

Discrete logarithm problem I

Hardness assumptions and usage

Tanja Lange

Eindhoven University of Technology

2MMC10 – Cryptology

Diffie–Hellman key exchange

- ▶ 1976 Diffie and Hellman introduce public-key cryptography.
- ▶ To use it, standardize group G and $g \in G$.
Everybody knows G and g as well as how to compute in G .
- ▶ Warning #1: Many G are unsafe!

Diffie–Hellman key exchange

- ▶ 1976 Diffie and Hellman introduce public-key cryptography.
- ▶ To use it, standardize group G and $g \in G$.
Everybody knows G and g as well as how to compute in G .
- ▶ Warning #1: Many G are unsafe!
 - ▶ $G = (\mathbf{Q}, \cdot), g = 2, h_A = 65536$

Diffie–Hellman key exchange

- ▶ 1976 Diffie and Hellman introduce public-key cryptography.
- ▶ To use it, standardize group G and $g \in G$.
Everybody knows G and g as well as how to compute in G .
- ▶ Warning #1: Many G are unsafe!
 - ▶ $G = (\mathbf{Q}, \cdot)$, $g = 2$, $h_A = 65536$ means $a = 16$.
In general, just check bitlength.
 - ▶ $G = (\mathbf{F}_p, +)$, i.e., A sends $h_A \equiv ag \pmod{p}$.

Diffie–Hellman key exchange

- ▶ 1976 Diffie and Hellman introduce public-key cryptography.
- ▶ To use it, standardize group G and $g \in G$.
Everybody knows G and g as well as how to compute in G .
- ▶ Warning #1: Many G are unsafe!
 - ▶ $G = (\mathbf{Q}, \cdot)$, $g = 2$, $h_A = 65536$ means $a = 16$.
In general, just check bitlength.
 - ▶ $G = (\mathbf{F}_p, +)$, i.e., A sends $h_A \equiv ag \pmod{p}$.
Can recover a using XGCD.
- ▶ Diffie and Hellman suggested $G = (\mathbf{F}_p^*, \cdot)$ with g a primitive element, i.e., a generator of the whole group.

Diffie–Hellman key exchange

- ▶ 1976 Diffie and Hellman introduce public-key cryptography.
- ▶ To use it, standardize group G and $g \in G$.
Everybody knows G and g as well as how to compute in G .
- ▶ Warning #1: Many G are unsafe!
 - ▶ $G = (\mathbf{Q}, \cdot)$, $g = 2$, $h_A = 65536$ means $a = 16$.
In general, just check bitlength.
 - ▶ $G = (\mathbf{F}_p, +)$, i.e., A sends $h_A \equiv ag \pmod{p}$.
Can recover a using XGCD.
- ▶ Diffie and Hellman suggested $G = (\mathbf{F}_p^*, \cdot)$ with g a primitive element, i.e., a generator of the whole group.
- ▶ Used in practice $G \subset (\mathbf{F}_p^*, \cdot)$ with g an element of large prime order.
- ▶ Miller and Koblitz suggested $G = E(\mathbf{F}_p, +)$, i.e., points on an elliptic curve over a finite field with addition of points.

Diffie–Hellman key exchange

- ▶ 1976 Diffie and Hellman introduce public-key cryptography.
- ▶ To use it, standardize group G and $g \in G$.
Everybody knows G and g as well as how to compute in G .
- ▶ Warning #1: Many G are unsafe!
 - ▶ $G = (\mathbf{Q}, \cdot)$, $g = 2$, $h_A = 65536$ means $a = 16$.
In general, just check bitlength.
 - ▶ $G = (\mathbf{F}_p, +)$, i.e., A sends $h_A \equiv ag \pmod{p}$.
Can recover a using XGCD.
- ▶ Diffie and Hellman suggested $G = (\mathbf{F}_p^*, \cdot)$ with g a primitive element, i.e., a generator of the whole group.
- ▶ Used in practice $G \subset (\mathbf{F}_p^*, \cdot)$ with g an element of large prime order.
- ▶ Miller and Koblitz suggested $G = E(\mathbf{F}_p, +)$, i.e., points on an elliptic curve over a finite field with addition of points.
- ▶ Used in practice $G \subset E(\mathbf{F}_p, +)$, i.e., prime-order subgroup of points on an elliptic curve over a finite field with addition of points.
We have seen how to compute $+$ on different curve shapes,
will now study security.

