inverse matrix $A^* = \begin{pmatrix} 14109 & 19322 \\ 59703 & 52039 \end{pmatrix}$

the modulus $t = 1$

The constants $n = 9,\ z = 6$

The key-matrix $C_n = \begin{pmatrix} 0{,}083777857 & 0{,}0465467 \\ 0{,}116032410 & 0{,}06222298 \end{pmatrix}$

Plaintext $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{Z}^2_{1000}$, $m = 1000$

Encryption $E\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 10^{-6} \cdot \lfloor 10^{6} \cdot \{ (C_9 \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}) \text{ MOD } 1 \} \rfloor$

Decryption: $D\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = A^{-1} \cdot \lfloor \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \cdot 64301 + 1/2 \rfloor \text{ MOD } 64301$

$$E\begin{pmatrix} 500 \\ 501 \end{pmatrix} = \begin{pmatrix} 0{,}20883 \\ 0{,}189918 \end{pmatrix}, \quad D\begin{pmatrix} 0{,}208830 \\ 0{,}189918 \end{pmatrix} = \begin{pmatrix} 500 \\ 501 \end{pmatrix}$$