An Applicable Public-Key-Cryptosystem Based On NP-Complete Problems

Björn Grohmann

nn@mhorg.de

1 Introduction

The expected security of some proposed Public-Key-Cryptosystems relys on the expected hardness of problems with the following pattern. Given a field $\mathbb{F}$, a matrix $A \in \mathbb{F}^{n \times m}$, a vector $\mathbf{b} \in \mathbb{F}^n$ and a set of vectors $\mathcal{S} \subseteq \mathbb{F}^m$. The question is: does there exist a vector $\mathbf{x} \in \mathbb{F}^m$ such that

$$A\mathbf{x} = \mathbf{b} \quad \text{and} \quad \mathbf{x} \in \mathcal{S}, \tag{1}$$

and if so, can $\mathbf{x}$ be computed efficiently?

In fact, depending on the field $\mathbb{F}$ and the set $\mathcal{S}$, there is a long list of problems that turn out to be NP-hard (resp. NP-complete), like e.g. SPARSE-APPROXIMATION, SYNDROME-DECODING, etc.

There is a strong hope that PKCs based on NP-hard problems will survive against attacks by Quantum-Computers in the future. One of the main challenges, besides security, in designing these systems is to make them efficient.

In this article, we will propose a (to the best of the authors' knowledge) new type of Public-Key-Cryptosystem, based on the pattern described above. The next section gives a detailed description of the system followed by an analysis of its performance and its expected security.

The proposed PKC has two main advantages that will make it applicable to the real world. The first one is a decoding algorithm with an expected runtime of $\mathbf{O}(n)$ elementary operations. The second advantage is that the corresponding matrix A is of the form $A = [A' \mid I]$, where I is the identity matrix and A' being completely random, in contrast to e.g. McEliece-Type systems (cf. [1]), which means that these entries can be replaced by the values of any (PRNG-) function and therefore the public key can be reduced to having linear size.

2 An Applicable Public-Key-Cryptosystem

We start by fixing some notation. Let $\mathbb{Z}$ be the set of integers. For a prime $p > 2$, the finite field with p elements will be denoted by $\mathbb{F}_p$ and its subgroup of non-zero elements by $\mathbb{F}_p^\times$. We will use a representation of elements of $\mathbb{F}_p$ of the form $\mathbb{F}_p = \{-(p-1)/2, \ldots, (p-1)/2\}$ and we will frequently view integers as elements of $\mathbb{F}_p$ and vice versa, if the context allows this. All vectors $\mathbf{x} \in \mathbb{F}_p^n$ will be viewed as column vectors, the transpose of a vector $\mathbf{x}$ will be denoted by $\mathbf{x}^T$. For two vectors $\mathbf{x} = (x_i)_i^T$ and $\mathbf{y} = (y_i)_i^T$ we denote their (inner) product by $\mathbf{x}^T\mathbf{y} = \sum_i x_i y_i$. For two integers s and t, with $s \leqslant t$, we will write $\langle s, t\rangle^n$ to denote the set of vectors $\mathbf{x}^T = (x_1, \ldots, x_n) \in \mathbb{Z}^n$ with $s \leqslant x_i \leqslant t$, for $i = 1, \ldots, n$, so, by abuse of notation, $\mathbb{F}_p^n = \langle -(p-1)/2, (p-1)/2\rangle^n$. Finally, the number of elements of a finite set $\mathcal{S}$ will be denoted by $|\mathcal{S}|$.

We now come to the description of the Public-Key-Cryptosystem. For this, Alice fixes a positive integer n, a prime p of size $\sim 2^{\sqrt{n \log n}}$, a (PRNG-) function $f : \mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{F}_p$ and a cryptographic hash-function $h : \mathbb{F}_p \longrightarrow \mathbb{F}_p$.

Her **secret key** will consist of a random element $\alpha \in \mathbb{F}_p$ and two randomly chosen (bit-) vectors $\mathbf{e} = (e_1, \ldots, e_n)^T$ and $\mathbf{e}' = (e'_1, \ldots, e'_n)^T$, with $e_i, e'_i \in \{0, 1\}$, for $i = 1, \ldots, n$.

