Code-based Cryptography

1. The parity-check matrix is $H = (1\,1 \dots 1\ -1)$. The subspace of codewords $y \in GF(q)^n$ is defined by $Hy^T = 0$ and has dimension $n-1$. The vectors of weight 1, i.e., the unit vectors e_i, do not satisfy $He_i^T = 0$. On the other hand, there are codewords of weight 2, e.g., $c = (1,\,0,\,\dots,\,0,\,1)$. This shows that C is a $[n, n-1, 2]$ code. Since $d = n-k+1 = n-(n-1)+1 = 2$, the code is MDS.
2. A q-ary lattice is defined by the rows of a matrix A over $\mathbb{Z}_q$, and hence by a subspace of $\mathbb{Z}_q^n$. A linear code is also given by a subspace of $\mathbb{Z}_q^n$. However, lattice problems are based on the Euclidean norm, whereas decoding uses the Hamming distance.
3. a) The rows are linearly independent, and so the dimension of the code is 4. One checks that $GG^T = 0$ and hence $(xG)G^T = 0$. It follows that all codewords $c \in C$ satisfy $c\,G^T = 0$. Since C and the subspace defined by $yG^T = 0$ both have dimension 4, we obtain $C = \{y \in GF(2)^8 \mid yG^T = 0\}$. Therefore, G is also the parity-check matrix of the code and self-dual.
   b) If a codeword of weight < 4 existed, then a subset of three columns of the parity-check matrix G would be linearly dependent. However, this is not the case.
   c) The syndrome is $yG^T = (1,0,0,0)$. This is the first column of the parity-check matrix G. Hence the coset leader is the error vector $e = (1,0,0,0,0,0,0,0)$ and the codeword is $c = y + e = (1,1,0,0,1,1,0,0)$. The information word is $x = (1,0,1,0)$ and $xG = c$.
4. Let $n = 16$ and $d = 5$.
   Sphere-covering bound: $A_q(16,5) \geq \frac{q^{16}}{V_q(16,4)}$. For $q = 2$, we get
   $$A_2(16,5) \geq \frac{2^{16}}{\binom{16}{0}+\binom{16}{1}+\binom{16}{2}+\binom{16}{3}+\binom{16}{4}} \approx 26.04.$$
   Hence a code with at least 27 codewords exists.
   Gilbert-Varshamov bound: if $V_q(15,3) < q^{16-k}$ then a linear $[n,k]$ code with minimum distance $\geq d$ exists. For $q = 2$, we have
   $$V_q(15,3) = \binom{15}{0} + \binom{15}{1} + \binom{15}{2} + \binom{15}{3} = 576 < 1024 = 2^{10} = 2^{16-6}.$$
   Hence a linear code of dimension 6 exists., i.e., with 64 codewords.
   Hamming bound: $A_q(16,5) \leq \frac{q^{16}}{V_q(16,2)}$. For $q = 2$, we get
   $$A_2(16,5) \leq \frac{2^{16}}{\binom{16}{0}+\binom{16}{1}+\binom{16}{2}} \approx 478.$$
   So the number of codewords is at most 478. Thus the Hamming bound shows that the largest linear code with $n = 16$ and $d = 5$ has $2^8 = 256$ codewords. Therefore, the $[16,8,5]$ Goppa code in Example 15.31 has maximal dimension.
5. Since $2ab = 0$ in $GF(2^m)$, one has $(a+b)^2 = a^2 + b^2$. If the exponent is a power of 2, then:

$$(a+b)^{2^k} = (((a+b)^2)^2 \dots)^2 = ((a^2+b^2)^2 \dots)^2 = \dots = ((a^4+b^4)\dots)^2 = a^{2^k} + b^{2^k}.$$

6. $GF(2^m)$ is the splitting field of $x^{2^m} - x$. Hence all elements $a \in GF(2^m)$ satisfy $a^{2^m} = a$, and $a^{2^{m-1}}$ is a square root of a. Since $a = -a$, the square root is unique. The preceding exercise shows that
$$\sqrt{a+b} = (a+b)^{2^{m-1}} = a^{2^{m-1}} + b^{2^{m-1}} = \sqrt{a} + \sqrt{b}.$$
7. Write the polynomial f as $f_0 + f_1$, where f_0 contains the even powers of x and f_1 the odd powers. Then $f = f_0 + xf_2$, where f_0 and f_2 only contain even powers of x. From the preceding exercise we know that the coefficients of f_0 and f_2, and hence the polynomials can be written as squares, i.e.,
$$f_0 = \alpha^2 \text{ and } f_2 = \beta^2$$
with α, $\beta \in GF(2^m)[x]$.
8. If g is irreducible, then $GF(2^m)[x]/(g(x))$ is a binary field of order $2^{m \deg(g)}$. We have seen in Exercise 6 that a unique square root exists in binary fields. The first formula can be shown by squaring both sides of the equation. Note that $x \bmod g(x)$ is an element of the above binary field, and so a unique element $\sqrt{x}$ exists. Squaring Huber's first formula gives the equivalent equation
$$x = g_1^2 g_2^{-2} \bmod g(x).$$
The assumption $g = g_1^2 + xg_2^2$ implies $g_1^2 = xg_2^2 \bmod g(x)$ and hence the assertion. Squaring Huber's second formula yields
$$x = v_2^2 g_1^2 + x^2 v_1^2 g_2^2 \bmod g(x)$$
Again, we use $g_1^2 = xg_2^2 \bmod g(x)$ and obtain the equivalent equation
$$x = xv_2^2 g_2^2 + xv_1^2 g_1^2 = x(v_2^2 g_2^2 + v_1^2 g_1^2) \bmod g(x)$$
This equation is true since $1 = v_1 g_1 + v_2 g_2$ implies $v_2^2 g_2^2 + v_1^2 g_1^2 = 1$.
9. We construct the field $GF(16)[x]/(g(x))$ and define the array `arr` of elements $\frac{1}{x-a} \bmod g(x)$, where $a \in GF(16)$.

```
sage: K.<z>=GF(2^4, name='z', modulus=x^4+x+1) # K=GF(16)
sage: PR.<x>=K[] # polynomial ring K[x]
sage: g=x^2+z^2*x+z;
sage: Rmodg=PR.quotient_ring(g) # GF(16)/(g(x))
sage: arr=[]  # define values 1/(x-a) mod g(x) for a in K
sage: for a in K.list():
          arr.append(1/Rmodg(x-a))
```

