COS 533: Advanced Cryptography Princeton University
Lecture 5 (February 18)
Lecturer: Mark Zhandry Scribe: Aleksa Milojevic

Notes for Lecture 5 - Algebraic Tools in Cryptography

1 Introduction

In this lecture, we introduce and investigate algebraic tools used in Cryptography, such as multiplicative groups over finite fields and elliptic curve groups. These are commonly used to achieve public-key encryption (PKE), code obfuscation, etc. We will try to give a broad overview of the techniques used.

2 Cryptographic groups

First, we begin by defining the mathematical notion of the group.

Definition 1. *A group $\mathbb{G}$ consists of an underlying set $\mathbb{G}$ and of a binary operation $\cdot : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ which satisfies the following properties:*

- *The operation $\cdot$ is associative, i.e. $a \cdot (b \cdot c) = (a \cdot b) \cdot c$, for all elements $a, b, c \in \mathbb{G}$.*
- *There is an element of $\mathbb{G}$ called the identity, and denoted by $1 \in \mathbb{G}$, for which we have $1 \cdot a = a \cdot 1 = a$ for all elements $a \in \mathbb{G}$.*
- *For all elements $a \in \mathbb{G}$, there is an inverse element $a^{-1} \in \mathbb{G}$ for which $a \cdot a^{-1} = a^{-1} \cdot a = 1$.*

The operation $\cdot$ is usually called multiplication. Moreover, we will mostly consider groups in which the operation $\cdot$ is commutative, i.e. $a \cdot b = b \cdot a$ for all $a, b \in \mathbb{G}$.[1]

In the cryptographic context, we will require the multiplication operation to be efficient. In other words, we need to be able to compute $a \cdot b$ efficiently when $a, b \in \mathbb{G}$ are given.

Intuitively, cryptographic groups are the groups in which no other operation except multiplication can be efficiently performed. One of the usual assumptions we make is the hardness of the Discrete Logarithm problem.

[1] Such groups in which $\cdot$ is commutative are sometimes called *Abelian* groups.

Definition 2. *(Discrete Logarithm assumption) Let $\mathbb{G}$ be a group. We say that $\mathbb{G}$ satisfies the Discrete Logarithm assumption if given elements $g, h = g^a \in \mathbb{G}$, where $a \in \mathbb{Z}$, it is impossible to compute a efficiently.*[2]

Discrete Logarithm assumption is usually taken as the minimal assumption on the group. However, we will often need to make even stronger assumptions on the group $\mathbb{G}$ in order to derive useful results. To find groups that satisfy the Discrete Logarithm assumption, let us consider two hypothetical attacks on the Discrete Logarithm problem.

One way to attack Discrete Logarithm problem is to try all possible values of a in a brute-force manner. Let the subgroup generated by g be $\langle g \rangle = \{1, g, g^2, \cdots, g^{k-1}\}$ where $g^k = 1$.[3] So, if k was polynomial in λ, we could just try out all possible values of $a \in \{0, 1, \dots, k\}$ and check if $g^a = h$. Thus, such an a could be found efficiently, and thus Discrete assumption would not hold. Therefore, we see k must be superpolynomial in λ. The ideal case, in which the size of $\langle g \rangle$ is maximal, happens when $\langle g \rangle = \mathbb{G}$. In this case, we say g is a *generator* of our group. Moreover, we say $k = |\langle g \rangle| = |\mathbb{G}|$ is the *order* of the element g and of the group $\mathbb{G}$. We denote it by $\mathrm{ord}(g) = |\langle g \rangle|$.

There is another attack that may be directed against Discrete Logarithm problem in case $\mathrm{ord}(g)$ has no large prime factors. To show how it works, we will assume that $\mathrm{ord}(g) = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_l^{\alpha_l}$, for some primes $p_1 < p_2 < \cdots < p_l$, and that $p_i \leq \mathit{poly}(\lambda)$. The idea is to determine a with respect to each of $p_j^{\alpha_j}$ separately and then use the Chinese Remainder Theorem to compute $a \mod \mathrm{ord}(g)$.

