# Wild McEliece Incognito

Daniel J. Bernstein[1], Tanja Lange[2], and Christiane Peters[3]

[1] Department of Computer Science
University of Illinois at Chicago, Chicago, IL 60607–7045, USA
djb@cr.yp.to
[2] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, Netherlands
tanja@hyperelliptic.org
[3] Department of Mathematics
Technical University of Denmark, 2800 Kgs. Lyngby, Denmark
c.p.peters@mat.dtu.dk

**Abstract.** The wild McEliece cryptosystem uses wild Goppa codes over finite fields to achieve smaller public key sizes compared to the original McEliece cryptosystem at the same level of security against all attacks known. However, the cryptosystem drops one of the confidence-inspiring shields built into the original McEliece cryptosystem, namely a large pool of Goppa polynomials to choose from.
This paper shows how to achieve almost all of the same reduction in key size while preserving this shield. Even if support splitting could be (1) generalized to handle an unknown support set and (2) sped up by a square-root factor, polynomial-searching attacks in the new system will still be at least as hard as information-set decoding.
Furthermore, this paper presents a set of concrete cryptanalytic challenges to encourage the cryptographic community to study the security of code-based cryptography. The challenges range through codes over $\mathbf{F}_2, \mathbf{F}_3, \ldots, \mathbf{F}_{32}$, and cover two different levels of how much the wildness is hidden.

**Keywords:** McEliece cryptosystem, Niederreiter cryptosystem, Goppa codes, wild Goppa codes, list decoding

## 1 Introduction

The McEliece cryptosystem [15] is based on classical Goppa codes (corresponding to genus-0 AG codes) over $\mathbf{F}_2$. A code is built using a Goppa

* This work was supported by the Cisco University Research Program, by the National Institute of Standards and Technology under grant 60NANB10D263, by the Danish Council for Independent Research under the Technology and Production Sciences (FTP) grant 11-105325, and by the European Commission under Contract ICT-2007-216676 ECRYPT II. This work was started while the third author was with Technische Universiteit Eindhoven and continued during her employment at the University of Illinois at Chicago. Permanent ID of this document: cd39ef08c48d12b29da6b9db66559c41. Date: 2011.09.15.

polynomial $g \in \mathbf{F}_{2^m}[x]$ for some integer $m$. For $\deg(g) = t$ the code can correct $t$ errors. Generalizations of the McEliece cryptosystem (or equivalently the Niederreiter cryptosystem [16]) using Goppa codes over larger fields $\mathbf{F}_q$ were investigated but not found to offer advantages for small $q$, since it was believed that for a code built from a polynomial $g$ of degree $t$ one could correct only $\lfloor t/2 \rfloor$ errors.

Peters showed in [18] that, despite this reduced error-correction capacity, codes over $\mathbf{F}_{31}$ offer advantages in key size compared to codes over $\mathbf{F}_2$ while maintaining the same security level against all attacks known. However, codes over smaller fields such as $\mathbf{F}_3$ were still not competitive in key size with codes over $\mathbf{F}_2$.

In [6] we introduced the "wild McEliece" cryptosystem, using Goppa codes over $\mathbf{F}_q$ built on polynomials of the form $g^{q-1}$. These codes have a better error-correction capacity: they can correct up to $\lfloor qt/2 \rfloor$ errors for $\deg(g) = t$. The extra factor $q/(q-1)$ makes "larger tiny fields" attractive and bridges the gap between $\mathbf{F}_2$ and $\mathbf{F}_{31}$. That paper contains cryptosystem parameters that minimize key size for different finite fields, subject to the requirement of achieving 128-bit security against information-set-decoding attacks.

This key-size optimization for 128-bit security reduces the number of irreducible polynomials $g$ below $2^{128}$ for $q \geq 11$, and below $2^{30}$ for $q \geq 31$. Enumerating all possibilities for $g$ thus becomes more efficient than performing information-set decoding. The parameters were intentionally chosen this way in [6]; otherwise the key-size benefit of wild McEliece would disappear as $q$ grows.

In McEliece's original proposal, a large space of possibilities for $g$ is the primary shield against structural attacks. There are secrets other than $g$, specifically a random support permutation $P$ and a random invertible matrix $S$, but Sendrier's support-splitting algorithm [21] quickly computes both $P$ and $S$ given $g$ and the public key. The cost of breaking McEliece's system is thus at most a small multiple of the number of choices of $g$: the attacker checks each possibility for $g$ with the support-splitting algorithm.

