Introduction to Cryptography
Part III: DLP, DH and ElGamal

André Schrottenloher

Inria Rennes
Team CAPSULE

[0]

The DL Problem

The DL Problem

Discrete Logarithm

Let G, $\cdot$ be a multiplicative group of order q and g a known element. Given g^a (where $a \hookleftarrow U(\mathbb{Z}_q)$), find a.

- $a \to g^a$ is **always easy**
- $g^a \to a$ is **sometimes hard**, but **not always**

Example: take N, k prime with N, a subgroup of $(\mathbb{Z}_N, +)$ generated by k.

- We can compute multiplicative inverses
- $ka \mod N \to a \mod N$ is easy

Safe primes

Remark: G in the DL problem can always be replaced by a cyclic group (generated by g).

Historical choice for DL groups:

- Work in the multiplicative group $\mathbb{Z}_p^*$, where p is prime
- Choose a subgroup of $\mathbb{Z}_p^*$ with large prime order
- Take g a generator of this group

- A **safe prime** p is such that $(p-1)/2$ is prime.
- This guarantees the existence of a large subgroup, in which we work.

$(p-1)/2$ is called a Sophie Germain prime.

Interlude: Pohlig-Hellman reduction

Reduce the DLP in a group of order $n = p_1 p_2$ to the DLP in groups of order p_1 and p_2 (if p_1, p_2 are coprime).

Algorithm:

- Let $h = g^a$
- Compute DL of $h^{p_2} = (g^{p_2})^a$ in the subgroup generated by g^{p_2} (of order p_1)

$\Longrightarrow$ get $a \mod p_1$

- Compute DL of $h^{p_1} = (g^{p_1})^a$ in the subgroup generated by g^{p_1} (of order p_2)

$\Longrightarrow$ get $a \mod p_2$

- Compute a using the CRT (since p_1, p_2 are coprime).

$\Longrightarrow$ we want to work in a group of **large prime order**.

Interlude: Pohlig-Hellman for a prime power

If $e \geq 2$, reduce the DLP in a group of order $n = p^e$ to e instances of DLP in a group of size p.

Algorithm (for $h = g^a$):

1. Initialize $x_0 = 0$
2. Compute $\gamma = g^{p^{e-1}}$ which has order p
3. For all $k = 0, \ldots, e-1$ do:
   - Compute the DL d_k of $h_k = (g^{-x_k} h)^{p^{e-1-k}}$ in the group $\langle\gamma\rangle$ generated by γ
   - Set $x_{k+1} = x_k + p^k d_k$

Then x_e is the DL. Indeed:

$$\gamma^{d_{e-1}} = (g^{-x_{e-1}} h) \implies h = g^{x_{e-1}} \gamma^{d_e} = g^{x_{e-1} + p^{e-1} d_{e-1}} \quad .$$

The non-trivial part is to prove that $h_k \in \langle\gamma\rangle$, which we have to prove by induction over k.

Interlude: DL in $\mathbb{Z}_p^*$ vs. elliptic curves

- The DL in $\mathbb{Z}_p^*$ can be solved in **subexponential time** using index calculus / sieving methods (similarly to factoring).
- p has to be large (2048-4096 bits) to ensure security.

Nowadays, we don't use DL in $\mathbb{Z}_p^*$ anymore, but groups of **points on elliptic curves.**

An elliptic curve (on $\mathbb{Z}_p$) is the set of points (x, y) defined by an equation of the form $y^2 = x^3 + ax + b$ (+ a 'point at infinity"). It can be equipped with an additive group law.

- When the elliptic curve is well-chosen, the DL is **hard**.
- The best known algorithms are **exponential** (this lecture + TD).

Solving the DLP

Solving the DLP

- The DLP can be solved in any group of order q in time $\mathcal{O}\left(\sqrt{q}\right)$.
- This is the best complexity known that works **for any group.**

An algorithm

Suppose $h = g^a$ and g are given.

1. Compute h^i for many random integers i
2. Compute g^j for many random integers j
3. Look for a pair (i, j) such that $i \neq j$ and $h^i = g^j$

From such a pair: $g^{ai} = g^j \implies ai = j \mod q \implies a = ji^{-1} \mod q$ (problem solved).