Given $g, h = g^a$, the attack proceeds by computing $g_j = g^{\mathrm{ord}(g)/p_j}, h_j = h^{\mathrm{ord}(g)/p_j}$. It is simple to see that $g_j^a = h_j$, just as $g^a = h$, and that $g_j^{p_j} = g^{\mathrm{ord}(g)} = 1$. Therefore, $\mathrm{ord}(g_j) = p_j$, and we can simply check the all the values $a_j \in \{0, 1, 2, \dots, p_j - 1\}$ to find the one for which $g_j^{a_j} = h_j$. This can be done efficiently, as we only need to check p_j values, and $p_j = \mathit{poly}(\lambda)$. Note that a also satisfies $g_j^a = h_j$, and thus we must have $a \equiv a_j (\mod p_j)$.

Now, we try to find $a(\mod p_j^2)$. To do this, we compute $g_j = g^{\mathrm{ord}(g)/p_j^2}, h_j = h^{\mathrm{ord}(g)/p_j^2}$ and look for a value of a_j which makes $g_j^a = h_j$. As before, finding any value a_j satisfying the above equation amounts to finding $a(\mod \mathrm{ord}(g_j))$. As now $\mathrm{ord}(g_j) = p_j^2$, this means that we are actually looking for the value $a(\mod p_j^2)$. The trick is that we do not need to check all of p_j^2 possibilities as we did before: we have already determined $a \mod p_j$ and we need to consider only the numbers $a \mod p_j, p_j + a$

[2] Note that the inverse operation of the discrete logarithm procedure would be computing $h = g^a$ from (g, a), where $g \in \mathbb{G}$, $a \in \mathbb{Z}$. This can be done efficiently using the technique of repeated squaring, using only $O(\log a)$ multiplication operations in $\mathbb{G}$. Moreover, note that a is determined only modulo $\mathrm{ord}(g)$, and if we wanted to be precise we should assume that it is hard to compute $a \mod \mathrm{ord}(g)$. However, this difference bears no practical importance.

[3] If $\mathbb{G}$ is finite, such integer k must always exist and is called the order of the element g.

$\mod p_j, \cdots, (p_j - 1)p_j + a \mod p_j$. Thus, in time $O(p_j)$ we can find $a(\mod p_j^2)$. This process can be continued for any higher power of p_j, thus giving the value $a(\mod p_j^{\alpha_j})$ in time $O(\alpha_j \cdot p_j)$.

Once we found $a(\mod p_j^{\alpha_j})$ for all j, it is enough to use Chinese Remainder theorem to compute $a \mod \text{ord}(g)$. However, this means we can solve Discrete Logarithm problem efficiently, which presents a problem.

Therefore, the ideal case would be if we had a group $\mathbb{G}$ and its generator g such that g had large prime order p.

Note that in this case we must have $\mathbb{G} \cong \mathbb{Z}_p$, where $\mathbb{Z}_p$ is the group consisting of residue classes modulo a prime p and the associated operation is addition modulo p. As the operation on $\mathbb{Z}_p$ is just addition, we see that exponentiation g^a in the group $\mathbb{Z}_p$ amounts to simply multiplying the corresponding elements. Therefore, Discrete Logarithm problem can be reduced to modular division, which is efficiently computable. The conclusion is that Discrete Logarithm is not hard in $\mathbb{Z}_p$. Thus, the only way for Discrete Logarithm to be hard in $\mathbb{G}$ is if the isomorphism $\mathbb{G} \to \mathbb{Z}_p$ was hard to compute.[4]

Now, we will try to construct such groups. We consider two main classes of such groups: multiplicative groups of finite fields and elliptic curve groups.

3 Multiplicative groups over finite fields

Definition 3. *A finite field $\mathbb{F}$ is consists of an underlying set $\mathbb{F}$ together with two operations, $+, \cdot : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$, which satisfy the following conditions:*

- *The set $\mathbb{F}$ is a commutative group with respect to the operation $+$, and it has identity $0 \in \mathbb{F}$.*
- *The set $\mathbb{F} \backslash \{0\}$ is a commutative group with respect to the operation $\cdot$, and it has identity $1 \in \mathbb{F}$.*
- *The operations $+$ and $\cdot$ satisfy distributivity, i.e. we have $a \cdot (b + c) = a \cdot b + a \cdot c$ for all $a, b, c \in \mathbb{F}$.*

Example 4. *Let q be a prime number. Then, the set $\mathbb{Z}_q$ is a field with operations of modular addition and modular multiplication. This field contains q elements.*

[4] It is clear that the isomorphism $\mathbb{Z}_p \to \mathbb{G}$ is efficiently computable, because such an isomorphism is given by $a \mapsto g^a$, for $a \in \mathbb{Z}_p$, which can be computed using repeated squaring, as discussed earlier. If the isomorphism $\mathbb{G} \to \mathbb{Z}_p$ was efficiently computable as well, we could just pass to $\mathbb{Z}_p$, solve the Discrete Logarithm problem there, and efficiently return to $\mathbb{G}$ after that.

