KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Computer Algebra for Cryptography

Project I

FINAL REPORT

Robin Martens

ACADEMIC YEAR 2024-2025

Task 1

Task 1.b

We first note that $|\mathbb{F}_N^n| = N^n$ and that every linear disequation $a \cdot s \neq b$ deletes $\frac{1}{N}$ from the solution space, so there remain $\frac{N-1}{N} \cdot N^n$ solutions after 1 disequation. After m disequations, we have $N^n \left(\frac{N-1}{N}\right)^m$ remaining solutions in the solution space, so we want that amount to be equal to 1. Solving that equation for m yields:

$$m = \left(\frac{\ln N}{-\ln(1-\frac{1}{N})}\right) n, \tag{1}$$

such that $C_N = -\frac{\ln N}{\ln(1-\frac{1}{N})}$.

Task 1.c

As N is a prime, we know that $\varphi(N) = N - 1$ and from above we have that the number of solutions after m disequations is given by

$$N^{n-m}(N-1)^m = N^{n-m}\varphi(N)^{m-1}, \tag{2}$$

such that the statement in this question is true.

Task 2

Task 2.a

In $\mathbb{F}_2$, there are only two possible coefficients for $\mathbf{a}$, $\mathbf{s}$ and b, so simply inverting b will transform the disequation into an equation. For example, if we have an $\mathbf{s}$ which complies with

$$a_1 s_1 + \cdots + a_n s_n \neq b \tag{3}$$

so the LHS is either 0/1 and the RHS is 1/0. then the same $\mathbf{s}$ will comply with

$$a_1 s_1 + \cdots + a_n s_n = \overline{b} \tag{4}$$

as now the LHS and RHS are equal (we can also invert the LHS, so $\mathbf{a}$).

Task 2.b

We first repeat Fermat's little theorem: for an $a \in \mathbb{F}_p$ with $\gcd(a, p) = 1$:

$$a^{p-1} = 1 \tag{5}$$

and thus, as p is a prime, $\gcd(a, p) = 1$ if and only if $a \neq 0$. With Fermat's little theorem, we can transform $a \neq 0$ into $a^{p-1} = 1$. Starting from the equation

$$\begin{aligned} a_1x_1 + \cdots + a_nx_n \neq b &\Leftrightarrow a_1x_1 + \cdots + a_nx_n - b \neq 0 \\ &\Leftrightarrow (a_1x_1 + \cdots + a_nx_n - b)^{p-1} = 1 \end{aligned} \tag{6}$$

and thus function f is $f(x_1, \ldots, x_n) = (a_1x_1 + \ldots x_na_n - b)^{p-1} - 1$.

Task 2.c

The table below summarizes the results for different values of r and p.

Table 1: Timing results for $n = 8$.

r	p	Time taken [s]	# eqs.
1	2	0.000	8
1	3	0.010	22
1	5	0.540	58
1	7	218.580	101
2	2	0.000	16
2	3	0.010	44
2	5	0.120	116
2	7	39.860	202
3	2	0.000	24
3	3	0.010	66
3	5	0.090	174
3	7	21.930	303

We can see that adding the field equations does not have an influence on the runtime of the algorithm, which is logical, given the fact that the field equations actually just state Fermat's little theorem and as p is prime, the field equation will hold for any member of the field $\mathbb{F}_p$.

Table 2: Timing results with added field equations and $n = 8$.

p	Time taken [s]	# eqs.
2	0.000	16
3	0.000	30
5	0.570	66
7	217.220	109

Task 2.d

Euler's congruence states that for an $a \in \mathbb{F}_N$ and $\gcd(a, N) = 1$, $a^{\phi(N)} = 1$. So now, we cannot transform the disequation $a_1x_1 + \cdots + a_nx_n \neq 0$ simply into an equation, as $a_1x_1 + \cdots + a_nx_n$ can be equal to a multiple of p, such that Euler's congruence does not work anymore.

