# Diffie-Hellman

## Notes for *Serious Cryptography* chapter 11

Jeffrey Goldberg
jeffrey@goldmark.org

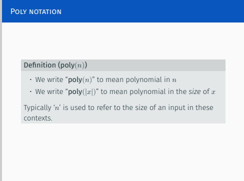
March 4, 2022

Last modified: May 8, 2024

## Definition (poly$(n)$)

- We write "**poly**$(n)$" to mean polynomial in $n$
- We write "**poly**$(|x|)$" to mean polynomial in the *size* of $x$

Typically '$n$' is used to refer to the size of an input in these contexts.




1. **poly**$(|x|)$ is (almost always) the same as **poly**$(\log x)$.
2. In much earlier sessions we talked about indistinguishable from random. "Perfect" meant tha there was no algorithm which could do the thing, while "cryptographic" or "assymtotic" meant there was no **poly**$(n)$ algorithm that could do the thing.

# DLP

When $p$ is appropriately chosen, and $g$ is a generator for $\mathbb{Z}_p^\times$, there is a **poly**$(|p|)$ algorithm to compute $A$ in (1)

$$A = g^a \pmod{p} \tag{1}$$

but there is no **poly**$(|p|)$ algorithm to compute $a$ in (2).

$$a = \log_g A \pmod{p} \tag{2}$$

We are going to turn the DLP into useful cryptography

# All in $\mathbb{Z}_p^\times$

- Unless otherwise stated, all of the math that follows is within the abelian finite cyclic group $\mathbb{Z}_p^\times$ in which $g$ is a generator.
- The group parameters, $p$ and $g$, are *not* secret.

# Diffie-Hellman Key Exchange (DHKE)

# Alice picks a secret

Alice picks a secret, little $a$, and generates a public big $A$.

$$A = g^a \tag{3}$$

Bob does similarly

$$B = g^b \tag{4}$$

# Never sending secrets

- Alice sends $A$ to Bob.
- Alice never transmits $a$.
- Bob sends $B$ to Alice.
- Bob never transmits $b$.

# Computing keys

Alice knows $a$ and $B$. She computes

$$\mathsf{k}_A = B^a \tag{5}$$

Bob knows $b$ and $A$. He computes

$$\mathsf{k}_B = A^b \tag{6}$$

# Deconstructing $\mathsf{k}_{A,B}$

$$\begin{aligned} \mathsf{k}_A &= B^a \\ &= \left(g^b\right)^a \\ &= g^{ab} \end{aligned} \tag{7}$$

$$\begin{aligned} \mathsf{k}_B &= A^b \\ &= \left(g^a\right)^b \\ &= g^{ba} \end{aligned} \tag{8}$$

# A common key

$$\mathsf{k}_A = g^{ba} = g^{ab} = \mathsf{k}_B \tag{9}$$

# A small example, $\mathsf{k}_A$

With

$$p = 59;\ g = 2;\ a = 20;\ A = g^a = 4;\ b = 9;\ B = g^b = 40 \quad (10)$$

$$\begin{aligned} \mathsf{k}_A = B^a &= 40^{20} &= 5 \\ = \left(g^b\right)^a &= \left(2^9\right)^{20} &= 5 \\ = g^{ab} &= 2^{9\cdot 20} = 2^{180} &= 5 \end{aligned}$$

## A small example, $\mathsf{k}_B$

Again with

$$p = 59;\ g = 2;\ a = 20;\ A = g^a = 4;\ b = 9;\ B = g^b = 40 \tag{10}$$

$$\begin{aligned} \mathsf{k}_B = A^b && = 46^9 && = 5 \\ = (g^a)^b && = \left(2^{20}\right)^9 && = 5 \\ = g^{ba} && = 2^{20\cdot 9} = 2^{180} && = 5 \end{aligned}$$

DHKE with $g$ a generator of $\mathbb{Z}_p^\times$

| Alice | | Bob |
|---|---|---|
| $a \leftarrow\$ \mathbb{Z}_p^\times$ | | $b \leftarrow\$ \mathbb{Z}_p^\times$ |
| $A \leftarrow g^a$ | | $B \leftarrow g^b$ |
| | $\xrightarrow{A}$ | |
| | $\xleftarrow{B}$ | |
| $\mathsf{k}_A \leftarrow B^a$ | | $\mathsf{k}_B \leftarrow A^b$ |

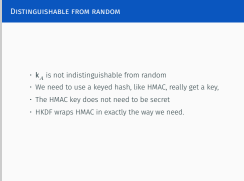
**Figure 1:** Example protocol diagram

# Diffie-Hellman Key Exchange (DHKE)

## Just a toy

# Distinguishable from random

- $\mathsf{k}_A$ is not indistinguishable from random
- We need to use a keyed hash, like HMAC, really get a key,
- The HMAC key does not need to be secret
- HKDF wraps HMAC in exactly the way we need.