She then computes for $i = 1, \ldots n$:

$$\epsilon_i = \left(\sum_{j=1}^{n} f(i, j) e_j \right) + \alpha e'_i. \tag{2}$$

Her **public key** consist of the data: n, p, f, h and $\epsilon = (\epsilon_1, \ldots, \epsilon_n)^T$.

Now, if Bob wants to send a **message** $\beta \in \mathbb{F}_p$ to Alice, he randomly selects two (bit-) vectors $\mathbf{d} = (d_1, \ldots, d_n)^T$ and $\mathbf{d}' = (d'_1, \ldots, d'_n)^T$, with $d_i, d'_i \in \{0, 1\}$, for $i = 1, \ldots, n$, and computes for $j = 1, \ldots, n$:

$$\delta_j = \left(\sum_{i=1}^{n} f(i, j) d_i \right) + \beta d'_j \tag{3}$$

as well as

$$\tau_B = \left(\sum_{i=1}^{n} \epsilon_i d_i \right) - \beta \tag{4}$$

and **sends** $\delta = (\delta_1, \ldots, \delta_n)^T$, τ_B and $h(\beta)$ to Alice.

To **decode** the message, Alice first computes

$$\tau_A = \left(\sum_{i=1}^{n} \delta_i e_i \right) - \tau_B \tag{5}$$

and then starts the following iteration process:

First, she picks integers $s_0 = \lfloor n/4 \rfloor$ and $t_0 = \lfloor n/4 \rfloor + 1$. At each step $k = 0, 1, 2, \ldots$, she tests if

$$h\left(\frac{\tau_A - s_k \alpha}{t_k} \right) = h(\beta), \tag{6}$$

where all computations take place in $\mathbb{F}_p$. If equation (6) holds, she saves $(\tau_A - s_k\alpha)/t_k \in \mathbb{F}_p$ and stops the process. Else, she puts (s_{k+1}, t_{k+1}) to be one of the remaining (integer) points having a small distance to the center (s_0, t_0), for example:

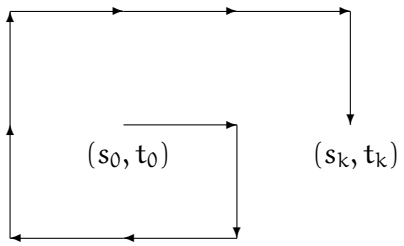

Figure I: "Snake-Decoding"

We will start the analysis by showing that this decoding procedure is in general quite efficient:

Theorem 1 *The decoding algorithm given above produces the message* β *with a probability close to* 1 *within an expected runtime of* $\mathbf{O}(n)$ *steps.*

Proof. The correctness of the Theorem follows from the fact that $\tau_A = s'\alpha + t'\beta$, with $s' = \sum_i e'_i d_i$ and $t' = 1 + \sum_i e_i d'_i$. Since the expected value of s' is $n/4$ (resp. $1 + n/4$ for t') and their variance is $\sqrt{n}\sqrt{3}/4$, the Theorem follows. □

Next, we will focus on the (expected) security of the system. For this, let $A \in \mathbb{F}_p^{n \times m}$ be any matrix and $\mathbf{b} \in \mathbb{F}_p^n$ a vector. We take a closer look at the solutions $(\mathbf{x}, \mathbf{y}, \lambda)$ of the equation

$$A\mathbf{x} + \lambda \mathbf{y} = \mathbf{b}, \tag{7}$$

with $\mathbf{x} \in \mathbb{F}_p^m$, $\mathbf{y} \in \mathbb{F}_p^n$ and $\lambda \in \mathbb{F}_p$.

In case of $n = m$, $A = (f(i,j))_{i,j}$ and $\mathbf{b} = \epsilon$ it is clear that any solution $(\mathbf{x}, \mathbf{y}, \lambda)$ of equation (7) leads to an effective (but not necessarily efficient) decoding algorithm of the system, since

$$\mathbf{x}^T\delta - \tau_B = \mathbf{d}^T\mathbf{y}\lambda + (1 + \mathbf{x}^T\mathbf{d}')\beta. \tag{8}$$

We may, for example, take $\lambda = 0$ and assume that the matrix A is invertible. Then, putting $x_0 = A^{-1}\epsilon$ would lead to

$$\mathbf{x}_0^T\delta - \tau_B = (1 + \mathbf{x}_0^T\mathbf{d}')\beta, \tag{9}$$

but since $\mathbf{d}'$ is a random bit-vector there is, in general, no efficient way to compute β without any knowledge of $\mathbf{d}'$.