Task 3

Task 3.a

Fermat's little theorem states that for an $a \in \mathbb{F}_p$, $a^{p-1} = 1$ if $a \neq 0$ and $a^{p-1} = 0$ if $a = 0$. Then we define, $f(x) = 1 - (x - c)^{p-1}$, such that $f(x) = 1$ if $x = c$ and $f(x) = 0$ if $x \neq c$. Extending this to c and d yields:

$$f(x, y) = \left(1 - (x - c)^{p-1}\right)\left(1 - (y - d)^{p-1}\right), \tag{7}$$

such that $f(x,y)=1$ if $(x,y)=(c,d)$ and $f(x,y)=0$ if $(x,y)\neq(c,d)$. Now, as stated in the assignment, $W(x,y)$ is now the sum of the above polynomials when $c+d\geq p$:

$$W(x,y)=\sum_{\substack{c,d\in\mathbb{F}_p\\ c+d\geq p}}\left(1-(x-c)^{p-1}\right)\left(1-(y-d)^{p-1}\right). \tag{8}$$

Task 3.b

We have $c=c_0+c_1p$ and $d=d_0+d_1p$, such that $c+d=(c_0+c_1p)+(d_0+d_1p)=(c_0+d_0)+(c_1+d_1)p$. Now, if $c_0+d_0\geq p$, we have to add 1 to the coefficient of p, which is $W(c_0,d_0)$. If $c_1+d_1\geq p$, we have to reduce modulo p, as the result $c+d$ will also be reduced modulo p^2.

Task 3.c

We first find the functions f_1 and f_2 from the hint. For that, we write all a_i, x_i and b as: $a_i=a_{i0}+pa_{i1}$, $x_i=x_{i0}+px_{i1}$ and $b=b_0+pb_1$. We fill this into the expression $a_1x_1+\dots+a_nx_n-b$ and after multiplying out and reducing mod p^2:

$$\begin{cases} f_0(x_{11},\dots,x_{n2})=a_{10}x_{10}+\dots+a_{n0}x_{n0}-b_0\\ f_1(x_{11},\dots,x_{n2})=a_{11}x_{10}+a_{10}x_{11}+\dots+a_{n_1}x_{n0}+a_{n0}x_{n1}-b_1\end{cases}. \tag{9}$$

Now, it is still possible that $f_0(s)\geq p$, in which case we would need to add something to f_1. We can use the W-function for this, where we have to decompose f_0 to sums of elements in $\mathbb{F}_p$. First, we note that

$$\begin{cases} 2x_{10}=\overline{2x_{10}}+W(x_{10},x_{10})\\ 3x_{10}=2x_{10}+x_{10}=\overline{3x_{10}}+W(x_{10},x_{10})+W(2x_{10},x_{10})\end{cases}, \tag{10}$$

where the $\bar{\cdot}$ indicates that the result is $<p$. If we generalize this, we find:

$$a_{10}x_{10}=\overline{a_{10}x_{10}}+p\sum_{i=1}^{a_{10}-1}W\left(ix_{10},x_{10}\right). \tag{11}$$

Applying this for all terms in f_0 we find:

$$f_1(x_{11},\dots,x_{n2})=\sum_{i=1}^{n}(\overline{a_{i1}x_{i0}}+\overline{a_{i0}x_{i1}})-b_1+\sum_{j=1}^{n}\sum_{k=1}^{a_{j0}-1}W(kx_{j0},x_{j0}). \tag{12}$$

This way, all the multiplications are smaller than p, the sum of these multiplications can still be $\geq p$. As an example, if we have $\overline{a}+\overline{b}+\overline{c}$, where $\overline{a},\overline{b},\overline{c}\in\mathbb{F}_p$, then

$$\begin{aligned}\overline{a}+\overline{b}+\overline{c}&=\overline{a+b}+\overline{c}+W(\overline{a},\overline{b})\\ &=\overline{a+b+c}+W(\overline{a},\overline{b})+W(\overline{a+b},\overline{c}).\end{aligned} \tag{13}$$

