Public-key cryptography based on discrete log and LWE

The DL, DH and DDH problems, and how they relate. The El Gamal cryptosystem and proof that it is secure if DDH is hard. Then some example of groups we can use, can be a subgroup of Z_p*, or you can talk about elliptic curves. You can also put less emphasis on El Gamal, for instance skip the example groups and go to LWE instead, define the problem and the cryptosystem and do the proof from the exercise that decryption works under a certain assumption about the noise distribution.

The "Elevator Pitch" (Synthesis)

> **Explain Public-key Crypto based on Discrete Log to a peer in 3 sentences. Focus on WHY we use it, not just how.**

Core Vocabulary & Syntax (Total Recall)

Discrete Log (DL) problem

> Given a group , generator , and element , find integer , such that .

> This is the "reverse" of modular exponentiation. While computing is efficient even for large , finding from seems to be computationally infeasible when is a sufficiently large prime (at least 2048 bits). This asymmetry - easy to compute forward, hard to reverse - makes it a one-way function suitable for cryptography.

> The DL problem is in many groups notoriously hard, for instance in .

Diffie-Hellman (DH) problem

> Given a group , generator , and , where are randomly and independently chosen from , compute .

> Notice that you're **not** asked to find or —just the shared secret . However, if we could find from , we could solve DH by a single exponentiation, so *The DH problem is no harder than the DL problem.*

Decisional Diffie-Hellman (DDH) problem

> The DH problem has a peculiar property, namely if I give you a group element and I claim it solves a DH instance, it is not clear that you can verify that the solution is correct, at least not unless you can solve DL. You would need to decide if, for given , it holds that . This seems to require that you solve DL (although it is NOT clear that this would be necessary).

> The idea behind the DDH is that you get an instance of the DH problem, plus an extra group element which is either a correct DH solution, or is a random element. You are then supposed to guess which case you are in.

> Formally:
> Given a group , generator , and , where are randomly and independently chosen from ; and where is chosen either as , or uniformly random from . Now guess which of the two cases we are in.

> Clearly, if you could solve DH, then you could solve DDH, by computing and comparing this to . Therefore, DDH is no harder than DH

Diffie-Hellman key exchange protocol (DHE)

> In 1977, Diffie and Hellman suggested to use the DH problem as a basis for establishing a shared secret two parties and that share no secret key in advance. The method for this is very simple, assuming that we already agreed (in public) on a group and a generator
>
> 1. chooses at random in , chooses at random in .
> 2. sends to , sends to .
> 3. computes and computes

> and compute the same value in the last step, since . An adversary observing the communication would need to solve an instance of the DH problem to find the shared value.

> This (or its elliptic curve version) is still often used: in TLS 1.3 handshakes, the use of (EC)DHE is mandatory, where a server and a client use it to agree on the key they would use for their further AES or

> Later on El Gamal suggested a way to turn this idea into a regular public key cryptosystem. What we do is essentially to consider 's first message in the above as a part of his public key, then 's part of the protocol can be modified to be an encryption:

El Gamal cryptosystem and proof it is secure (provided DDH is)

> To turn the DH key exchange protocol into a public-key crypto system, what we do is essentially to consider 's first message as a part of his public key, then 's part of the protocol can be modified to be an encryption.
>
> **Key generation** On group and generator choose at random from . Then the public key is the specification of and , while the secret key is . The plaintext space is while the ciphertext space is .
> **Encryption** To encrypt , we choose at random from , and the ciphertext is .
> **Decryption** To decrypt ciphertext , compute .

> To see that decryption works, simply plug in for in the decryption algorithm:

> The problem of decrypting an El Gamal ciphertext (without the secret key) is equivalent to solving the DH problem.

> If the DDH problem is hard, then the El Gamal cryptosystem is CPA secure.

LWE. LWE based cryptosystem and proof decryption works (under certain noise distribution assumption)

> The LWE problem is defined over a finite field where is a prime, and all computations in the following will be modulo , even if we do not say so explicitly. A secret vector → will be involved.

> The problem is: You are given → (columns of mx) and → → where the 's are random but numerically “small”. The goal is now to find →.

> The 's being random and numerically "small" can mean different things, but always means that they are much smaller than . For instance, can be chosen uniformly in the interval $\sqrt{}$ $\sqrt{}$. Or it can be chosen using a discrete Gaussian distribution with mean 0 and standard deviation much smaller than . It turns out that the exact details of this are not very important as to how hard the problem is.
> It is fine if we simply assume that the 's are chosen with a distribution with the following important property: even if you sample values distributed according to , take numeric value and sum them, the result will be smaller than with overwhelming probability.