As we have already seen, bit-vectors do the trick, although these are not the only solutions which may lead to efficient decoding algorithms. Before we come to a general description of suitable solutions, we note down the following

Theorem 2 *Given a prime $p > 2$, positive integers n, m, a matrix $A \in \mathbb{F}_p^{n\times m}$ and a vector $\mathbf{b} \in \mathbb{F}_p^n$. Deciding, whether there exist vectors $\mathbf{x} \in \{0,1\}^m$, $\mathbf{y} \in \{0,1\}^n$ and an element $\lambda \in \mathbb{F}_p$, such that $A\mathbf{x} + \lambda\mathbf{y} = \mathbf{b}$, is NP-complete.*

Proof. This is a special case of Theorem 4, see below. □

To describe those solutions that might lead to an efficient decoding algorithm of the system, we denote by

$$X_p = \varprojlim_n \mathbb{F}_p^n \tag{10}$$

the projective limit (see [2]) of the vector spaces $\mathbb{F}_p^n$, $n = 1, 2, \ldots$, equipped with the natural embeddings $\iota_n : \mathbb{F}_p^n \hookrightarrow X_p$. We further define a counting function

$$\kappa : X_p \longrightarrow \mathbb{Z} \cup \{\infty\} \tag{11}$$

in the following way: Let $\overline{\mathbf{x}} \in X_p$. If there exists a positive integer m and a vector $\mathbf{x} \in \mathbb{F}_p^m$, such that $\iota_m(\mathbf{x}) = \overline{\mathbf{x}}$, then

$$\kappa(\overline{\mathbf{x}}) = \left|\left\{\mathbf{x}^T\mathbf{c} \mid \mathbf{c} \in \{0,1\}^m\right\}\right|, \tag{12}$$

else, we set $\kappa(\overline{\mathbf{x}}) = \infty$.

Note, that for $\lambda \in \mathbb{F}_p^\times$ we have $\kappa(\lambda\overline{\mathbf{x}}) = \kappa(\overline{\mathbf{x}})$ and also $\kappa(\overline{\mathbf{x}} + \overline{\mathbf{y}}) \leqslant \kappa(\overline{\mathbf{x}})\kappa(\overline{\mathbf{y}})$,

whenever $\kappa(\overline{\mathbf{x}}), \kappa(\overline{\mathbf{y}}) \notin \{0, \infty\}$. In what follows, we will always write $\kappa(\mathbf{x})$ instead of $\kappa(\iota_m(\mathbf{x}))$ for a vector $\mathbf{x} \in \mathbb{F}_p^m$.

It is obvious that the computation of κ is a rather "difficult task" in general, even for finite dimensions:

Theorem 3 *Given a prime $p > 2$, positive integers n, t and a vector $\mathbf{x} \in \mathbb{F}_p^n$. Deciding, whether $\kappa(\mathbf{x}) < t$, is NP-hard and coNP-hard.*