Next: compute the complexity of this approach.

Interlude: birthday paradox

What is the probability of two students (among 20) having the same birthday?

$$1 - (1)(1 - 1/365)(1 - 2/365) \cdots (1 - 19/365) \simeq 0.41 \quad .$$

Lemma

Let $y_1, \ldots, y_\ell$ be random (uniform) samples in a set of size N. A **collision** is a pair (y_i, y_j) such that $y_i = y_j$ **and** $i \neq j$. There exists a collision:

- With prob. at most $\ell^2/2N$
- With prob. at least $\frac{\ell(\ell-1)}{4N}$ if $\ell \leq \sqrt{2N}$

Intuition:

- Each pair has probability $1/N$ of forming a collision
- There are $\ell^2/2$ pairs $\implies$ this gives the upper bound
- But they are not independent

The constant is not that important. It can be made more precise.

Interlude: birthday paradox (ctd.)

Write $NoColl_i$ the event "no collision among $y_1, \ldots, y_i$."

$\Pr[NoColl_\ell] = \Pr[NoColl_1] \cdot \Pr[NoColl_2 | NoColl_1] \cdots \Pr[NoColl_\ell | NoColl_{\ell-1}]$.

Also: $\Pr[NoColl_1] = 1$, and $\Pr[NoColl_{i+1} | NoColl_i] = 1 - i/N$ (the new element must be different from the i previous ones)

$$\implies \Pr[NoColl_\ell] = \prod_{i=1}^{\ell-1} (1 - i/N)$$

Now we do some bounding: $\forall i, 1 - i/N \leq e^{-i/N}$:

$$\Pr[NoColl_\ell] \leq e^{-\sum_{i=1}^{\ell-1} i/N} = e^{-\ell(\ell-1)/2N} \quad .$$

And for $x < 1$, $1 - x/2 \geq e^{-x}$:

$$\Pr[Coll] = 1 - \Pr[NoColl_\ell] \geq 1 - e^{-\ell(\ell-1)/2N} \geq \frac{\ell(\ell-1)}{4N} \quad .$$

Conclusion

Powers of h and g give us random elements of the group (heuristically).
A collision occurs after computing $\mathcal{O}(\sqrt{q})$ powers. This algorithm has:

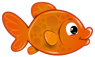
- Time $\widetilde{\mathcal{O}}(\sqrt{q})$ (optimal, up to small factors)
- Memory $\mathcal{O}(\sqrt{q})$ (not optimal)

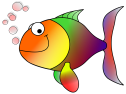
We can do better: $\mathcal{O}(\sqrt{q})$ time and $\mathcal{O}(1)$ memory (see TD).

Diffie-Hellman Key Exchange

The Diffie-Hellman key-exchange

Public parameters: a cyclic group $\mathbb{G}$ and a generator g of order q.





1.	**Alice** chooses $\mathsf{a} \in \{1, \ldots, q-1\}$		**Bob** chooses $\mathsf{b} \in \{1, \ldots, q-1\}$
2.	**Alice** sends g^{a}	$\rightarrow$	
3.		$\leftarrow$	**Bob** sends g^{b}
4.	**Alice** computes $(g^{\mathsf{b}})^{\mathsf{a}}$		**Bob** computes $(g^{\mathsf{a}})^{\mathsf{b}}$

$\mathsf{k} = (g^{\mathsf{b}})^{\mathsf{a}} = (g^{\mathsf{a}})^{\mathsf{b}}$ is the **shared secret key**.

Do not use this in practice.

DH security

- The adversary observes only g^a, g^b where $a, b \hookleftarrow U(\{1, \ldots, q-1\})$.
- Recovering g^{ab} = the **computational DH problem** (CDH)

Many security proofs are based instead on the **decisional** DH problem (DDH).