Adding this to f_1 yields:

$$f_1(x_{11},\dots,x_{n2})=\sum_{i=1}^{n}(\overline{a_{i1}x_{i0}}+\overline{a_{i0}x_{i1}})-b_1+\sum_{j=1}^{n}\sum_{k=1}^{a_{j0}-1}W(kx_{j0},x_{j0})+\sum_{j=2}^{n}W\left(\sum_{k=1}^{j-1}a_{k0}x_{k0},a_{j0}x_{j0}\right). \tag{14}$$

Now, we have f_0 and f_1 and we know the following:

$$a_1x_1+\dots+a_nx_n\neq b\Leftrightarrow f_0(x_{11},\dots,x_{n2})+pf_1(x_{11},\dots,x_{n2})\neq 0. \tag{15}$$

Then we use the same trick as in Task 2, namely applying Fermat's Little Theorem:

$$\begin{cases} g_0 = (f_0 - b_0)^{p-1} \\ g_1 = (f_1 - b_1)^{p-1} \end{cases} \tag{16}$$

both of which are either 0 or 1, but they cannot be 0 at the same, otherwise the condition above would no longer hold. It is however possible that in the special case of $p = 2$, both $g_0 = 1$ and $g_1 = 1$, such that their sum is $0 mod 2$. For that case, we again add a W function. The final function then becomes:

$$f = \left((f_0 - b_0)^{p-1} + (f_1 - b_1)^{p-1} + W(f_1, f_2)\right)^{p-1} - 1. \tag{17}$$

Due to the fact that the sum within the outer parenthesis is non-zero, Fermat's Little Theorem states that raising it to the power $p - 1$ makes it 1. Subtracting by one then leads to the desired property $f(s) = 0$ iff the disequality holds.

Task 3.c

Table 3 shows the timing results for $p^2 = 4$ and $p^2 = 9$ for r ranging from 1 to 3. We see an opposite pattern w.r.t. Task 2, as here, the larger r, the longer it takes to solve the system, which can be explained by the fact that increasing r leads to a very large system of equations, which takes longer to solve.

Table 3: Timing results for $n = 8$.

r	p^2	Time taken [s]	# eqs.
1	4	0.010	78
1	9	0.310	299
2	4	0.010	155
2	9	0.510	597
3	4	0.010	232
3	9	0.800	896

Task 3.e

I would think solving the disequations over a field would be easier, as in a field a multiplicative inverse is always defined, which is not the case in a ring, such that operations on elements of the field become easier.

Task 4

Table 4 shows the results for the degree of regularity for increasing values of n and $m = C_N \cdot n$. We see that if n increases, d_{reg} will also increase, so if we extrapolate, d_{reg} would tend to ∞. This indicates that the larger n, the more difficult it is to solve the problem, which can also be seen based on m, the number of equations. The complexity of the system is then given by the theory as

$$O\left(\left(m\binom{n + d_{\text{reg}} - 1}{d_{\text{reg}}}\right)^{\omega}\right). \tag{18}$$

For the exhaustive search, there would be p^n amount of different secrets available. The comparison in complexity can be seen in the table.

Table 4: Degree of regularity for n and m.

n	m	d_{reg}	Complexity	3^n
10	27	3	$\mathcal{O}(2.10e11)$	5.9e4
110	298	11	$\mathcal{O}(4.14e52)$	3.04e52
210	569	18	$\mathcal{O}(1.48e87)$	1.57e100
310	840	24	$\mathcal{O}(8.03e117)$	8.08e147
410	1111	30	$\mathcal{O}(2.33e148)$	4.17e195
510	1382	37	$\mathcal{O}(1.61e182)$	2.15e243
610	1653	43	$\mathcal{O}(3.19e212)$	1.11e291
710	1924	49	$\mathcal{O}(5.57e242)$	Inf
810	2195	55	$\mathcal{O}(8.91e272)$	Inf
910	2466	61	$\mathcal{O}(1.3425e303)$	Inf

Task 5

Task 5.a

Our method is to calculate the expected amount of monomials in $\mathbb{F}_p$ of degree $p-1$, which is then the expected amount of disequations we need, as every disequation will give rise to a linear equation and we need a full rank matrix to solve that problem. The estimation lies in the fact that it's possible that some disequations give rise to linear dependent equations, in which case we need more disequations. The amount of monomials is given by the 'stars and bars' method. Using that method, we find that if we have n variables and m powers, the amount of monomials is given by $\binom{m+n-1}{n-1}$. Now we want to know the amount of monomials of degree $p-1$ or *less*, so we have to take the sum:

$$\sum_{i=1}^{p-1} \binom{n+i-1}{n-1}. \tag{19}$$

We start from $i = 1$, as we're not counting the monomial 1, which is handled separately.