Proof. To begin with, it is clear that an algorithm that can efficiently decide whether $\kappa(\mathbf{x}) < t$, for any t, can be used to efficiently compute $\kappa(\mathbf{x})$ itself.
Next, we shall use a well known map from Boolean functions to a set of integers (see [3]): let $F = (v_{00} \vee v_{01} \vee v_{02}) \wedge \cdots \wedge (v_{(k-1)0} \vee v_{(k-1)1} \vee v_{(k-1)2})$, with $v_{ij} \in \{x_0, \ldots, x_{m-1}\} \cup \{\neg x_0, \ldots, \neg x_{m-1}\}$, be a Boolean function in 3-conjunctive normal form, having k clauses and m variables. We may further assume w.l.o.g., that each variable appears at most once in each clause.
We start by defining integers X_{ij} (resp. Y_{ij}), for $i = 0, \ldots, m-1$ and $j = 0, \ldots, k-1$, in the following way: whenever "x_i" (resp. "$\neg x_i$") appears in the j-th clause, we put $X_{ij} = 10^{j+m}$ (resp. $Y_{ij} = 10^{j+m}$), else we set $X_{ij} = 0$ (resp. $Y_{ij} = 0$).
We further define for each i integers $a_i = 10^i + \sum_j = X_{ij}$ and $b_i = 10^i + \sum_j = Y_{ij}$ and for each j integers $c_j = 10^{j+m}$, $d_j = 2c_j$. The last integer we define will be $e = 2\sum_{j=0}^{k-1} d_j + \sum_{i=0}^{m-1} 10^i$.
Finally, we define vectors $\mathbf{x} = (a_0, \ldots, a_{m-1}, b_0, \ldots, b_{m-1}, c_0, \ldots, c_{k-1}, d_0, \ldots, d_{k-1})^T$ and $\mathbf{x_e} = (a_0, \ldots, a_{m-1}, b_0, \ldots, b_{m-1}, c_0, \ldots, c_{k-1}, d_0, \ldots, d_{k-1}, e)^T$ and we may assume that $p > 10^{m+k}$ and therefore view $\mathbf{x}$ (resp. $\mathbf{x_e}$) as an element of $\mathbb{F}_p^{2(m+k)}$ (resp. $\mathbb{F}_p^{2(m+k)+1}$).
After all this preparation and after what has been said, it is now easy to see, that F is satisfiable if and only if there exists a vector $\mathbf{z} \in \{0,1\}^{2(m+k)}$ such that $\mathbf{x}^T\mathbf{z} = e$ if and only if $\kappa(\mathbf{x_e}) < 2\kappa(\mathbf{x})$. This implies the NP-hardness. But since $\kappa(\mathbf{x_e}) \leqslant \kappa(e)\kappa(\mathbf{x}) = 2\kappa(\mathbf{x})$ the coNP-hardness also follows. □

Notabene. Please note that the statement of the last Theorem does not imply that any problem involving "κ" is automatically a (NP-) hard problem. For example: "Given $\mathbf{x} \in \mathbb{F}_p^n$, decide if $\kappa(\mathbf{x}) < n$" is not likely to be NP-hard. Why? Because this problem is in coNP and therefore its NP-hardness would imply "NP=coNP".

The classification of vectors $\mathbf{x}$ with "small $\kappa(\mathbf{x})$" appears to be a difficult task. In what follows, we will try to tackle this problem from two sides, but before that, we state the following

Definition 1 *Let l be a positive integer, $A \in \mathbb{F}_p^{n \times m}$ be a matrix and $\mathbf{b} \in \mathbb{F}_p^n$ a vector. We*

say that a solution $(\mathbf{x}, \mathbf{y}, \lambda)$ *of the equation*

$$A\mathbf{x} + \lambda\mathbf{y} = \mathbf{b}, \tag{13}$$

with $\mathbf{x} \in \mathbb{F}_p^m$, $\mathbf{y} \in \mathbb{F}_p^n$ *and* $\lambda \in \mathbb{F}_p$, *has* **level** l, *if*

$$(nm)^{l-1} < \max\left(\kappa(\mathbf{x}), \kappa(\lambda\mathbf{y}), \kappa(\mathbf{x})\kappa(\lambda\mathbf{y})\right) \leqslant (nm)^l. \tag{14}$$

It is clear that a "low-" or "moderate-level-solution" is a necessary (but not necessarily sufficient) condition for the decoding algorithm described above to work efficiently. On the other hand, as we have seen in the preceding Theorem, the determination of the level of a solution can be quite hard.

Nevertheless, there are solutions that can be classified quite easily, for example, all solutions of the form $(\mathbf{x}, \mathbf{y}, \lambda)$, with $\mathbf{x} \in \{0,1\}^m$, $\mathbf{y} \in \{0,1\}^n$ and $\lambda \in \mathbb{F}_p$ have level 1, and we know from Theorem 2 that they are, in general, hard to compute. The same holds for example for **sparse** solutions, i.e. solutions where most of the entries of $\mathbf{x}$ and $\mathbf{y}$ are zero. Since this is a well studied problem (cf. the SPARSE-APPROXIMATION-PROBLEM, which is known to be NP-hard in general) we may refer the reader to the literature.