More generally, for any prime power q^k, there exists a unique field $\mathbb{F}_{q^k}$ with q^k elements, up to an efficiently computable isomorphism. All such fields can be viewed as the extensions of $\mathbb{Z}_q$, i.e. $\mathbb{Z}_q$ can be found as a subfield within $\mathbb{F}_{q^k}$.

For a field $\mathbb{F}_{q^k}$, we can consider the set $\mathbb{F}_{q^k}\backslash\{0\}$ as a group with the operation $\cdot$. This group is called the *multiplicative group of* $\mathbb{F}$, and it is a *cyclic* group of order $q^k - 1$. In other words, there exists an element $g \in \mathbb{F}_{q^k}$ with order $q^k - 1$ which generates the whole group.

Note that these groups do not usually have prime order as we wanted. This is because $q^k - 1$ can be a prime number only when $q = 2$ and k is prime. If $q > 2$, it must be odd, and thus $q^k - 1$ cannot be prime because it is even. If k is not prime, then $q^k - 1$ can be factored and we can prove $q^d - 1 | q^k - 1$ for all divisors $d|k$. Thus, the group $\mathbb{F}_{q^k}\backslash\{0\}$ is not quite what we want.

The idea is then to take a large subgroup $\mathbb{G} \subset \mathbb{F}_{q^k}\backslash\{0\}$ of prime order. For example, if both q and $\frac{q-1}{2}$ are prime,[5] then we can take $\mathbb{G}$ to be a subgroup of order $p = \frac{q-1}{2}$ of $\mathbb{Z}_q^* = \mathbb{Z}_q\backslash\{0\}$. This group then contains precisely the even powers of the generator g of $\mathbb{Z}_q^*$, i.e. it contains precisely the quadratic residues. As $\mathbb{G}$ has prime order, any of its non-trivial elements is a generator, so we can easily find its generator by finding an arbitrary non-trivial quadratic residue.

4 Application: El Gamal Public key encryption

The setup is the following: Alice would like to make it possible for anyone to send encrypted messages to her. To do this, she generates two keys, the public key pk and the secret key sk, and broadcasts pk to everyone. Then, Bob should be able to encrypt his message to Alice using her public key pk, while only Alice should be able to decrypt the ciphertext using her secret key sk. We formalize this interaction as follows:

Definition 5. *A public key encryption (PKE) scheme is a set of three PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *which satisfy the following:*

- *The algorithm* Gen *generates the secret and the public key:* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
- *The algorithm* Enc *takes the public key and a message* m*, and outputs a ciphertext* $c = \mathsf{Enc}(\mathsf{pk}, m)$.
- *The algorithm* Dec *takes the public key and a ciphertext* c*, and outputs a message* $m = \mathsf{Dec}(\mathsf{sk}, c)$.

Correctness requirement: $\mathbb{P}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m] = 1$ *for all messages* m.

[5] Such primes are called *safe*.

To define security, we consider the following game between the Challenger Ch and the adversary A. First Ch gets a bit $b \in \{0,1\}$, and generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and sends pk to A. Then A chooses two messages m_0, m_1 and sends them to Ch, who responds with $\mathsf{Enc}(pk, m_b)$. Then A outputs b'.[6] We will denote this game by $\mathsf{IND} - \mathsf{PubK}_b$.

Definition 6. *We say that* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a secure scheme if for all PPT adversaries A there exists a negligible function $\varepsilon = \varepsilon(\lambda)$ such that:*

$$\left|\mathbb{P}[1 \leftarrow \mathsf{IND} - \mathsf{PubK}_0(A, \lambda)] - \mathbb{P}[1 \leftarrow \mathsf{IND} - \mathsf{PubK}_1(A, \lambda)]\right| \leq \varepsilon$$

Now, we show how to construct such a scheme using cryptographic groups.