If the solution from the linear system is not unique, we have to do extra work to verify which of the solutions is the actual solution of the system of disequations.

5.c

The table below shows the results for $n = 8$:

p	Expected # eqs.	Needed # eqs.
3	65	66
5	1000	1001
7	8007	8008

We see that the total number of needed equations is almost the same as the expected number of equations calculated above. The discrepancy can be explained by the fact that by chance, some linearized equations can be linearly dependent.

Task 6

For proof of completeness, we have to show that when both Peggy and Victor follow the protocol, the `Verify` of Victor will always return `true`. We have two cases: $c = 0$ and $c = 1$:

- $c = 0$: then the response is C. By design, $M = C \cdot A \cdot C^{\text{trunc}(n)}$, so the verify will always return `true`.
- $c = 1$: then the response is $\mathbf{m}$. We then find $M \cdot \mathbf{m} = C \cdot A \cdot C^{\text{trunc}(n)} \cdot \left(C^{\text{trunc}(n)}\right)^{-1} \mathbf{s} = C \cdot A \cdot \mathbf{s} \not\equiv \mathbf{0}$, as by design $C \cdot A \cdot \mathbf{s} \not\equiv \mathbf{0}$.

Task 8

We first note that finding $\mathbf{t}$ is sufficient for finding $\mathbf{s}$ due to the Moore-Penrose inverse of A:

$$\mathbf{s} = A^{+} A \mathbf{t}, \tag{20}$$

with A^{+} the pseudo-inverse. Secondly, we note that when receiving $\mathbf{C}$, we actually get m disequations, as $\mathbf{C} \cdot \mathbf{t} \not\equiv \mathbf{0}$.

Once we have enough disequations, we can all linearize them via the methods described in Task 5, which will then give us a system of linear equations, which is will yield $\mathbf{t}$. As explained above, from $\mathbf{t}$, we can derive $\mathbf{s}$. The method from Task 5 only works when N is prime.

Task 9

The most important point to see is that for a signature to succeed, the following relations between A, M, C and $\mathbf{m}$ should hold:

$$\begin{cases} M = C \cdot A \cdot C^{\mathrm{trunc}(n)} \\ M \cdot \mathbf{m} \not\equiv \mathbf{0} \end{cases} . \tag{21}$$

We note that neither of the two conditions requires $\mathbf{s}$, explicitly, which makes it possible to attack the scheme. The most difficult problem is finding an $\mathbf{m}$, given an M, so we evade that by choosing an M and $\mathbf{m}$ such that $M \cdot \mathbf{m} \not\equiv 0$.

Then the next step is finding the C matrix such that the first condition holds. This is also difficult, as the equation is non-linear due to the fact that $C^{\mathrm{trunc}(n)}$ is derived from C. We can make this equation easier by choosing $C^{\mathrm{trunc}(n)} = I$, such that in the scenario that $m = 3$ and $n = 2$, we get:

$$\begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \\ m_5 & m_6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & c_1 \\ 0 & 1 & c_2 \\ c_3 & c_3 & c_5 \end{bmatrix} \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \\ a_5 & a_6 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} . \tag{22}$$

Here we can see that we have 5 unknowns in the C-matrix and 6 unknowns in the M-matrix, but that we only have 6 equations. We can solve that issue by choosing some unknowns fixed values, such that we remain with 6 unknowns. This is not generally solvable, as (1) the system must be solvable and (2) we're working with a non-prime order of the ring, which means that not all elements have an inverse. Furthermore, if we choose fixed values for unknowns, we give precedence to the m-values, as that will make it easier to find the $\mathbf{m}$ later.

This method is however not generalizable and due to time constraints, I did not implement it.