1. $g^{ab} < p$. Unless $p$ is a power of 2 (it isn't) there will be bit sequences that can't ber $g^{ab}$.
2. Other keyed hashes could be used. BLAKE3 is an obvious candidate.
3. One might think that a small distinguishability in the leading bit doesn't matter. And maybe it doesn't, but other security proofs depend in indistinguishability.

# Open to manipulation

- DHKE works against a passive attacker who can observe the exchange
- DHKE does not work if attacker can interfere with communication
- DHKE needs a mutually authenticated channel with data integrity

# A large subgroup

- Solving a discrete logarithm in $\mathbb{Z}_p^\times$ can be broken down into solving the problem for all of the subgroups of $\mathbb{Z}_p^\times$.
- Picking a **safe prime** $p$ ensures that there will be a large subgroup of size $(p-1)/2$.

A large subgroup

- Solving a discrete logarithm in $\mathbb{Z}_p^\times$ can be broken down into solving the problem for all of the subgroups of $\mathbb{Z}_p^\times$.
- Picking a **safe prime** $p$ ensures that there will be a large subgroup of size $(p-1)/2$.

1. With $p, q$ both prime and $p = 2q + 1$ the term for $q$ is a Sophie Germain prime.
2. Germain proved Fermat's *Last* Theorem held for primes of this sort.
3. This is the only relevance of Fermat's *Last* theorem to what we do. Later, we will talk about Fermat's *Little* Theorem.

# Computing decision problems

# Two more problems

**Definition (CDH)**

Computing $g^{ab}$ given only $p, g, g^a, g^b$ is known as the "Computational Diffie-Hellman" problem.

**Definition (DDH)**

Distinguishing between $g^{ab}$ and $g^r$ for some random $r$ given only $p, g, g^a, g^b$ is known as the "Decisional Diffie-Hellman" problem.

**Definition (CDH)**
Computing $g^{ab}$ given only $p, g, g^a, g^b$ is known as the "Computational Diffie-Hellman" problem.

**Definition (DDH)**
Distinguishing between $g^{ab}$ and $g^r$ for some random $r$ given only $p, g, g^a, g^b$ is known as the "Decisional Diffie-Hellman" problem.

1. The DDH does *not* hold for $\mathbb{Z}_p^\times$
2. Given $g^x$ it is easy to compute whether $x$ is odd or even. And so given $g^a$ and $g^b$ can know whether $ab$ is odd or even. This gives us a 0.75 probability of determining whether we got $g^{ab}$ or $g^r$.
3. There are ways to tinker with the group to avoid this.

# Relative hardness

- The DLP is *at least* as hard as the CDH problem.
- The CDH problem is *at least* as hard DDH.
- This means that the DDH is the *strongest condition*.

# Relative conditions

- Something that depends on the hardness of the DLP does not necessarily depend on the hardness of the CDH.
- Something that depends on the hardness of the CDH also depends on the hardness of the DLP, but might not depend on the hardness of the DDH.
- Something that depends on the hardness of the DDH also depends on the hardness of the CDH and DLP.
- This means that the DDH is the *strongest condition*.

# Problems and assumptions

- The DLP *assumption* is that the DLP is hard.
- The CDH *assumption* is that the CDH problem is hard.
- The DDH *assumption* is that the DDH problem is hard.

# Assume the least

We prefer cryptographic systems that rely on the weakest assumptions.

**Example**

Imagine two cryptographic schemes $\alpha$ and $\beta$ which differ only in that $\alpha$'s security relies on the DDH while $\beta$'s does not, we should prefer $\beta$.

# ElGamal Public Key Encryption

# ElGamal Encryption

1. Alice picks secret $a$ and publishes $A = g^a$
2. Bob picks an ephemeral secret $b$ and computes a shared secret $s = A^b$.
3. Bob computes $B = g^b$.
4. To encrypt message $m$ Bob computes $c = m \cdot s$.
5. Bob sends $B$ and $c$ to Alice.

# Decryption

1. Alice computes $s = B^a$
2. Alice computes $s^{-1}$ (There is a fast way to do this)
3. Alice computes $m = c \cdot s^{-1}$

## Decryption

1. Alice computes $s = B^a$
2. Alice computes $s^{-1}$ (There is a fast way to do this)
3. Alice computes $m = c \cdot s^{-1}$

---

1. This is what I get for teaching DH before RSA. I haven’t taught how to compute modular inverses.

# Bob encrypts $m = 19$ to Alice

$$p = 23;\ g = 5;\ a = 17;\ A = g^a = 15;\ b = 10;\ m = 19 \tag{11}$$

$$\begin{aligned} s = A^b &= 15^{10} &= 3 \\ B = g^b &= 5^{10} &= 9 \\ c = m \cdot s &= 19 \cdot 3 &= 11 \end{aligned}$$

# Alice decrypts

$$p = 23;\ g = 5;\ a = 17;\ A = g^a = 15;\ b = 10;\ m = 19 \qquad (11)$$

$$\begin{aligned} s = B^a &= 9^{17} &= 3 \\ s^{-1} = & = 3^{-1} &= 8 \\ m = c \cdot s^{-1} &= 11 \cdot 8 &= 19 \end{aligned}$$