丁津泰教授的《现代密码学》笔记

周康韫

April 8, 2024

Contents

1 Lecture 5 Multivariate Cryptography (I)

Firstly, we shall review the foundation of classical (asymmetric) cryptography. We have introduced, in the previous lectures, about 2 difficult problems that have been currently assumed to be unsolvable in polynomial time: Large number factorization problem, and Discrete logarithm problem. The difficulty of these two problems are the foundations of RSA and Diffie-Hellman encrytion scheme. You might want to ask a question: what will happen if these two problems are solved in the future? In short, the answer for this problem is PQC (Post-Quantum Cryptography).

This lecture will now move on to PQC. We have no time to introduce PQC in details, but I would like to mention a few names. Firstly, Richard Feynman, perhaps the most important figure in anything related to quantum and most interesting scientist, who described the idea of quantum computing in 1981, at a conference hosted by MIT' s Laboratory for Computer Science. I highly recommend the book *Surely You're Joking, Mr. Feynman!* for all of you to read.

As the idea of quantum computing appeared in 1980s, an important question for this idea is: what problems can quantum computing do better (than classical computing)? In 1980-90, nobody known the answer, and quantum computing is just an interesting idea. However, in 1994, Peter Shor from MIT suddenly published a paper, which gave an expected answer for that question.

The answer is, quantum computer can solve factorization problem as well as DL in polynomial time.

In Shor's original paper [2], he described an algorithm that works only on quantum computer, which solves factorization and DL in polynomial time. This algorithm, known as Shor's algorithm, is the earliest quantum algorithm in the history.

In 2000, Issac Chuang came to MIT from IBM. This man spent 15 million USD to build a quantum computer (prototype) with 7 "Qbit"s, to complete one simple task: factorize

$$15 = 3 \times 5,$$

using Shor's algorithm. At the same year, I got a tenure in the USA, and I felt like it is time to try something new other than algebra (representation theory, more precisely), so I dived into the area of PQC, which aim to find replacement for encryption schemes based on DL, factorization (in Zn or elliptic curve group) in the future.

in 2006, the first PQC conference had been held and in 2008 there was the second. At this time, people have not paid much attention on PQC, but everything changed in 2015 Aug 19th, when the US government announced that they want to replace the current encrytion standard. This decision was out of nowhere, and we don't know why this happened, but anyway it made PQC a hot field. NIST(National Institute of Standards and Technolog) set the deadline on Nov 31st, 2017, and mathematicians all around the world started to submit PQC schemes to them. On July 2022, NIST announced the first group of winners:

CRYSTALS-Kyber, lattice crpto, for PKE/KEM;

CRYSTALS-Dilithium; FALCON, lattice crpto, for signature;

SPHINCS, Hash-based, for signature.

And June 2023, NIST announced another round of qualifiers, including UOV (Unbalanced Oil and Vinegar) scheme. This is a crpto scheme based on multivariate equation. I am very confident that UOV will be standardized, perhaps in 10 years, every PC will be equipped with UOV scheme. In this course, we will first cover the idea of Multivariate, and then Lattice.

(I should mention here that 3-SAT problem, a known NPC problem, is *more or less* equivalent to solving multivariate equation in $GF(2)$. This equivalence is tricky because the coefficient size before and after problem reduction might change drasticly, and it is hard to define whether one problem can be reduced to another in polynomial time.)

Now we start talk about Multivariate equation in encryption. In most cases, we consider quadratic multivariate equations. Why not higher degree? Here is an easiest example:

$$x_1x_2x_3 = 5$$

One can easily transform it to a quadratic equation with 1 more variable y:

$$x_1x_2 - y = 0; x_3y = 5.$$

It is obvious that all multivariate equations with degree higher than 2 can be reduced to quadratic equation by adding more variables, so it is meaningless to use higher degree.