We will assume that we have a family of groups $\mathbb{G}(\lambda)$ of prime order p, where $p \sim \exp(\lambda)$, and we will assume that our message space is $\mathbb{G}$. Consider the following set of algorithms:

Gen: choose a random generator $g \leftarrow \mathbb{G}$, and a random element $a \leftarrow \mathbb{Z}_p$. Put $\mathsf{sk} = (g, a), \mathsf{pk} = (g, h = g^a)$. [7]

$\mathsf{Enc}((g, h), m)$: pick a random $r \leftarrow \mathbb{Z}_p$, compute the ciphertext $c = (g^r, h^r \cdot m)$.

$\mathsf{Dec}((g, a), (c_1, c_2))$: compute $c_2 \cdot c_1^{-a}$.

This scheme is correct as $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = \mathsf{Dec}(\mathsf{sk}, (g^r, h^r \cdot m)) = h^r \cdot m \cdot (g^r)^{-a} = g^{ar} \cdot m \cdot g^{-ar} = m$.

How do we prove security of such a scheme? The Discrete Logarithm assumption implies it is hard to compute $\mathsf{sk} = (g, a)$ from $\mathsf{pk} = (g, g^a)$, but it does not appear enough to fully justify the security of the scheme. Therefore, we have to introduce a new assumption:

Definition 7. *(DDH - Decisional Diffie-Hellman assumption) Let $\mathbb{G}$ be a group. Let D_0 be the distribution of the quadruples (g, g^a, g^b, g^{ab}) over the random choice of a generator $g \leftarrow \mathbb{G}$ and $a, b \leftarrow Z_p$, and let D_1 be the distribution of the quadruples (g, g^a, g^b, g^c) over the random choice of a generator $g \leftarrow \mathbb{G}$ and $a, b, c \leftarrow Z_p$. We say that $\mathbb{G}$ satisfies the DDH assumption the distributions D_0 and D_1 are computationally indistinguishable.*

It can be shown that DDH assumption is at least as strong as the Discrete Logarithm assumption, but it is not clear if the reverse implication holds. Thus, as we are not able to derive DDH from Discrete Logarithm, we must assume it holds.

Now, we are in the position to prove the security of the above scheme: the idea is, as usual, to use hybrids. Define the following hybrid distribution: let H_0 be the

[6] Note that there are no CPA queries in this game. This is because A can compute the encryption of any message m using his knowledge of the public key.

[7] Note that sk is hard to compute even if one knows pk. This is a necessary condition for security, but it is not sufficient.

distribution $(g, g^a, g^r, g^{ar}m_0)$, H_1 be $(g, g^a, g^r, g^c m_0)$, H_2 be $(g, g^a, g^r, g^c m_1)$ and H_3 be $(g, g^a, g^r, g^a r m_1)$, where $a, r, c \in \mathbb{Z}_p$ are random elements and $g \in \mathbb{G}$ is a generator. If we are able to show that every two pairs of adjacent hybrids are indistinguishable, then so are H_0 and H_3 as well.

Note that H_1 and H_2 are in fact the same distribution, for the values $g^c m_0$ and $g^c m_1$ both follow the uniform distribution in $\mathbb{G}$. To see why, put $s = g^{f(s)}$ for all elements $s \in \mathbb{Z}_p$ and observe that $\mathbb{P}[g^c m_0 = s] = \mathbb{P}[g^c g^{f(m_0)} = g^{f(s)}] = \mathbb{P}[c \equiv f(s) - f(m_0)(\mod p)] = \frac{1}{p}$, because c follows uniform distribution. As the same argument can be applied to $g^c m_1$, we see that H_1 and H_2 are indistinguishable even in the information-theoretic sense.

As for the hybrids H_0 and H_1, we can see that every adversary distinguishing between these two hybrids could be turned into a DDH adversary after dividing the last coordinate with m_0. Thus, if DDH assumption holds, hybrids H_0 and H_1 are computationally indistinguishable. As the similar argument works for H_2 and H_3, we see that H_0 and H_3 must indeed be computationally indistinguishable if DDH holds in $\mathbb{G}$. Thus, the security of the ElGamal scheme follows from DDH.

5 Elliptic curves

Definition 8. *An elliptic curve E over a field $\mathbb{F}$ is the set of solutions (x, y) to the equation $y^2 = x^3 + ax + b$.*[8] *The discriminant of an elliptic curve is $\Delta = 16(4a^3 - 27b^2)$.*[9]

To get some intuition, we will assume $\mathbb{F} = \mathbb{R}$ in the beginning.