There is yet a larger class of solutions, whose level can be reasonably bounded from above, but nevertheless are not easy to compute in general:

Theorem 4 *Given a prime* $p > 2$, *positive integers* n, m, *a matrix* $A \in \mathbb{F}_p^{n\times m}$ *and a vector* $\mathbf{b} \in \mathbb{F}_p^n$. *Deciding, whether there exist vectors* $\mathbf{x} \in \langle -r, r\rangle^m$, $\mathbf{y} \in \langle -r, r\rangle^n$, *with a positive integer* $r < \sqrt{p}$, *and an element* $\lambda \in \mathbb{F}_p$, *such that* $A\mathbf{x} + \lambda\mathbf{y} = \mathbf{b}$, *is NP-complete.*

Proof. Again, let $F = (v_{00} \vee v_{01} \vee v_{02}) \wedge \cdots \wedge (v_{(k-1)0} \vee v_{(k-1)1} \vee v_{(k-1)2})$, where $v_{ij} \in \{x_0, \ldots, x_{m-1}\} \cup \{\neg x_0, \ldots, \neg x_{m-1}\}$, be a Boolean function in 3-conjunctive normal form, with k clauses and m variables, and let us again assume, that each variable appears at most once in each clause.
We will define a matrix $A = (a_{ij})_{i,j} \in \mathbb{F}_p^{(k+2)\times m}$ and a vector $\mathbf{b} = (b_0, \ldots, b_{k+1})^T \in \mathbb{F}_p^{(k+2)}$ in the following way: the i-th row of A, where $i = 0, \ldots, k-1$, correponds to the i-th clause of F in a sense that if "x_j" appears in that clause, we put $a_{ij} = 1$, if "$\neg x_j$" is in the clause, we set $a_{ij} = -1$, else $a_{ij} = 0$. We further set $b_i = r + 1 - s_i$, where $s_i \in \{0,1,2,3\}$ denotes the number of negative literals of clause i.
The last two rows of A are set to zero, while we define $b_k = 1$ and $b_{k+1} = r$.
Assuming p to be large enough, we now show that F is satisfiable if and only if the equation $A\mathbf{x} + \lambda\mathbf{y} = \mathbf{b}$ has a solution of the required form:
If we suppose that F is satisfiable with an assignment $\hat{\mathbf{x}} = (\hat{x}_0, \ldots, \hat{x}_{m-1})^T \in \{\text{TRUE}, \text{FALSE}\}^m$, then, setting $\mathbf{x} = (x_0, \ldots, x_{m-1})^T$, with $x_j = 1$, if $\hat{x}_j = \text{TRUE}$,

else $x_j = 0$, for $j = 0, \ldots, m-1$, $\mathbf{y} = (y_0, \ldots, y_{k-1}, 1, r)^T$, with $y_i = b_i - \sum_j a_{ij}x_j$, for $i = 0, \ldots, k-1$, and $\lambda = 1$, leads to a solution of the given equation.
If, on the other hand, we are given a solution $(\mathbf{x}, \mathbf{y}, \lambda)$ of the required form it then follows from $b_k = 1$, $b_{k+1} = r$ and $r < \sqrt{p}$ that $\lambda = \pm 1$ and therefore, if we define $\hat{\mathbf{x}} = (\hat{x}_0, \ldots, \hat{x}_{m-1})^T$ such that $\hat{x}_j$ = TRUE, if $x_j \in \{1, \ldots, r\}$, else $\hat{x}_j$ = FALSE, for $j = 0, \ldots, m-1$, this leads to $F(\hat{\mathbf{x}}) =$ TRUE, by construction of the matrix A. □

As is easily seen, for all vectors $\mathbf{x} \in \langle -r, r\rangle^n$, we have $\kappa(\mathbf{x}) \leqslant 2rn + 1$. At least for small r, this is quite moderate. Let for a moment, $c > 0$ denote an integer constant, and let us further call a vector $\mathbf{x} \in \mathbb{F}_p^n$ **smooth**, if $\mathbf{x} \in \langle -c, c\rangle^n$. It should be clear that "smooth solutions" lead to efficient decoding algorithms, but as the preceding Theorem shows, are hard to compute in general. As already mentioned above, the same is true for "sparse solutions". It therefore seems reasonable to believe that a combination of both will cover a large part of the class of "moderate-level-solutions", i.e. those solutions that would challenge the security of our system.