This attack fails against [6], because there is another shield in [6]: a secret support set. In McEliece's original proposal, the support set was all of $\mathbf{F}_{2^m}$; however, one can define Goppa codes using smaller support sets. We chose parameters in [6] so that there are more than $2^{256}$ possible support sets. There is no known attack against the McEliece system with secret support sets, even if the Goppa polynomial is *published*; in particular, the support-splitting algorithm uses the support set as input.

However, a secret support set has far less history than a secret choice of $g$, and therefore cannot inspire as much confidence. One can reasonably worry that there is a generalization of support-splitting that handles many support sets more efficiently than separately trying each possible support set. Parameters relying on secret support sets were marked with biohazard symbols in [6].

In this paper we hide the wild codes in two ways, achieving almost all of the key-size benefit of wild McEliece without sacrificing the confidence provided by a large space of polynomials. First, we consider codes built on polynomials $f \cdot g^{q-1}$; for $\deg(f) = s$ and $\deg(g) = t$ these codes can correct up to $\lfloor (s + qt)/2 \rfloor$ errors. A small extra factor $f$ makes the space of polynomials too large to search. Second, we use subcodes as suggested by Berger and Loidreau in [2]. The combination of these defenses leads to slightly larger key sizes but requires the attacker to see simultaneously through subcodes, secret support sets, and a huge set of polynomials.

This paper also announces a web page of code-based crypto challenges and presents some sample challenges. The challenges cover finite fields as large as $\mathbf{F}_{32}$ and start with training challenges that should be easy to break but still can show which attacks are faster than others. We originally considered issuing challenges with several different wildness percentages, ranging from 100% wild codes ($g^{q-1}$) to 50% wild codes ($fg^{q-1}$ with $\deg(f) \approx (q-1)\deg(g)$) and beyond, but we decided to focus on percentages close to 100%, since those are adequate to prevent polynomial enumeration. For each set of parameters, a public key and a ciphertext are presented.

**Acknowledgement:** The authors are grateful to Peter Beelen for interesting discussions and in particular for allowing us to use his suggestion of the extra factor $f$ as a way to hide the wildness of $g^{q-1}$.

## 2 An extra shield for wild Goppa codes

Fix a prime power $q$; a positive integer $m$; a positive integer $n \leq q^m$; an integer $t < n/m$; distinct elements $a_1, \ldots, a_n$ in $\mathbf{F}_{q^m}$; and a polynomial $g(x)$ in $\mathbf{F}_{q^m}[x]$ of degree $t$ such that $g(a_i) \neq 0$ for all $i$.

We denote the linear code consisting of all words $c = (c_1, \ldots, c_n)$ in $\mathbf{F}_{q^m}^n$ satisfying

$$\sum_{i=1}^{n} \frac{c_i}{x - a_i} \equiv 0 \pmod{g(x)} \tag{2.1}$$

by $\Gamma_{q^m}(a_1, \ldots, a_n, g)$; this is a special case of a generalized Reed–Solomon code over $\mathbf{F}_{q^m}$ having dimension $n - t$.

The *Goppa code* $\Gamma_q(a_1, \ldots, a_n, g)$ with *Goppa polynomial* $g(x)$ and *support* $a_1, \ldots, a_n$ is the restriction of $\Gamma_{q^m}(a_1, \ldots, a_n, g)$ to the field $\mathbf{F}_q$, i.e., the set of elements $(c_1, \ldots, c_n)$ in $\mathbf{F}_q^n$ that satisfy (2.1); this code $\Gamma_q(a_1, \ldots, a_n, g)$ has *dimension* at least $n - mt$ and *minimum distance* at least $t + 1$. These codes were introduced in [11] and [12].

Goppa codes can be decoded by any decoder for generalized Reed–Solomon codes. For example, Berlekamp's algorithm corrects $\lfloor t/2 \rfloor$ errors; see, e.g., [3]. Note that $t + 1$ is a lower bound for the minimum distance. There are Goppa codes whose minimum distance is much larger. Binary Goppa codes have minimum distance at least $2t + 1$ as shown in [11], and allow fast decoding of $t$ errors. The standard $t$-error decoding algorithm for binary Goppa codes, in the typical case that $g$ is monic and irreducible, is Patterson's algorithm from [17]. There are polynomial-time list-decoding algorithms that decode more errors; for more information and references see, e.g., [4], [1], and [5].

In this paper we use the McEliece cryptosystem, the Niederreiter cryptosystem, etc. with codes of the form $\Gamma_q(a_1, \ldots, a_n, fg^{q-1})$, where $f$ and $g$ are coprime squarefree monic polynomials.