1.1 Tame Transformation and Encryption

Since it is well-known that solving multivariate equation is computationally difficult (or more officially, NPC-hard), many cryptographers, including Diffie and Tsutomu Matsumoto had tried to design crpto schemes based on it. An idea is to use a special type of transformation, called Tame transformation:

$$F(x_1, x_2, ..., x_n) = \begin{pmatrix} x_1 \\ x_2 + f_2(x_1) \\ x_3 + f_3(x_1, x_2) \\ x_n + f_n(x_1, ..., x_{n-1}) \end{pmatrix}$$

where f_i are quadratic polynomials of $(x_1, ..., x_i)$.

Tame transformation is a non-linear transformation of vector x to y which is easy to find inverse. Since it preserves x_1, it is easy to find inverse: just calculate $x_2 = y_2 - f_2(x_1)$; and $x_3, ..., x_n$ one by one. As a result, one may try to design a public key cryptosystem based on Tame transformation as the following:

- Private key:T, F, S, where T and S are invertible linear transformations. (Attention: when stating T and S are invertible here, it means they are invertible limited on their image set. which means, they can be only injective but not subjective.)
- Public key: P, where $P = T \circ F \circ S$.

This looks pretty nice: the two linear transformations T, S hides the vulnerable part x_1. With T, S known, one can easily solve the equation

$$y = P(x) = T \otimes F \otimes S(x) \iff P(x) = T^{-1} y S^{-1}.$$

However, with T, S unknown, it should be hard to directly solve the multivariate quadratic equation $P(x) = y$. (This is not true.)

Now the problem is: Can you break it? The answer is ”yes”.
(Warning: we may use row vector instead of column vector in this

Proposition 1.1. *Let $P(x) = (p_1(x), ..., p_n(x))$. Then, $\exists (a_1, ..., a_n) \neq 0$, s.t. $\sum_{i=1}^n a_i p_i$ is a non-trivial linear transformation.*

Proof.

$$F_1 = proj_1 \circ T^{-1} \circ P$$

is such a linear transformation, where $proj_1$ denotes taking the first element. □

You might be curious that how do we find such coefficients $(a_1, ..., a_n)$. We demonstrate using an example.

In this example, we work in GF(5). Let

$$F(x_1, x_2) = \begin{pmatrix} x_1 \\ x_2 + 2{x_1}^2 \end{pmatrix};\ S = \begin{pmatrix} 2 & 3 \\ 1 & 0 \end{pmatrix}; T = \begin{pmatrix} 1 & 2 \\ 1 & 3 \end{pmatrix}.$$

$$P(x_1, x_2) = T \circ F \circ S(x_1, x_2) = \begin{pmatrix} (2x_1 + 3x_2) + 2(x_1 + 2(2x_1 + 3x_2))^2 \\ (2x_1 + 3x_2) + 3(x_1 + 2(2x_1 + 3x_2))^2 \end{pmatrix}.$$

It is easy to see that $3p_1 - 2p_2 = 2x_1 + 3x_2$. Now, one can reduce the number of variable to 1, and directly solve the equations.

This example seemingly over-simplified the problem, but it actually captured everything we need to do here. For more complicated P (with more variables), generally we have the following equations:

$$P_1 = q_1 + l_2;$$

$$P_2 = q_2 + l_2;$$

$$...$$

$$P_n = q_n + l_n.$$

Where l_n are purely linear and q_n are purely quadratic. We already know that there is some linear transformation of P is linear:

$$A \cdot P = (a_1, ..., a_n) \cdot (P_1, ..., P_n) = linear$$

which means A will eliminate all quadratic terms in P. Then one can simply solve a linear equation for coefficients of $x_i x_j$ (so that the coefficients are all 0):

$$A \cdot (q_1, ..., q_n) = 0.$$

Though the number of equations is much larger than n, the kernel is at least 1 dimensional (by prop 4.1). It might appear a kernel with dim > 1, then one can solve the equation set up by coefficients of linear terms:

$$A \cdot l = (a_1, ..., a_n) \cdot (l_1, ..., l_n) \neq 0.$$

Combine the solution of this "equation" and the previous equation, one can get the desired solution $A = (a_1, ..., a_n)$ such that

$$(a_1, ..., a_n) \cdot (P_1, ..., P_n) = \sum_{i=1}^{n} h_i x_i; h \neq 0.$$

In another word, we have proven that we can always recover some none-zero linear combination of $x_1, ..., x_n$ from $P(x_1, ..., x_n)$.