Theorem 5 *Let $c, t > 0$ be integer constants. Given a prime $p > 2$, positive integers n, m, a matrix $A \in \mathbb{F}_p^{n\times m}$ and a vector $\mathbf{b} \in \mathbb{F}_p^n$. Deciding, whether there exist vectors $\mathbf{x} = \sum_{i=1}^t \alpha_i \mathbf{x}_i$ and $\mathbf{y} = \sum_{i=1}^t \beta_i \mathbf{y}_i$, with $\alpha_i, \beta_i \in \mathbb{F}_p$, $\mathbf{x}_i \in \langle -c, c\rangle^m$, $\mathbf{y}_i \in \langle -c, c\rangle^n$, for $i = 1, \ldots, t$, and an element $\lambda \in \mathbb{F}_p$, such that $A\mathbf{x} + \lambda\mathbf{y} = \mathbf{b}$, is NP-complete.*

Proof. With the knowledge of the previous Theorem, the proof is quite simple: all we need to do is to "force" the vectors $\mathbf{x}$ and $\mathbf{y}$ to match the pattern of Theorem 4. For this, set $r = ((2c+1)^t - 1)/2$. Since c and t are constant, we may assume that $r < \sqrt{p}$. We now extend the matrix $A \in \mathbb{F}_p^{(k+2)\times m}$ and the vector $\mathbf{b} = (b_0, \ldots, b_{k+1})^T \in \mathbb{F}^{(k+2)}$ from the proof of Theorem 4 to a matrix

$$A' = \left(\begin{array}{c|c} A & 0 \\ \hline 0 & I_{2r} \\ \hline 0 & 0 \end{array}\right) \in \mathbb{F}_p^{(k+2+2r+2r)\times(m+2r)}, \tag{15}$$

where I_{2r} is the identity matrix of rank $2r$, and a vector

$$\mathbf{b}' = (b_0, \ldots, b_{k+1}, -r, \ldots, -1, 1, \ldots, r, -r, \ldots, -1, 1, \ldots, r)^T \in \mathbb{F}_p^{(k+2+2r+2r)}. \tag{16}$$

These modifications make sure that for any solution $(\mathbf{x}, \mathbf{y}, \lambda)$, we have $\mathbf{x} \in \langle -r, r\rangle^{m'}$ and $\lambda\mathbf{y} \in \langle -r, r\rangle^{n'}$, with $m' = m + 2r$ and $n' = k + 2 + 2r + 2r$. The rest runs analogous to the proof of Theorem 4. □

Let us summerize the discussion:

• We have presented a Public-Key-Cryptosystem with an efficient decoding algorithm, where all of its main components can be chosen completely at random.

• It has been proven, that a direct computation of Alice' secret key, resp. Bobs private data is in general a NP-complete task.

• It has been shown that a necessary conditon for other solutions to work, is that they have to have a small "level", and that the computation of a "level" itself can turn out to be a (very) hard problem in general.

• We have started to analyse the class of "moderate-level-solutions" and it has been shown that the problem of computing "smooth solutions", "sparse solutions" or a combination of both is in general NP-complete.

References

[1] McEliece, J.R.: A public-key cryptosystem based on algebraic coding theory. In: Deep Space Network Progress Report 42-44, Jet Propulsion Lab, California Institute of Technology, pp. 114-116 (1978)

[2] Neukirch, J.: Algebraische Zahlentheorie. Springer, Berlin, Heidelberg (1992)

[3] Schoening, U: Theoretische Informatik kurz gefasst. BI-Wissenschaftsverlag, Mannheim (1992)

[4] Shor, P.W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. arXiv:quant-ph/9508027 (1995)