If $g = 1$ then $\Gamma_q(a_1, \ldots, a_n, fg^{q-1})$ is the squarefree Goppa code $\Gamma_q(a_1, \ldots, a_n, f)$; these are, for $q = 2$, the traditional codes used in the McEliece cryptosystem. If $f = 1$ then $\Gamma_q(a_1, \ldots, a_n, fg^{q-1})$ is the *wild Goppa code* $\Gamma_q(a_1, \ldots, a_n, g^{q-1})$, which we proposed in [6] for the *wild McEliece cryptosystem*; what makes these codes interesting is that they can correct $\lfloor qt/2 \rfloor$ errors, or even slightly more using list decoding.

The Goppa code with polynomial $fg^{q-1}$ has dimension at least $n - m(s + (q - 1)t)$, where $s$ is the degree of $f$ and $t$ is the degree of $g$. Theorem 2.1 below says that $fg^q$ gives the same Goppa code. It follows that $\Gamma_q(a_1, a_2, \ldots, a_n, fg^{q-1})$ has minimum distance at least $s+qt+1$. One can plug $fg^q$ into (e.g.) the alternant decoder described in [6, Section 5] to efficiently decode $\lfloor (s + qt)/2 \rfloor$ errors, or into the list-decoder described in [5] to efficiently decode more errors.

Theorem 2.1 is a special case of a theorem of Sugiyama, Kasahara, Hirasawa, and Namekawa [24]. To keep this paper self-contained we give a streamlined proof here, generalizing the streamlined proof for $f = 1$ from [6].

**Theorem 2.1** *Let $q$ be a prime power. Let $m$ be a positive integer. Let $n$ be an integer with $1 \leq n \leq q^m$. Let $a_1, a_2, \ldots, a_n$ be distinct elements of $\mathbf{F}_{q^m}$. Let $f$ and $g$ be coprime monic polynomials in $\mathbf{F}_{q^m}[x]$ that both*

*do not vanish at any of* $a_1, \ldots, a_n$*. Assume that* $g$ *is squarefree. Then* $\Gamma_q(a_1, a_2, \ldots, a_n, fg^{q-1}) = \Gamma_q(a_1, a_2, \ldots, a_n, fg^q)$*.*

*Proof.* If $\sum_i c_i/(x - a_i) = 0$ in $\mathbf{F}_{q^m}[x]/(fg^q)$ then certainly $\sum_i c_i/(x - a_i) = 0$ in $\mathbf{F}_{q^m}[x]/(fg^{q-1})$.

Conversely, consider any $(c_1, c_2, \ldots, c_n) \in \mathbf{F}_q^n$ such that $\sum_i c_i/(x - a_i) = 0$ in $\mathbf{F}_{q^m}[x]/(fg^{q-1})$; i.e., $fg^{q-1}$ divides $\sum_i c_i/(x - a_i)$ in $\mathbf{F}_{q^m}[x]$. We need to show that $fg^q$ divides $\sum_i c_i/(x - a_i)$ in $\mathbf{F}_{q^m}[x]$, in particular that $g^q$ divides $\sum_i c_i/(x - a_i)$ in $\mathbf{F}_{q^m}[x]$. Find an extension $k$ of $\mathbf{F}_{q^m}$ so that $g$ splits into linear factors in $k[x]$. Then $\sum_i c_i/(x - a_i) = 0$ in $k[x]/g^{q-1}$, so $\sum_i c_i/(x - a_i) = 0$ in $k[x]/(x - r)^{q-1}$ for each factor $x - r$ of $g$. The elementary series expansion

$$\frac{1}{x - a_i} = -\frac{1}{a_i - r} - \frac{x - r}{(a_i - r)^2} - \frac{(x - r)^2}{(a_i - r)^3} - \cdots$$

then implies

$$\sum_i \frac{c_i}{a_i - r} + (x - r) \sum_i \frac{c_i}{(a_i - r)^2} + (x - r)^2 \sum_i \frac{c_i}{(a_i - r)^3} + \cdots = 0$$

in $k[x]/(x - r)^{q-1}$; i.e., $\sum_i c_i/(a_i - r) = 0$, $\sum_i c_i/(a_i - r)^2 = 0$, $\ldots$, $\sum_i c_i/(a_i - r)^{q-1} = 0$. Now take the $q$th power of the equation $\sum_i c_i/(a_i - r) = 0$, and use the fact that $c_i \in \mathbf{F}_q$, to obtain $\sum_i c_i/(a_i - r)^q = 0$. Work backwards to see that $\sum_i c_i/(x - a_i) = 0$ in $k[x]/(x - r)^q$.