Hardness assumptions

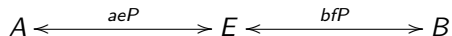
- ▶ Computational Diffie-Hellman Problem (CDHP):
Given P, aP, bP compute abP .
- ▶ Decisional Diffie-Hellman Problem (DDHP):
Given P, aP, bP , and cP decide whether $cP = abP$.
- ▶ Discrete Logarithm Problem (DLP):
Given P, aP , compute a .
- ▶ If one can solve DLP, then CDHP and DDHP are easy.

Hardness assumptions

- ▶ Computational Diffie-Hellman Problem (CDHP):
Given P, aP, bP compute abP .
- ▶ Decisional Diffie-Hellman Problem (DDHP):
Given P, aP, bP , and cP decide whether $cP = abP$.
- ▶ Discrete Logarithm Problem (DLP):
Given P, aP , compute a .
- ▶ If one can solve DLP, then CDHP and DDHP are easy.
- ▶ If one can solve CDHP, then DDHP is easy.
- ▶ In many groups, DLP and CDHP are equally hard (up to some constants).
- ▶ In some groups, DDHP is significantly easier than CDHP.

Practical problems

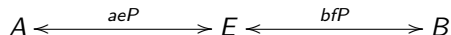
- ▶ Eve can set up a *man-in-the-middle* attack:



- ▶ E chooses e and f , presents eP to Alice as Bob's key, and fP to Bob as Alice's key.
- ▶ E computes both DH keys aeP and bfP .
- ▶ E decrypts everything from A and reencrypts it to B and vice versa.

Practical problems

- ▶ Eve can set up a *man-in-the-middle* attack:



- ▶ E chooses e and f , presents eP to Alice as Bob's key, and fP to Bob as Alice's key.
- ▶ E computes both DH keys aeP and bfP .
- ▶ E decrypts everything from A and reencrypts it to B and vice versa.
- ▶ This attack requires E to be in charge of the network.
We typically assume such strong attackers.
- ▶ This attack cannot be detected unless A and B compare their keys out of band.

Semi-static DH

- ▶ A cryptosystem combining public-key and symmetric-key crypto is called a *hybrid system*¹.
- ▶ Alice publishes long-term public key $P_A = aP$, keeps long-term private key a .
- ▶ Any user can encrypt to Alice using this key:
 - ▶ Pick random k and compute $R = kP$.
 - ▶ Encrypt message m using symmetric keys derived from $\text{KDF}(kP_A)$, for key-derivation function $\text{KDF} : G \rightarrow \{0, 1\}^n$,
 - ▶ Send ciphertext c along with R .
 - ▶ Alice decrypts, by obtaining symmetric key from

$$\text{KDF}(aR) = \text{KDF}(akP)$$

¹Now also used for combining pre-quantum and post-quantum systems.

Semi-static DH

- ▶ A cryptosystem combining public-key and symmetric-key crypto is called a *hybrid system*¹.
- ▶ Alice publishes long-term public key $P_A = aP$, keeps long-term private key a .
- ▶ Any user can encrypt to Alice using this key:
 - ▶ Pick random k and compute $R = kP$.
 - ▶ Encrypt message m using symmetric keys derived from $\text{KDF}(kP_A)$, for key-derivation function $\text{KDF} : G \rightarrow \{0, 1\}^n$,
 - ▶ Send ciphertext c along with R .
 - ▶ Alice decrypts, by obtaining symmetric key from

$$\text{KDF}(aR) = \text{KDF}(akP)$$

- ▶ Alice's key here is static, Bob's key is ephemeral.
- ▶ Note: ephemeral does not mean one-time; it means that is not long term.

¹Now also used for combining pre-quantum and post-quantum systems.

Semi-static DH

- ▶ A cryptosystem combining public-key and symmetric-key crypto is called a *hybrid system*¹.
- ▶ Alice publishes long-term public key $P_A = aP$, keeps long-term private key a .
- ▶ Any user can encrypt to Alice using this key:
 - ▶ Pick random k and compute $R = kP$.
 - ▶ Encrypt message m using symmetric keys derived from $\text{KDF}(kP_A)$, for key-derivation function $\text{KDF} : G \rightarrow \{0, 1\}^n$,
 - ▶ Send ciphertext c along with R .
 - ▶ Alice decrypts, by obtaining symmetric key from

$$\text{KDF}(aR) = \text{KDF}(akP)$$

- ▶ Alice's key here is static, Bob's key is ephemeral.
- ▶ Note: ephemeral does not mean one-time; it means that is not long term.
- ▶ Attacker solving DLP or CDHP can *compute* shared secret. Attacker solving DDHP can *confirm* guess.

¹Now also used for combining pre-quantum and post-quantum systems.