Securitatea Sistemelor Informatice

- Curs 10.2 -
Problema logaritmului discret

Adela Georgescu

Facultatea de Matematică și Informatică
Universitatea din București
Anul universitar 2022-2023, semestrul I

Algoritmi pentru calculul logaritmului discret

- Reamintim PLD:

Algoritmi pentru calculul logaritmului discret

- Reamintim PLD:
- Fie $\mathbb{G}$ un grup ciclic de ordin q (cu $|q| = n$) iar g este generatorul lui $\mathbb{G}$.

Algoritmi pentru calculul logaritmului discret

- Reamintim PLD:
- Fie $\mathbb{G}$ un grup ciclic de ordin q (cu $|q| = n$) iar g este generatorul lui $\mathbb{G}$.
- Pentru fiecare $h \in \mathbb{G}$ există un unic $x \in \mathbb{Z}_q$ a.î. $g^x = h$.

Algoritmi pentru calculul logaritmului discret

- Reamintim PLD:
- Fie $\mathbb{G}$ un grup ciclic de ordin q (cu $|q| = n$) iar g este generatorul lui $\mathbb{G}$.
- Pentru fiecare $h \in \mathbb{G}$ există un unic $x \in \mathbb{Z}_q$ a.î. $g^x = h$.
- PLD cere găsirea lui x știind $\mathbb{G}, q, g, h$; notăm $x = \log_g h$;

Algoritmi pentru calculul logaritmului discret

- Reamintim PLD:
- Fie $\mathbb{G}$ un grup ciclic de ordin q (cu $|q| = n$) iar g este generatorul lui $\mathbb{G}$.
- Pentru fiecare $h \in \mathbb{G}$ există un unic $x \in \mathbb{Z}_q$ a.î. $g^x = h$.
- PLD cere găsirea lui x știind $\mathbb{G}, q, g, h$; notăm $x = \log_g h$;
- Atenție! Atunci când $g^{x'} = h$ pentru un x' arbitrar (deci NU neapărat $x \in \mathbb{Z}_q$), notăm $log_g h = [x' \bmod q]$

Algoritmi pentru calculul logaritmului discret

- Problema PLD se poate rezolva, desigur, prin forță brută, calculând pe rând toate puterile x ale lui g până când se găsește una potrivită pentru care $g^x = h$;

Algoritmi pentru calculul logaritmului discret

- Problema PLD se poate rezolva, desigur, prin forță brută, calculând pe rând toate puterile x ale lui g până când se găsește una potrivită pentru care $g^x = h$;
- Complexitatea timp este $\mathcal{O}(q)$ iar complexitatea spațiu este $\mathcal{O}(1)$;

Algoritmi pentru calculul logaritmului discret

- Problema PLD se poate rezolva, desigur, prin forță brută, calculând pe rând toate puterile x ale lui g până când se găsește una potrivită pentru care $g^x = h$;
- Complexitatea timp este $\mathcal{O}(q)$ iar complexitatea spațiu este $\mathcal{O}(1)$;
- Dacă se precalculează toate valorile (x, g^x), căutarea se face în timp $\mathcal{O}(1)$ și spațiu $\mathcal{O}(q)$;

Algoritmi pentru calculul logaritmului discret

- Problema PLD se poate rezolva, desigur, prin forță brută, calculând pe rând toate puterile x ale lui g până când se găsește una potrivită pentru care $g^x = h$;
- Complexitatea timp este $\mathcal{O}(q)$ iar complexitatea spațiu este $\mathcal{O}(1)$;
- Dacă se precalculează toate valorile (x, g^x), căutarea se face în timp $\mathcal{O}(1)$ și spațiu $\mathcal{O}(q)$;
- Sunt de interes algoritmii care pot obține un timp mai bun la rulare decât forța brută, realizând un compromis spațiu-timp.