By hypothesis $g$ is the product of its distinct linear factors $x - r$. Therefore $g^q$ is the product of the coprime polynomials $(x - r)^q$, and $\sum_i c_i/(x - a_i) = 0$ in $k[x]/g^q$; i.e., $\sum_i c_i/(x - a_i) = 0$ in $\mathbf{F}_{q^m}[x]/g^q$. Finally, $f$ is coprime to $g^q$, so $\sum_i c_i/(x - a_i) = 0$ in $\mathbf{F}_{q^m}[x]/(fg^q)$. □

## 3 Attacks and defenses

Generic attacks against code-based cryptosystems are those whose hardness depends only on the code parameters $q, n, k$ and the number $w$ of errors. For $q > 2$ the most efficient generic attack stated in the literature is the generalized information-set-decoding attack described in [18]. As far as we know, generic attacks are the largest threat against the wild McEliece system and the wild McEliece incognito system, when parameters are chosen sensibly.

The extra factor $f$ described in the previous section allows us to increase the number of Goppa polynomials so that an attacker cannot enumerate all polynomials $f$ and $g$ of the given degrees in less time than

performing information-set decoding. We actually suggest increasing the number of polynomials to the square of this, in case there is some square-root attack against the space of polynomials. We also retain the defense used in [6], namely choosing the support as a secret proper subset of $\mathbf{F}_{q^m}$, again with the number of possibilities being the square of the cost of information-set decoding.

One might think that the factorizability of $fg^{q-1}$ is somehow analogous to the concatenated structure attacked in [20]. However, one cannot even begin the attack of [20] without finding low-weight words in the dual code. We have checked in examples of various sizes that the dual code of $\Gamma_q(a_1, \ldots, a_n, fg^{q-1})$ does not have words of low weight, so attacks of this type do not apply. Note that any severe problem with factorizability would also break the original McEliece system, since every polynomial can be factored over a suitable extension field.

To make structural attacks even harder we suggest using an idea of Berger and Loidreau [2] which they introduced in an attempt to protect Generalized Reed-Solomon (GRS) codes, namely to add $\ell$ additional rows to the parity-check matrix. There are $\binom{k}{\ell}_q = \frac{(1-q^k)(1-q^{k-1})\cdots(1-q^{k-\ell+1})}{(1-q)(1-q^2)\cdots(1-q^\ell)}$ subspaces of dimension $\ell$ in a $k$-dimensional code over $\mathbf{F}_q$; this is a very large number even for $\ell = 1$. Wieschebrink showed in [25] that the structure of GRS can still be detected despite the extra defense, but the attack relies strongly on properties of GRS and does not seem to carry over to wild Goppa codes and their close relatives.

We emphasize that these defenses have very low cost, only slightly increasing the size of the public key compared to pure wild McEliece. The effect of [2] is that in systematic form the public key has $(n-k+\ell)(k-\ell)$ entries instead of $(n-k)k$; this is a negligible effect for small $\ell$. The small effect of $f$ on the key size is illustrated with optimized numerical examples in Section 5. There are even cases (e.g., $2^{100}$ security for $q = 31$) where the improved granularity of $fg^{q-1}$ allowed our computations to find *smaller* keys for $fg^{q-1}$ than for $g^{q-1}$ at the same security level.

## 4 Challenges

We have created a spectrum of cryptanalytic challenges as a way to measure and focus progress in attacking our proposals. Each challenge consists of a public key and a ciphertext; we challenge the readers to find a matching plaintext or even to find the secret keys. Our challenges are online at `http://pqcrypto.org/wild-challenges.html`. We intend to keep this web page up to date to show

- any solutions (plaintexts) sent to us — with credit to the first solver of each challenge, and with as much detail as the solver is willing to provide regarding how the challenge was cryptanalyzed;
- any secret keys sent to us — again with credit to the first solver of each challenge;
- cryptanalytic benchmarks — measurements of the speed of publicly available cryptanalytic software for the smaller challenges, as a way for the community to verify and demonstrate improvements in attack algorithms;
- predictions — estimates of how difficult the larger challenges will be to break.

Our challenges, like the RSA Factoring Challenge (see [19] and [26]) and the Certicom ECC Challenges (see [8]), cover a wide range of levels of difficulty. The smallest challenges require only a small amount of computer time; the larger challenges increase rapidly in difficulty. However, we did not imitate (e.g.) the 1000× increase in difficulty between the ECC2K-108 and ECC2K-130 challenges in [8]. That increase has kept the list of solved ECC2K challenges static since 2000, not reflecting the impact of more than a decade of advances in computer technology; we prefer smaller gaps between challenges.