Once this step is done, the remaining steps are easy: firstly, one reduce the number of variables of P, by simply replacing one variable x_j ($h_j \neq 0$) by linear combination of other variables.

It might not be that easy to see that after this reduction, P is still a composition of one linear transformation, one tame reduction and another linear combination. We first show this is true when $A \cdot P = x_1$ exactly. If so, simply take $x_1 \equiv 0$ in P. Since $S((0, x_2, ..., x_n)^T)$ is an invertible linear transformation of $(x_2, ..., x_n)$, it is clear that P remains similar structure. Now for general $A \cdot P$, we can always use a linear transformation (which can be absorbed into S) to make it be x_1.

Based on what we have shown, we can repeat the process of variable reduction again and again, until there is only 1 variable, and then solve the quadratic equation directly. The "Tame transformation scheme" is easily broken in polynomial time.

1.2 Minus Method

We may conclude on the failure of the above scheme: two linear transformations are not enough to "conceal" or to "hide" the linear part $F_1(x) = x_1$. One may try to fix this problem by a "Minus method": delete the first entry, or the first k entries of F, for example:

$$F(x) = \begin{pmatrix} x_3 + f_3(x_1, x_2) \\ x_4 + f_4(x_1, x_2, x_3) \\ ... \\ x_n + f_n(x_1, ..., x_n) \end{pmatrix}.$$

Then there will be no linear entry in f. Notice after this "minus method", F is no longer an injection and one would not be able to use it as an encryption scheme, but only can be used in signature system. Is this enough to ensure the security?

The answer is still a "no", but the attack method must be modified. Notice that

$$P_j = q_j + l_j, \quad j = k + 1, ..., n$$

still holds here, where q_j is a purely quadratic function of $(x_1, ..., x_j)$, i.e. a quadratic form with rank j on $\mathbb{R}^n$. More precisely, q_j has the following form:

$$q_j = \begin{pmatrix} Q^j & 0 \\ 0 & 0 \end{pmatrix},$$

where Q^j is a $j \times j$ symmetric matrix. Hence, we will have quadratic forms of rank $k+1, k+2, ..., n$ in $F(x)$. Notice that linear transformation S will not change the rank for each quadratic form (it act on q_j as $q_j \to S^T q_j S$). Thus, if we can find a linear combination of q_j (this is a linear transformation on P and can be absorbed into T), such that the result quadratic form has a lowest rank ($\leq k+1$), then we can reduce the problem just like in the previous section by reducing the number of variables. This is known as the rank attack (or MinRank problem attack), see [1]. Diffie and Shamir had tried to fix this problem, but none of their attempt worked.
Remark: Professor had claimed this attack quite easy to figure out ourselves in the lecture, but I seriously doubt this claim.

1.3 Lifting from Finite Field

Now we introduce another construction based on multivariate quadratic equation. This idea is described by Tsutomu Matsumoto in 1988 and (sadly) broken in 1995.

Consider the following maps:

$$GF(q)^n \xrightarrow{S} GF(q)^n \xrightarrow{F} GF(q)^n \xrightarrow{T} GF(q)^n$$

Where we want S and T be linear and invertible, F quadratic and invertible. Notice that $GF(q)^n$ can be viewed as a finite field $GF(q^n)$ in the following way:

$$c = (c_0, ..., c_{n-1}) \in GF(q)^n \leftrightarrow c = c(x) = \sum_{i=0}^{n-1} c_i x_i \in GF(q)[x]/g(x) \cong GF(q^n),$$

where $g(x)$ is a degree n irreducible polynomial in $GF(q)$. In our context here, $q = 2^k$ for some k.