> The motivation for the name of the problem is that the adversary gets some partial information on →and tries to learn it from this information, but the problem is non-trivial because the "errors" are added in. Of course, without the errors the problem would be easy once since then one just solves the linear equations to get →.

> There is also a decision version of LWE (think DDH for DH): you are given random → and then either → → or where the are uniformly random. The goal is now to decide which case you are in.

> Decision LWE (DLWE) is hard if any probabilistic polynomial-time (PPT) algorithm can distinguish the two cases with only negligible advantage in (where is polynomial in).

> To make a secret-key cryptosystem from LWE:
> **Key Generation** Secret key: random vector → . Public key: , where → → → , where → are uniformly random and the are chosen according to .
> **Encryption** Message is a bit . Choose random bits and then the ciphertext is
>
> **Decryption** To decrypt , compute → , and output 0 if this value is closer to 0 than it is to . Output 1 otherwise.

Correctness of LWE Decryption

Any given looks like → → → . We can also think of these as rows to a matrix and a vector describing a system of linear equations.

(Since , the sum is really just selecting a subset of the public key components and adding them together. Knowing that all 's solve to , all items of this subset and their sum will also solve to .)

The full cyphertext is:

→ → → → —

When decrypting, we compute:

→ →

→ → — → →

→ → → → —

—

We know that – , and because

| | —

For we expect to be closer to than to –, meaning –. Indeed, we have:

| — | — —

For we expect to be closer to – than to , meaning – –. We have:

— | — —| — —

Logic Flow / Mechanism (Process)

El Gamal Key Generation, Encryption, Decryption

1. Run **GGen**(k) to get specification of group **G** and generator **α**.
2. [_____________________]
3. [_____________________]
4. Public key: specification of **G** and **β**; secret key: **a**.

El Gamal Encryption of (m \in G)

1. Choose **r** at random from (\Z_t).
2. [____________________]
3. Ciphertext: ((\alpha^r, \beta^r m)).

El Gamal Decryption of ((c, d))

1. [____________________]
2. Output: (m).

Reduction: CPA Security of El Gamal (Theorem 9.7)

1. Assume adversary **Adv** breaks CPA security of El Gamal with advantage (\epsilon).
2. [____________________]
3. [____________________]
4. **B** solves DDH with advantage at least (\epsilon), contradiction if DDH hard.

The "Exam Trap" (Distinctions)

Distinguish between DL, DH, and DDH problems based on input, output, and hardness implications. Complete the matrix:

Problem	Input	Output/Task	Implied by DL?	Implies DDH?
DL	(G, \alpha, \beta)	[Fill in]	-	[Fill in]
DH	(G, \alpha, \alpha^a, \alpha^b)	[Fill in]	Yes (Lemma 9.1)	[Fill in]
DDH	(G, \alpha, \alpha^a, \alpha^b, \alpha^c)	[Fill in]	Yes	No (Lemma 9.2)

*Why is DDH not hard in (\Z_p^) (Lemma 9.10)? Specify the attack using parities.**

Exam Simulation

Oral Exam Question 1: "Present the El Gamal cryptosystem: specify key generation, encryption, decryption algorithms, and verify decryption works by direct computation."

Oral Exam Question 2: "Prove Theorem 9.7: If DDH is hard w.r.t. GGen, then El Gamal is CPA secure. Construct the reduction algorithm B using adversary Adv as a subroutine."

Oral Exam Question 3: "Explain why (\Z_p^*) cannot yield CPA-secure El Gamal (Lemma 9.10). Then describe prime order subgroups of (\Z_p^*) for safe primes and the encoding from (\Z_q) to G."

Source Map

- CryptographyV6.pdf | Page 115-117 | Covers: DL, DH, DDH problems and definitions
- CryptographyV6.pdf | Page 117 | Covers: El Gamal cryptosystem (general version), decryption verification (Lemma 9.6)
- CryptographyV6.pdf | Page 116 | Covers: Theorem 9.7 (CPA security if DDH hard), Exercise 9.1 (proof sketch)
- CryptographyV6.pdf | Page 120 | Covers: Why DDH not hard in (\Z_p^*) (Lemma 9.10), safe primes, prime order subgroups
- **Source data missing for LWE problem and cryptosystem** (no details in provided excerpts)