Each of our challenges is labelled by (1) "wild McEliece" for [6], or "wild Mceliece incognito" for this paper; (2) a field size $q$; (3) a key size expressed in kilobytes. Each challenge also has public parameters $m, n, s, t$ chosen as discussed below. After choosing these parameters we built the challenge as follows:

- Choose a secret sequence of $n$ distinct elements $a_1, \ldots, a_n$ of $\mathbf{F}_{q^m}$.
- Choose a secret irreducible polynomial $g$ of degree $t$ in $\mathbf{F}_{q^m}[x]$. If $g$ has any of $a_1, \ldots, a_n$ as roots, repeat this step. (This can occur only for $t = 1$.)
- Choose a secret irreducible polynomial $f$ of degree $s$ in $\mathbf{F}_{q^m}[x]$. If $f$ has any of the $a_1, \ldots, a_n$ as roots, repeat this step. (In principle we should, but we did not, also check for the rare possibility that $s = t$ and $f = g$.)
- Write down an $(n-k) \times n$ parity-check matrix $H$ for the Goppa code $\Gamma_q(a_1, \cdots, a_n, fg^{q-1})$, where $k = n - m(s + (q-1)t)$.
- Row-reduce $H$ so that it begins with an $(n-k) \times (n-k)$ identity matrix and continues with an $(n-k) \times k$ public key. If this fails (i.e., the first $n-k$ columns of $H$ are not invertible), go back to the first step.

- Choose a secret plaintext. Here we use the Niederreiter variant [16]: a plaintext is a random element of $\mathbf{F}_q^n$ of Hamming weight $w$, where $w = \lfloor (s + (q-1)t)/2 \rfloor$. (This can be made CCA2-secure with negligible loss of efficiency, by techniques analogous to the techniques of [14].) For simplicity we do not use list decoding here. We also do not use the Berger–Loidreau defense.
- Multiply the secret plaintext by the row-reduced $H$, obtaining a public ciphertext in $\mathbf{F}_q^{n-k}$.
- As a verification step, use the secret key to legitimately decrypt the ciphertext, and then check that the result matches the original plaintext.
- Throw away all the secret information, leaving only the ciphertext and the public key.

We wrote a script in the Sage computer-algebra system [23] to do all this, relying on Sage's random-number generator to produce all secrets; the Sage documentation indicates that the random-number generator is cryptographic. This script appears on our web page. The script was designed mainly as a reference implementation, easy to understand and easy to verify; it was not designed for speed. However, we did incorporate a few standard speedups (such as a balanced product tree inside interpolation in generalized Reed–Solomon decryption) to reduce the time spent generating challenges.

We formatted each challenge as a text file containing cryptosystem parameters, a ciphertext, and a public key. For example, here is our 20kB "wild McEliece incognito" challenge for $q = 13$, except that in the actual file there are various additional numbers in place of the dots:

```
kilobytes = 19.9869819590563
q = 13
m = 3
n = 472
s = 7
t = 3
u = 43
k = 343
w = 23
ciphertext = [7, 4, 12, 7, 7, ..., 2, 8, 10, 5, 0]
recovered_plaintext_using_secret_key = True
pubkeycol129 = [9, 11, 0, 4, 9, ..., 4, 12, 8, 1, 3]
pubkeycol130 = [5, 4, 12, 7, 2, ..., 6, 12, 5, 11, 12]
```

```
...
pubkeycol471 = [0, 1, 11, 3, 6, ..., 11, 12, 4, 11, 3]
```

In this example there are approximately $2^{1644}$ possible sets $\{a_1, \ldots, a_{472}\}$, and approximately $2^{107}$ possible pairs $(f, g)$. This challenge has wildness percentage 84% because $\deg(g^{q-1}) = 36$ accounts for 84% of $u = \deg(fg^{q-1}) = 43$. The ciphertext is a column vector containing $n-k = 129$ elements of $\mathbf{F}_{13}$. This column vector is a sum of nonzero coefficients times $w = 23$ columns chosen secretly from the 472 columns of the row-reduced $H$; these 472 columns consist of $k = 343$ public-key columns shown in the challenge file, and 129 extra columns containing an identity matrix.

The public key in this challenge has $343 \cdot 129 \cdot \log(13)/\log 2 \approx 163733$ bits of information, slightly below the advertised "20kB" (163840 bits). A standard radix-13 integer encoding of the matrix would fit into 163734 bits but would take some work to uncompress. Packing each 129-entry column separately into 478 bits would consume 163954 bits. A standard 4-bit encoding of $\mathbf{F}_{13}$ would consume only slightly more space, 21.6kB.