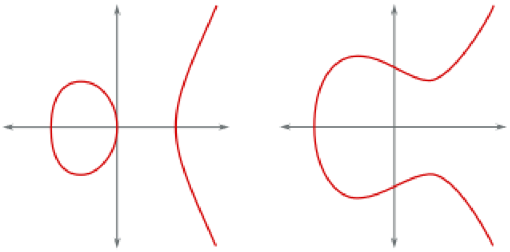

Figure 1: Graphs of elliptic curves over the real numbers

[8] This form of the equation is called the Weierstrass form. Although elliptic curves may be defined more generally, using an equation that is quadratic in y and cubic in x, every such curve can be transformed into the Weierstrass form if $\operatorname{char} \mathbb{F} \neq 2, 3$.

[9] This quantity is useful because it characterizes the behavior of the elliptic curve, such as whether it has 1 or 3 roots over the real number etc.

Note that the graph of an elliptic curve is symmetric around the x-axis. This is because only y^2 appears in the equation, and thus if $(x, y) \in E$, we also have $(x, -y) \in E$. Similarly, for very negative X, we will have $x^3 + ax + b < 0$ and thus there will be no solutions. For positive x, one has $|y| = \sqrt{x^3 + ax + b} \sim x^{3/2}$, and thus the graph increases to the infinity super-linearly.

Using an elliptic curve E, we can define a group $\mathbb{G}$ whose elements are the points of E and the point at infinity ∞ (which we can think of as having coordinates $(0, \infty)$, i.e. being all the way at the top of the y-axis). The operation in this group, which we call addition, is defined the following way: for any two points P, Q we draw a line through PQ and intersect it with the elliptic curve to get a third point. Then, $P + Q$ is defined to be the reflection of this new intersection point over the x-axis. This process is depicted in the following diagram:

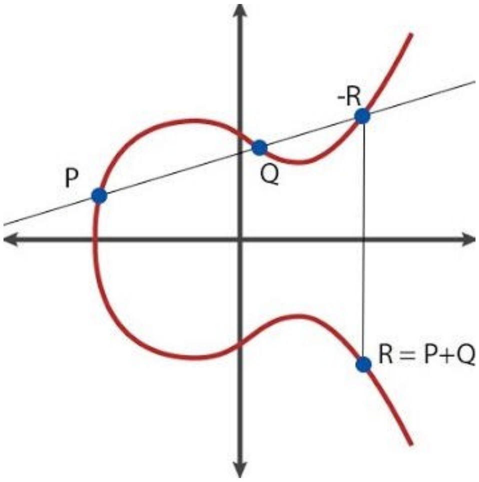

Figure 2: Group law on an elliptic curve

The case depicted in the picture is only the generic case and many edge cases could appear: P could be ∞, it could be that $P = Q$ and that there is no well-defined line through these two points, etc. In these cases, one could make special definitions, where a line through ∞ and P would just be the vertical line through P, a line through two identical points would correspond to the tangent, etc. However, we will derive a formula from which we will be able to characterize the addition law without the need for special cases.

It is instructive to try to compute the coordinates of the sum of P and Q, in order to check that this process is really well-defined (i.e. that the line PQ intersects E for the third time etc.). In order to do this, let $P = (x_P, y_P), Q = (x_Q, y_Q)$. The line through P, Q can be written as $y = rx + s$, where $r = \frac{y_Q - y_P}{x_Q - x_P}$ and $s = \frac{x_Q y_P - x_P y_Q}{x_Q - x_P}$. Thus, the intersection of this line with E can be found from

$$x^3 + ax + b = y^2 = (rx + s)^2$$

Rearranging gives $x^3 - r^2x + (a - 2rs)x + (b - r^2) = 0$. Note that this equation has roots x_P, x_Q, x_R, and therefore it can be factored as

$$x^3 - r^2x + (a - 2rs)x + (b - r^2) = (x - x_P)(x - x_Q)(x - x_R).$$

By opening the brackets on the right hand side and comparing the coefficients next to x^2, we find that

$$x_P + x_Q + x_R = r^2 = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)^2.$$

Thus, we can express $x_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)^2 - x_P - x_Q$. As mentioned above, this formula can be used to define addition for any two points on the curve.

6 Next time

Next time, we will consider the elliptic curves over the finite fields, which are more useful in cryptography.