We use Patterson's decoding algorithm:

```
sage: def patterson(w):
            e=vector(GF(2),16)
            Syn=w*vector(arr) # syndrome Syn(w)
            if Syn==0:
                  return(e,w) # no error
            T=1/(w*vector(arr))
            if (T==Rmodg(x)):
                  sigma=x
            else:
                  R=(T-Rmodg(x))^128;  # sqrt(T-x)
                  a0=R.lift() # lift R to K[x]
                  b0=1
                  sigma=a0*a0+x*b0*b0
            i=0
            for k in list(K):
                  if ((sigma.subs(x=k))==0): # error positions
                        e[i]=1
                  i=i+1
            return(e,w+e) # error vector and codeword
```

Patterson's decoding algorithm is applied to $(1,1,1,1,1,1,1,0,0,1,1,0,1,0,0,0)$.

```
sage: y=vector(GF(2),[1,1,1,1,1,1,1,0,0,1,1,0,1,0,0,0])
sage: e, c = patterson(y)
sage: e
(0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0)
sage: c
(1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0)
```

We obtain the codeword $c = (1,1,1,1,1,1,1,0,0,0,1,0,0,0,0,0)$.

10. We decode $y_1 = (0,1,0,0,0,1,0,1,1,1,0,0,0,1,1,1)$.

```
sage: y1=vector(GF(2),[0,1,0,0,0,1,0,1,1,1,0,0,0,1,1,1])
sage: e, c = patterson(y1);  c
(0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1)
```

Hence the codeword is $c = (0,1,0,0,0,1,0,1,1,0,0,0,1,1,1,1)$.

11. If the same plaintext x is encrypted twice using the generator matrix G_1, then $y_1 = xG_1 + e_1$ and $y_2 = xG_1 + e_2$. Adding the ciphertexts gives $y_1 + y_2 = e_1 + e_2$. Hence an adversary obtains the combined error vector and learns possible error positions. This problem no longer exists with Pointcheval's generic conversion, since a random string r is encrypted instead of a plaintext x. Encrypting the same plaintext twice gives uncorrelated ciphertexts.

12. We compute the parity-check matrix H of the Goppa code from a given array `arr` of elements $\frac{1}{x-a} \bmod g(x)$, where $a \in GF(16)$.

```
sage: H16=matrix(K,2,16)
            for i in range(0,2):
                  for j in range(0,16):
                        H16[i,j]=list(arr[j])[i]
```

```
sage: H=matrix(GF(2),8,16)
sage: for i in range(0,2):
            for j in range(0,16):
                  H16[i,j]=list(arr[j])[i]
                  hbin=bin(eval(H16[i,j]._int_repr()))[2:]
                  hbin='0'*(4-len(hbin))+hbin; hbin = list(hbin);
                  H[4*i:4*(i+1),j] = vector(map(GF(2),hbin));
sage: H
[0 1 0 1 1 0 0 1 0 1 1 1 1 0 0 1]
[0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 1]
[1 1 0 0 0 1 0 1 1 1 1 1 1 1 0 0]
[0 0 0 0 0 0 1 1 0 0 1 1 1 0 1 1]
[1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0]
[0 0 0 1 1 0 1 0 1 1 1 1 0 1 1 1]
[0 1 0 1 0 1 1 0 1 0 0 0 0 0 0 1]
[1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0]
```

The given syndrome is $syn = (0,0,1,1,1,1,0,1)$. We compute a vector z with $zH^T = syn$.

```
sage: syn = vector(GF(2),[0, 0, 1, 1, 1, 1, 0, 1])
sage: z=H.solve_right(syn)  # a solution of Hz^T = syn
sage: z
(1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)
```

We obtain for example $z = (1,0,1,1,0,1,0,1,0,0,0,0,0,0,0,0)$. Now we decode z using Patterson's algorithm.

```
sage: e,c = patterson(z)
sage: print(e); print(c)
(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)
(0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0)
```

The codeword is $c = (0,0,1,1,0,1,0,1,0,0,0,0,0,0,1,0)$.