The generalized information-set-decoding attack introduced by Peters in [18] will break this challenge in roughly $2^{53}$ bit operations. This is obviously feasible.

As another example, our 40kB "wild McEliece" challenge for $q = 31$ has $m = 2$, $n = 666$, $s = 0$, $t = 2$, $k = 546$, and $w = 31$. In this case security relies critically on the defense suggested in [6]: there are only about $2^{19}$ possible polynomials $g$, but there are almost $2^{850}$ possible support sets. Information-set decoding will break this challenge in roughly $2^{89}$ bit operations.

As a final example, our 20kB "wild McEliece" challenge for $q = 3$ has $m = 6$, $n = 729$, $s = 0$, $t = 16$, $k = 537$, and $w = 24$. In this case there is only 1 possible set $\{a_1, \ldots, a_{729}\}$, namely all of $\mathbf{F}_{3^6}$, but there are approximately $2^{148}$ possible polynomials $g$. Information-set decoding will break this challenge in roughly $2^{54}$ bit operations. Does knowing the support help any attack?

We considered a huge number of possible parameters $m, n, s, t$ for each challenge, and many parameters for the attack in [18], subject to the key-size constraint $k(n-k)\log_2 q \leq 8192K$, where $K$ is the specified number of kilobytes in the key. We assigned a security level $2^b$ to $(m, n, s, t)$ according to an approximation to the cost of the attack in [18]. For the "wild McEliece incognito" challenges we rejected $(m, n, s, t)$ whenever the number of polynomials was below $2^{2b}$, and we also rejected $(m, n, s, t)$ whenever the number of support sets was below $2^{2b}$. For the "wild McEliece" challenges we did not require these defenses separately:

we allowed the product of the number of polynomials and the number of support sets to be as small as $2^{2b}$. Subject to these constraints we selected $(m, n, s, t)$ for each challenge to maximize $b$. This procedure matches how we would expect parameters to be chosen in practice.

## 5 Parameters

In this section we propose parameters $(n, k, s, t)$ for the McEliece cryptosystem using codes $\Gamma = \Gamma_q(a_1, \ldots, a_n, fg^{q-1})$ that provide $2^{128}$ security against information-set decoding and that have more than $2^{256}$ choices of $fg^{q-1}$. Our parameter search uses the analysis of information-set decoding in [18]. We chose the code length $n$, the degree $s$ of $f$, the degree $t$ of $g$ and the dimension $k = n - \lceil \log_q n \rceil (s + (q-1)t)$ of $\Gamma$ to minimize the key size $\lceil (n-k)k \log_2 q \rceil$ for 128-bit security when $w$ errors are added. Table 5.1 gives an overview. The last column of the table shows the "wildness percentage" $p$, i.e., the contribution of $g^{q-1}$ to the Goppa polynomial, measured in terms of how its degree relates to the overall degree.

| $q$ | key size | $n$ | $k$ | $s$ | $t$ | $w$ | $p$ |
|---|---|---|---|---|---|---|---|
| 3 | 186 kB | 2136 | 1492 | 0 | 46 | 69 | 100% |
| 4 | 210 kB | 2252 | 1766 | 0 | 27 | 54 | 100% |
| 5 | 191 kB | 1878 | 1398 | 0 | 24 | 60 | 100% |
| 7 | 170 kB | 1602 | 1186 | 8 | 16 | 60 | 92% |
| 8 | 187 kB | 1628 | 1204 | 8 | 14 | 60 | 92% |
| 9 | 205 kB | 1668 | 1244 | 10 | 12 | 59 | 91% |
| 11 | 129 kB | 1272 | 951 | 17 | 9 | 58 | 84% |
| 13 | 142 kB | 1336 | 1033 | 17 | 7 | 54 | 83% |
| 16 | 157 kB | 1328 | 1010 | 16 | 6 | 56 | 85% |
| 17 | 162 kB | 1404 | 1113 | 17 | 5 | 51 | 82% |
| 19 | 169 kB | 1336 | 1015 | 17 | 5 | 56 | 84% |
| 23 | 183 kB | 1370 | 1058 | 16 | 4 | 54 | 85% |
| 25 | 189 kB | 1314 | 972 | 18 | 4 | 59 | 84% |
| 27 | 200 kB | 1500 | 1218 | 42 | 2 | 48 | 55% |
| 29 | 199 kB | 1390 | 1081 | 19 | 3 | 53 | 82% |
| 31 | 88 kB | 856 | 626 | 25 | 3 | 59 | 78% |
| 32 | 89 kB | 852 | 618 | 24 | 3 | 60 | 79% |

Table 5.1: Optimized parameters $(n, k, s, t)$ for wild Goppa codes over $\mathbf{F}_q$ achieving 128-bit security when introducing $w = \lfloor (s + qt)/2 \rfloor$ errors.