We construct F in the following manner: pick $\theta \in \{1, 2, ..., n-1\}$ s.t. $\gcd(q^\theta + 1, q^n - 1) = 1$. This ensures that

$$F' : GF(q^n) \to GF(q^n), c \mapsto c^{q^\theta + 1}$$

is a 1 to 1 map (since it is a 1 to 1 map in the multiplication group $GF(q^n)*$). We claim that if we lift this map to $F : GF(q)^n \to GF(q)^n$ (or by abuse of notations, consider F' as a map on $GF(q)^n$), then it is a quadratic map.

Proof. Since $c \mapsto c^q$ in field with $char = q$ is a field endomorphism (known as Frobenius map), $c \mapsto c^q$ is obviously an invertible linear map. Hence $c \mapsto c^{q^\theta}$ is an invertible linear map. Hence, we have:

$$F(c) = c^{q^\theta} \times c;$$

This is a quadratic map, by polynomial multiplication (all coefficients are purely quadratic terms). $\square$

Lecture 5 ends here.

2 Lecture 6 Multivariate Cryptography (II)

2.1 Rank Attack

In this lecture, we shall first explain the rank attack performed on the encryption scheme based on some triangular maps. Recall the triangular transformation (with the first $k-1$ rows eliminated):

$$F : \begin{pmatrix} x_1 \\ ... \\ x_n \end{pmatrix} \mapsto \begin{pmatrix} x_k + f_{k-1}(x_1, ..., x_{k-1}) \\ ... \\ x_n + f_{n-1}(x_1, ..., x_{n-1}) \end{pmatrix}.$$

And denote the quadratic forms in the i-th entry of P as P_i. (We can simply drop the linear terms here.) Since the linear transformations T and S act on each quadratic terms as the following:

$$S : f_i \mapsto f_i' := S^T f_i S;$$

$$T : \begin{pmatrix} f_{k-1}' \\ ... \\ f_{n-1}' \end{pmatrix} \mapsto T \begin{pmatrix} f_{k-1}' \\ ... \\ f_{n-1}' \end{pmatrix} = \begin{pmatrix} p_{k-1} \\ ... \\ p_{n-1} \end{pmatrix}$$

It is clear that any invertible linear transformation S will not change the rank of each quadratic forms f_i, and T will only mix the $rank = k_1, ..., n-1$ quadratic forms $f_{k-1}' ... f_{n-1}'$ together. Rank attack aims to find a p_i with highest rank $(n-1)$, and find a_j such that

$$Rank(a_j p_i + p_j = n-2).$$

We first show that why this strategy works, and then show how this strategy works.

Suppose we have already find such a set of $q_j = p_i + a_j p_j$ (totally $n-k$ of them). We will assume (for now) that each $q_j = a_j p_i + p_j$ is a linear combination of $f_{n-2}', ..., f_{k-1}$, which means we have eliminated the terms f_n' completely. Now, we perform rank attack on the $n-k$ quadratic forms of rank $n-2$ again the reduce the rank and size by 1. If we are lucky enough, we will finally arrive at a rank $k-1$ quadratic form which is a linear combination of f_i's. We hope this is f_{k-1}', and finding f_{k-1}' will in fact completely break the system.

A lot of problem might arise in this process.

Problem 1: Why (and How) we can find $a_j p_i + p_j$ with lower ranks?

Answer: We can assume that in general all p_i has rank $n-1$, in a probabilistic sense. For fixed p_i, p_j we try to solve the following equation:

$$rank(p_i + p_j x) \leq n-1.$$

This is equivalent to:

$$\forall M, det(M) = 0$$

where M is $n-2$ dimensional minus (n-2 维子阵) of $p_i + p_j x$. The equation $det(M) = 0$ will give a polynomial of x with $deg \leq n-1$. We will get different polynomials of x by different M, and by finding their gcd, we can find a low degree polynomial and solve it for $x = a_j$.
It might happen that the rank equation $rank(p_i + p_j x) \leq n-1$ has more than 1 solutions, and we are only interested in the only solution that eliminates f'_{n-1}.