Distinguish the two cases:

- RAND: a distribution g^a, g^b, g^c where $a, b, c \hookleftarrow U(\{1, \ldots, q-1\})$
- DDH: a distribution g^a, g^b, g^{ab} where $a, b \hookleftarrow U(\{1, \ldots, q-1\})$

DDH is difficult in G is no PPT adversary $\mathcal{A}$ can exhibit non-negligible advantage:

$$\mathrm{Adv}(\mathcal{A}) = \left| \Pr\left[\mathcal{A} \xrightarrow{RAND} 1\right] - \Pr\left[\mathcal{A} \xrightarrow{DDH} 1\right] \right| .$$

The complete DDH game

The DDH game is played between a **challenger** $\mathcal{C}$ and an **adversary** $\mathcal{A}$.

- $\mathcal{C}$ chooses (G, g)
- $\mathcal{C}$ chooses $x, y \hookleftarrow U(\mathbb{Z}_q)$ and b
- RAND case $(b = 0)$: $z \hookleftarrow U(\mathbb{Z}_q)$; DDH case $(b = 1)$: $z = xy$
- $\mathcal{C}$ sends (g, g^x, g^y, g^z) to $\mathcal{A}$
- $\mathcal{A}$ returns a bit b'
- If $b = b'$, $\mathcal{A}$ wins

DDH is difficult in G is for any PPT adversary $\mathcal{A}$:

$$\mathrm{Adv}(\mathcal{A}) = \left| \Pr\left[\mathcal{A} \text{ wins}\right] - \frac{1}{2} \right| = \mathrm{negl}(n) \quad .$$

DH security (ctd.)

$$\text{DLP} > \text{CDH} > \text{DDH}$$

- If we can solve DLP we can solve CDH
- If we can solve CDH we can solve DDH

Not an equivalence: there are "gap" groups where CDH is hard and DDH is easy.

The ElGamal PKE

ElGamal PKE

We are now in a group G where DDH is hard.

We are constructing a public-key encryption scheme based on this.

ElGamal PKE

Public parameters $(\mathbb{G}, q, g)$ (q is the order of $\mathbb{G}$, g a generator)

KeyGen:

- Sample $x \hookleftarrow U(\mathbb{Z}_q)$
- $\mathsf{sk}, \mathsf{pk} = x, g^x := h$

Enc $\mathsf{m} \in \mathbb{G}$

- Sample $y \hookleftarrow U(\mathbb{Z}_q)$
- Return $c_1, c_2 := (g^y, h^y \cdot m)$

Dec $c = (c_1, c_2)$

- Return $m = c_2({c_1}^{-x})$

Correctness.

$$c_2({c_1}^{-x}) = h^y m g^{-xy} = g^{xy} m g^{-xy} = m \ .$$

ElGamal security

Lemma
If DDH is difficult in G, then ElGamal is IND-CPA.

The proof is a **reduction**: given $\mathcal{A}$ that breaks IND-CPA security of ElGamal, construct $\mathcal{A}'$ that breaks DDH.

We say that the IND-CPA security of ElGamal **reduces to** DDH.

Proof

Consider an adversary $\mathcal{A}$ playing the IND-CPA game for ElGamal:

- Initialization: the challenger chooses a key (x, g^x), a bit b, and sends g^x to $\mathcal{A}$
- $\mathcal{A}$ chooses m_0, m_1 and sends them to $\mathcal{C}$
- $\mathcal{C}$ computes $c_1, c_2 = \mathsf{Enc}(\mathsf{pk}, m_b)$ and sends (c_1, c_2) to $\mathcal{A}$
- $\mathcal{A}$ computes b', wins if $b' = b$

We show that if DDH is difficult:

$$\mathrm{Adv}^{CPA}(\mathcal{A}) = |\Pr[\mathcal{A} \text{ Win}] - 1/2| \leq \mathrm{negl}(n)$$

For this we use $\mathcal{A}$ to define an adversary $\mathcal{B}$ against DDH.

Internally, $\mathcal{B}$ will run $\mathcal{A}$. When running inside $\mathcal{B}$, $\mathcal{A}$ still believes that they are in the IND-CPA game: all messages sent and received match those of the game.

Proof (ctd.)