Figure 5.1 illustrates for $q = 13$ that, given a particular key size, higher wildness percentages generally add extra security against information-set decoding. The figure compares Goppa codes with no correction factor (100% wild) to Goppa codes where the degrees of $f$ and $g^{q-1}$ are balanced (50% wild), and to Goppa codes without the wild trick (0% wild). We emphasize that adding our shield against polynomial-searching attacks does not require dropping the wildness percentage from 100% all the way down to 50%; the parameters suggested in Table 5.1 typically have very small extra factors $f$, profiting from the higher error-correction capability induced by $g^{q-1}$.

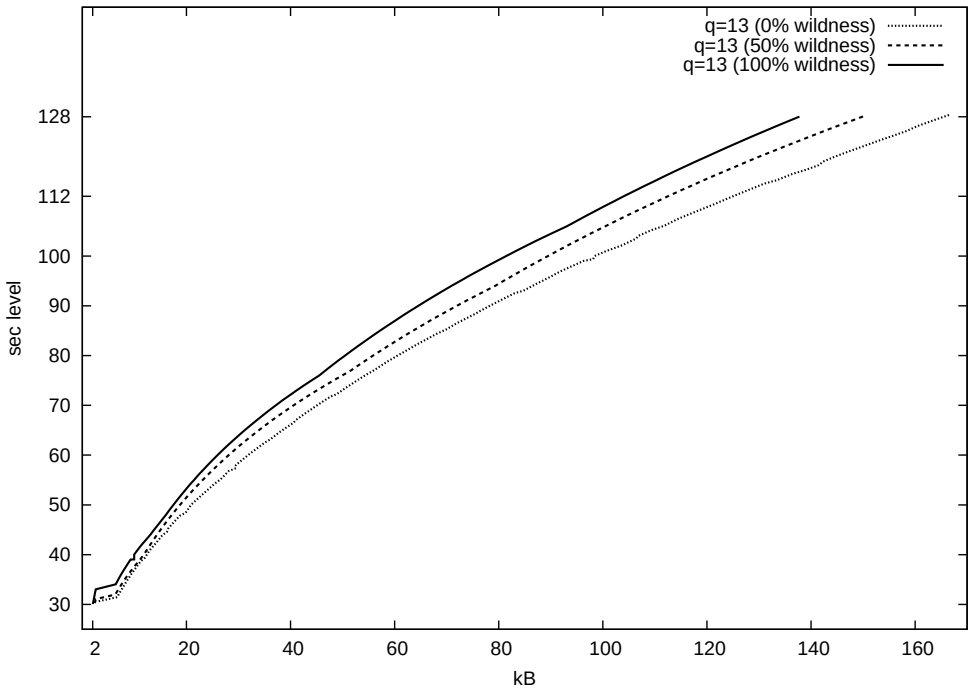


Fig. 5.1: Security levels attained for wild McEliece keys with different wildness percentages for $q = 13$.

## References


[1] Daniel Augot, Morgan Barbier, Alain Couvreur, *List-decoding of binary Goppa codes up to the binary Johnson bound* (2010). URL: http://arxiv.org/abs/1012.3439. Citations in this document: §2.

[2] Thierry P. Berger, Pierre Loidreau, *How to mask the structure of codes for a cryptographic use*, Designs, Codes and Cryptography **35** (2005), 63–79. MR 2006d:94038. URL: http://www.springerlink.com/index/JR001118R1567U13.pdf. Citations in this document: §1, §3, §3.

[3] Elwyn R. Berlekamp, *Algebraic coding theory*, Aegean Park Press, 1984. ISBN 0894120638. Citations in this document: §2.

[4] Daniel J. Bernstein, *List decoding for binary Goppa codes*, in IWCC [10] (2011), 62–80. URL: http://cr.yp.to/papers.html#goppalist. Citations in this document: §2.

[5] Daniel J. Bernstein, *Simplified high-speed high-distance list decoding for alternant codes*, in PQCrypto [27] (2011). URL: http://cr.yp.to/papers.html#simplelist. Citations in this document: §2, §2.

[6] Daniel J. Bernstein, Tanja Lange, Christiane Peters, *Wild McEliece*, in SAC 2010 [7] (2011), 143–158. URL: http://eprint.iacr.org/2010/410. Citations in this document: §1, §1, §1, §1, §1, §1, §2, §2, §2, §3, §4, §4.