Here is an example of undesired solution:

$$f'_{n-j} = \begin{pmatrix} I_{n-j} & 0 \\ 0 & 0 \end{pmatrix}; p_1 = f'_{n-1} + f'_{n-3}; p_2 = f'_{n-1} - 3f'_{n-3}.$$

It is clear that $q = p_1 + p_2 = 2f'_{n-1} - 2f'_{n-3}$ and $q' = p_1 - p_2 = 4f'_{n-3}$ are all solutions to $rank(p_1 + xp_2) \leq n-1$. However, we are only interested in the latter solution since it eliminates the f'_{n-1} term, but not the previous one.

(Notice that this example also shows that rank of p_i might not be $n-1$ even if it contains f'_{n-1} term.)

This can happen in the process of rank attack, however, we can always check whether a solution is "undesirable": by checking whether some linear combination of a solution and another solution will increase the rank again to $n-1$. In the previous example, one will easily find that $rank(q+q') = n-2$, which implies there must be f'_{n-1} term in q or q'. If this happens, we will discover and abandon the "undesirable solution".

Problem 2: If such problems can happen, why rank attack still works?

Answer: It still works since **in a probabilistic sense**. In cryptography, we never require our attack to work 100% of time (deterministically), we only hope it to work 20% of time, or even 2% of time and we are satisfied.

We will now give a sketch of explanation of why rank reduction attack can really complete in polynomial time, with a high probability to success. The whole attack requires $\leq n$ steps of rank reduction, and in each step, one need to solve $\leq n$ rank equations, each of them is finding gcd of $\leq n^2$ polynomials). The complexity of this process is controlled by, say, $O(n^5)$, and we only need to check that the "confirmation step" for each solutions to be "desirable" (the highest rank term has been eliminated) can complete in polynomial time with a high probability. We need to estimate the probability that our checking works. We look at the following equation:

$$rank(k(f'_{n-1} + \sum_{i=k+1}^{n-2} \alpha_i f'_i) + \sum_{i=k+1}^{n-2} \beta_i f'_i)) = n - 1.$$

If the checking does not work, then it implies that $\forall k \neq 0 \in F_q$, we always have $rank = n - 1$. Again, this situation can often happen (counterexamples are easy to construct, if we take each f'_i diagonal). The actually attack follows a backward search strategy. Instead of checking everytimes when we meet a solution (a linear combination of quadratic forms that lowers the rank), we always take some such solutions and proceed to the next step of rank reduction. If there are some "undesirable solutions" in it, then at some step, it is **very likely** that we will encounter the situation that a rank equation cannot be solved, which means at least one of the two quadratic forms is flawed as it contains higher rank term.
Now we try to explain why this process is likely to end in polynomial time.

Proposition 2.1. *Expectation of number of undesirable solution for a rank equation to yield is approximately bounded by $\frac{1}{p-1}$, where p is the size of field.*

We will not prove this statement (I haven't found a way to rigorously prove this), so we will just use it as a truth from experiments. As a result of this proposition, we can assume that after the first step of rank reduction, we will have in total around $\frac{p}{p-1}n$ solutions (forms with $rank < n - 1$), with exactly $n - 1$ of them are desirable solutions. What we try to do now is to estimate the time we need for discovering one of the solutions to be undesirable (which means, we have removed one undesirable solution). If this process can finish in polynomial time, then since there are in total at most $O(n)$ undesirable solutions, in polynomial time we can remove all undesirable solutions and there will be only desirable ones.