Here is our adversary $\mathcal{B}$ playing the DDH game:

- (G, q, g) is fixed
- $\mathcal{C}$ chooses $x, y \hookleftarrow U(\mathbb{Z}_q)$ and b
- RAND case $(b = 0)$: $z \hookleftarrow U(\mathbb{Z}_q)$; DDH case $(b = 1)$: $z = xy$
- $\mathcal{C}$ sends (g^x, g^y, g^z) to $\mathcal{B}$

- $\mathcal{B}$ sends g, g^x to $\mathcal{A}$
- $\mathcal{A}$ chooses m_0, m_1 and sends them to $\mathcal{B}$
- $\mathcal{B}$ chooses b', computes $(g^y, g^z \cdot m_{b'})$ and sends it to $\mathcal{A}$
- $\mathcal{A}$ returns a bit b'' to $\mathcal{B}$

- If $b' = b''$ ($\mathcal{A}$ wins in their game), $\mathcal{B}$ returns 1, else 0

See next slide.

Proof (ctd.)

Here is all the activity between $\mathcal{C}$, $\mathcal{B}$ and $\mathcal{A}$. Notice that **all that $\mathcal{A}$ ever sees is an IND-CPA game** where $\mathcal{B}$ acts as the challenger.

$\mathcal{C}$		$\mathcal{B}$		$\mathcal{A}$
$b \hookleftarrow U(\{0,1\})$ If $b = 0$, $z \hookleftarrow U(\mathbb{Z}_q)$ else $z = xy$				
	$\xrightarrow{g^x, g^y, g^z}$		$\xrightarrow{g, g^x}$	
		Choose b'		Choose m_0, m_1
			$\xleftarrow{m_0, m_1}$	
		$c_1, c_2 := (g^y, g^z \cdot m_{b'})$		
			$\xrightarrow{c_1, c_2}$	
				Compute b''
			$\xleftarrow{b'}$	
		If $b' = b''$ ($\mathcal{A}$ wins) output 1 else 0		

Proof (ctd.)

We study $\mathcal{B}$.

In the RAND case: $(b = 0)$

- z is uniform and independent, so $c_2 = g^z m_b$ is uniform and independent
- $\mathcal{A}$ cannot distinguish the ciphertexts
- $\mathcal{B}$ returns 1 with probability $1/2$
- $\Pr[\mathcal{B} \text{ wins} | RAND] = 1/2$

In the DDH case: $(b = 1)$

- $z = xy$ and the ciphertext is valid
- $\mathcal{B}$ returns 1 iff $\mathcal{A}$ wins

$$\Pr[\mathcal{B} \text{ wins} | DDH] = \Pr[\mathcal{A} \text{ wins}]$$

In total:

$$|\Pr[\mathcal{B} \text{ wins}] - \frac{1}{2}| =$$

$$|\frac{1}{2}\Pr[\mathcal{B} \text{ wins} | DDH] + \frac{1}{2}\Pr[\mathcal{B} \text{ wins} | RAND] - \frac{1}{2}| = \frac{1}{2}|\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}| \ .$$

Proof (end)

For any adversary $\mathcal{A}$ against IND-CPA, there exists an adversary $\mathcal{B}$ against DDH that:

- Takes the same time to run as $\mathcal{A}$
- Satisfies:

$$\left|\Pr\left[\mathcal{B} \text{ wins}\right] - \frac{1}{2}\right| = \frac{1}{2}\left|\Pr\left[\mathcal{A} \text{ wins}\right] - \frac{1}{2}\right|$$

If DDH is difficult, for any PPT adversary $\mathcal{B}$ against DDH, $\left|\Pr\left[\mathcal{B} \text{ wins}\right] - \frac{1}{2}\right| = \mathrm{negl}(n)$

$$\Downarrow$$

For any PPT adversary $\mathcal{A}$ against ElGamal, $\left|\Pr\left[\mathcal{A} \text{ wins}\right] - \frac{1}{2}\right| = \mathrm{negl}(n)$

If DDH is difficult, then ElGamal is secure in the group G.

Discussion

One of the advantages of ElGamal compared to RSA:

The group is fixed. Multiple users can work in the same group (vs. need to regenerate $N = PQ$).

In crypto standards (e.g. NIST SP 800-186 for elliptic curves), there is a specification of groups that you can use.

One of the disadvantages of ElGamal & RSA:

It's not post-quantum :(