[7] Alex Biryukov, Guang Gong, Douglas R. Stinson (editors), *Selected areas in cryptography — 17th international workshop, SAC 2010, Waterloo, Ontario, Canada, August 12–13, 2010, revised selected papers*, Lecture Notes in Computer Science, 6544, Springer, 2011. See [6].

[8] Certicom, *Certicom ECC Challenge* (1997). URL: http://www.certicom.com/images/pdfs/cert_ecc_challenge.pdf. Citations in this document: §4, §4.

[9] Pascale Charpin (editor), *Livre des résumés — EUROCODE 94, Abbaye de la Bussière sur Ouche, France, October 1994*, 1994. See [20].

[10] Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, Chaoping Xing (editors), *Coding and cryptology — third international workshop, IWCC 2011, Qingdao, China, May 30–June 3, 2011, proceedings*, Lecture Notes in Computer Science, 6639, Springer. See [4].

[11] Valery D. Goppa, *A new class of linear error correcting codes*, Problemy Peredachi Informatsii **6** (1970), 24–30. Citations in this document: §2, §2.

[12] Valery D. Goppa, *Rational representation of codes and* $(L, g)$*-codes*, Problemy Peredachi Informatsii **7** (1971), 41–49. Citations in this document: §2.

[13] Kwangjo Kim (editor), *Public key cryptography: proceedings of the 4th international workshop on practice and theory in public key cryptosystems (PKC 2001) held on Cheju Island, February 13–15, 2001*, Lecture Notes in Computer Science, 1992, Springer, 2001. See [14].

[14] Kazukuni Kobara, Hideki Imai, *Semantically secure McEliece public-key cryptosystems — conversions for McEliece PKC*, in PKC 2001 [13] (2001), 19–35. MR 2003c:94027. Citations in this document: §4.

[15] Robert J. McEliece, *A public-key cryptosystem based on algebraic coding theory*, JPL DSN Progress Report (1978), 114–116. URL: http://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF. Citations in this document: §1.

[16] Harald Niederreiter, *Knapsack-type cryptosystems and algebraic coding theory*, Problems of Control and Information Theory **15** (1986), 159–166. Citations in this document: §1, §4.

[17] Nicholas J. Patterson, *The algebraic decoding of Goppa codes*, IEEE Transactions on Information Theory **21** (1975), 203–207. Citations in this document: §2.

[18] Christiane Peters, *Information-set decoding for linear codes over* $\mathbf{F}_q$, in PQCrypto 2010 [22] (2010), 81–94. URL: http://eprint.iacr.org/2009/589. Citations in this document: §1, §3, §4, §4, §4, §5.

[19] RSA Laboratories, *The RSA Factoring Challenge* (1991). URL: http://www.rsa.com/rsalabs/node.asp?id=2092. Citations in this document: §4.

[20] Nicolas Sendrier, *On the structure of a randomly permuted concatenated code*, in EUROCODE 94 [9] (1994), 169-173. Citations in this document: §3, §3.

[21] Nicolas Sendrier, *Finding the permutation between equivalent linear codes: the support splitting algorithm*, IEEE Transactions on Information Theory **46** (2000), 1193–1203. MR 2001e:94017. URL: http://hal.inria.fr/docs/00/07/30/37/PDF/RR-3637.pdf. Citations in this document: §1.

[22] Nicolas Sendrier (editor), *Post-quantum cryptography, third international workshop, PQCrypto, Darmstadt, Germany, May 25-28, 2010*, Lecture Notes in Computer Science, 6061, Springer, 2010. See [18], [25].

[23] William Stein (editor), *Sage Mathematics Software (Version 4.4.3)*, The Sage Group, 2010. URL: http://www.sagemath.org. Citations in this document: §4.

[24] Yasuo Sugiyama, Masao Kasahara, Shigeichi Hirasawa, Toshihiko Namekawa, *Further results on Goppa codes and their applications to constructing efficient binary codes*, IEEE Transactions on Information Theory **22** (1976), 518–526. Citations in this document: §2.

[25] Christian Wieschebrink, *Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes*, in PQCrypto 2010 [22] (2010), 61–72. Citations in this document: §3.

[26] Wikipedia, *RSA Factoring Challenge — Wikipedia, The Free Encyclopedia*, accessed 01 July 2011 (2011). URL: http://en.wikipedia.org/wiki/RSA_Factoring_Challenge. Citations in this document: §4.

[27] Bo-Yin Yang (editor), *Post-quantum cryptography, fourth international workhop, PQCrypto, Taipei, Taiwan, November 29–December 02, 2011*, Lecture Notes in Computer Science, 2011. See [5].