How exactly do we confirm that one solution is undesirable? Besides trying to find an x such that $rank(p_i + p_j x) < rank(p_i) = rank(p_j)$, we can also check if for any $x \in F_p$, $rank(p_i + p_j x) > rank(p_i) = rank(p_j)$. If the later happens, or the previous not stands (cannot find solution), we will immediately know that at least one of p_i, p_j is undesirable. (We have explained above.)

Proposition 2.2. *The probability for this checking to fail is strictly less than 1 and the upper bound is only related to the field size p. I guess it is also* $\frac{1}{p-1}$.

If the checking fails successively for a "depth" of a (which is to say, if a $rank = w$ solution is undesirable and we failed to discover this until we reduce the rank to $w-a$), which is an event of probability less than $(p-1)^{-a}$. In this situation, we find that in total $a+1$ $rank = n-2$ solutions can be undesirable. We can assume there are around $\frac{p}{p-1}a$ solutions that might replace such solutions (among $a+1$ $rank = n-2$ solutions there are likely to be more than $\frac{p-2}{p-1}$ of them to be "the only choice" in the rank equation and confirmed desirable, while others might be replaced by other solutions in rank equation). This makes us satisfied, since the situation we must one by one check in total less than $p^{\frac{1}{p-1}a}$ situations is only $\frac{1}{p-1}^{a}$, so the probability of removing an undesirable solution takes more than polynomial time is exponentially low, which makes the whole algorithm polynomial time.(Notice that this is not a probabilistic algorithm, but a deterministic one: we will always know in some step that one undesirable solution is undesirable. An analogy is, we toss a coin and hope it land with face on top, and we know whether each toss get face or tail. We hope to succeed in total around $\frac{1}{p-1}n$ times, this obviously can be done in polynomial time with good chance.) This completes the proof.

One might still doubt that the "checking" may not work, that in extreme situation, one may arrive and some form with $rank = k-1$ or even $< k-1$, but it still an undesirable solution. For example, if $f'_{n-i} = \begin{pmatrix} I_{n-i} & 0 \\ 0 & 0 \end{pmatrix}$, then $rank(f'_{n-1} - f'_{n_2}) = 1 < k-1$ and the checking will not work. Well, the interesting point here is that such a solution with low rank is exactly the solution we want, at least as much as we want to find f_{k-1}, since the low rank of any quadratic form will allow us to perform variable reduction! And when we recall our algorithm, one will find that we only stop and go backward to remove some undesirable solution only when we cannot precede to lower the rank: some rank equation has no solution at all. So we don't necessary arrive a f'_{k-1} or some f'_{k+a}–the algorithm might yield some different low rank quadratic form, which is as good as f'_{k-1} or even better!

Remark: This is not a rigorous proof: I made some unverified assumptions, and I did not estimate the probability precisely but only give an extremely loose bound. If you find better proof, please contact me.

2.2 Linearization equations

Now we will move on to field lifting scheme described in the previous lecture. We have explained how it works in the last lecture, and now we introduce a method of breaking it: Linearization Equation Method.

This method is discovered by a French mathematician Jacques Patarin in 1999. Before this surprising discovery, this encryption scheme had nearly been accepted as a encryption standard by Japan.

We recall the encryption scheme:

- Private key: T, θ, S;
- Public key: $P = \begin{pmatrix} p_1 \\ ... \\ p_n \end{pmatrix}$ Where each p_i are quadratic funtions (again, we drop linear terms here.)
- Plaintext: $x' = (x'_1, ..., x'_n)$;
- Cyphertext: $y' = p(x'_1...x'_n) = (y'_1...y'_n)$.

We will use x'_i, y'_i to denote some specific text and x_i, y_i to denote variables. Now we present the central claim of linearization equation.

For all fixed p, there exist a list of linearly independent equations:

$$\sum_{i,j=1}^{n} a_{i,j} x_i y_j = 0, \forall y = p(x).$$

If we have found a large list of such equations, we can easily recover x' from y'. It is obvious that if we find n such equations, then the solution for x' given y' is unique. Even if we only have $n - t$ such linearly independent equations, we can still list all the solutions in the kernel and try them one by one, when t is small enough. Hence, finding such linearization equations will completely break the system.

Now we are left with two problems: (I) How many such equations exist? (II) How do we find such equations? In real life, the second problem is more important for us, since if it works, it directly implies that enough such equations exist.

We shall first address the second problem, since the answer is surprisingly easy. Recall that with public key P, we can always calculate y' from any x' we want. Then, one can try to solve $(a_{i,j})$ by solving the following equation:

$$\sum_{i,j} x'_i y'_j a_{i,j} = 0$$

Given any x', y', this is a linear equation of size $\frac{n(n-1)}{2}$, and it is very likely that with enough trials for different x', y', we can find nearly all such linearization equations. (It is easy to calculate the probability of collecting all linear independent sets of $(a_{i,j})$ in the first k trials by markov chain: $E(k) = N \log N$, assuming the probability of getting each sets $(a_{i,j})$ are equal if we choose x', y' randomly.)

Now we are left with the only problem: how many such equations exist?

Firstly, we observed that their is a one to one map from linearization equations of $P = T \circ F \circ S; y = P(x)$ to linearization equations of $y = F(x)$, by the folllowing construction:

$$x^T A y = 0 \iff (Sx)^T ((S^{-1})^T A T)(T^{-1} y) = 0.$$

Hence, we only need to estimate the number of linearization equations for $F : x \mapsto y = x^{q^\theta + 1}$. From this we have:

$$y^{q^\theta - 1} = x^{q^{2\theta} - 1}$$
$$\iff y^{q^\theta} x - y x^{q^{2\theta}} = 0.$$

Here (by abuse of notations), x, y are in the extended field F_{q^n}, so y^{q^θ} is actually a linear transformation of y and $x^{q^{2\theta}}$ is actually a linear transformation of x. Hence, the above is actually a linearization equation, and we need to find what is its rank. Instead of counting rank, we use a classical technique: count the dimension of kernel (solution space of x). (In fact, this is what we really care about.)

We claim that for every fixed y, there are exactly q solutions for x, containing 1 trivial solution $x = 0$, when n is a prime number, which is to say that the kernel is 1-dimensional.

Proof. We only need to estimate the number of solutions for equation

$$x^{q^{2\theta - 1}} = c$$

for some $c \in F^*_{q^n}$. From the construction we know at least 1 solution (x, y) exists, so we only need to prove for the upper bound, which means we can just assume $c = 1$. Then number of solutions for $x^{q^{2\theta-1}} = 1$ is $\gcd(q^{2\theta-1}, q^n - 1)$, since the order of the multiplication group $F_{q^n}^*$ is $q^n - 1$. Notice that $\gcd(q^{\theta+1}, q^n - 1) = 1$ by definition of θ, so we only need to find $\gcd(q^{\theta-1}, q^n - 1)$. This is just $q^{\gcd(\theta,n)}$ (**The proof of this is left as homework**) and by definition n is prime, so number of non-zero solutions is exactly $q - 1$. □

Now we have completely described and proven the attack by linearization equations towards the previous encryption scheme, where Matsumoto and Imai had been defeated by Patarin. Later there are other on multivariate encryption schemes, which are mainly two types: those insist on the construction of $x^{q^{\theta-1}}$ has quadratic function of x, since it is very easy to store the private key when the quadratic map is nothing but a large number $\theta < n$. One of the ideas involving hiding the field (HFE) had appeared, and completely defeated in 2021 by Prof. Ding. Another idea is really genius: it is based on linearization equations (more or less as a private key). This method is developed by Patarin, which is called OV, Oil and Vinegar scheme; this scheme is later broken by Shamir (the ”S” in RSA), and later modified to be UOV, unbalanced Oil and Vinegar scheme, which is considered very safe at least for now. We will move on to OV and UOV in the next lecture.

Lecture 6 ends here.