Algoritmi pentru calculul logaritmului discret

- Se cunosc mai mulți astfel de algoritmi împărțiți în două categorii:

Algoritmi pentru calculul logaritmului discret

- Se cunosc mai mulți astfel de algoritmi împărțiți în două categorii:
  - algoritmi *generici* care funcționează în grupuri arbitrare (i.e. orice grupuri ciclice);

Algoritmi pentru calculul logaritmului discret

- Se cunosc mai mulți astfel de algoritmi împărțiți în două categorii:
  - algoritmi *generici* care funcționează în grupuri arbitrare (i.e. orice grupuri ciclice);
  - algoritmi *non-generici* care lucrează în grupuri *specifice* - exploatează proprietăți speciale ale anumitor grupuri

Algoritmi pentru calculul logaritmului discret

- Se cunosc mai mulți astfel de algoritmi împărțiți în două categorii:
  - algoritmi *generici* care funcționează în grupuri arbitrare (i.e. orice grupuri ciclice);
  - algoritmi *non-generici* care lucrează în grupuri *specifice* - exploatează proprietăți speciale ale anumitor grupuri

- Dintre algoritmii generici enumerăm:

Algoritmi pentru calculul logaritmului discret

- Se cunosc mai mulți astfel de algoritmi împărțiți în două categorii:
  - algoritmi *generici* care funcționează în grupuri arbitrare (i.e. orice grupuri ciclice);
  - algoritmi *non-generici* care lucrează în grupuri *specifice* - exploatează proprietăți speciale ale anumitor grupuri

- Dintre algoritmii generici enumerăm:

- Metoda **Baby-step/giant-step**, datorată lui Shanks, calculează logaritmul discret într-un grup de ordin q în timp $\mathcal{O}(\sqrt{q} \cdot (\log q)^c)$;

Algoritmi pentru calculul logaritmului discret

- Se cunosc mai mulți astfel de algoritmi împărțiți în două categorii:
  - algoritmi *generici* care funcționează în grupuri arbitrare (i.e. orice grupuri ciclice);
  - algoritmi *non-generici* care lucrează în grupuri *specifice* - exploatează proprietăți speciale ale anumitor grupuri
- Dintre algoritmii generici enumerăm:
- Metoda **Baby-step/giant-step**, datorată lui Shanks, calculează logaritmul discret într-un grup de ordin q în timp $\mathcal{O}(\sqrt{q} \cdot (\log q)^c)$;
- pentru $g \in mathbbG$ generator, elementele lui $\mathbb{G}$ sunt

$$1 = g^0, g^1, g^2, ..., g^{q-1}, g^q = 1$$

Algoritmi pentru calculul logaritmului discret

- Se cunosc mai mulți astfel de algoritmi împărțiți în două categorii:
  - algoritmi *generici* care funcționează în grupuri arbitrare (i.e. orice grupuri ciclice);
  - algoritmi *non-generici* care lucrează în grupuri *specifice* - exploatează proprietăți speciale ale anumitor grupuri

- Dintre algoritmii generici enumerăm:

- Metoda **Baby-step/giant-step**, datorată lui Shanks, calculează logaritmul discret într-un grup de ordin q în timp $\mathcal{O}(\sqrt{q} \cdot (\log q)^c)$;

- pentru $g \in mathbbG$ generator, elementele lui $\mathbb{G}$ sunt

$$1 = g^0, g^1, g^2, ..., g^{q-1}, g^q = 1$$

- știm că $h = g^x$ se află între aceste valori

Metoda Baby-step/giant-step

Metoda Baby-step/giant-step

- marcam și memorăm anumite puncte din grup, aflate la distanța $t = \lfloor \sqrt{q} \rfloor$ (*giant-steps*)

$$g^0, g^t, g^{2t}, \ldots, g^{\lfloor q/t \rfloor \cdot t}$$

Metoda Baby-step/giant-step

- marcam și memorăm anumite puncte din grup, aflate la distanța $t = \lfloor \sqrt{q} \rfloor$ (*giant-steps*)

$$g^0, g^t, g^{2t}, ..., g^{\lfloor q/t \rfloor \cdot t}$$

- știm că $h = g^x$ se află în unul din aceste intervale

Metoda Baby-step/giant-step

- marcam și memorăm anumite puncte din grup, aflate la distanța $t = \lfloor \sqrt{q} \rfloor$ (*giant-steps*)

$$g^0, g^t, g^{2t}, ..., g^{\lfloor q/t \rfloor \cdot t}$$

- știm că $h = g^x$ se află în unul din aceste intervale
- calculand, cu *baby-steps*, valorile

$$h \cdot g^1, h \cdot g^2, ..., h \cdot g^t$$

Metoda Baby-step/giant-step

- marcam și memorăm anumite puncte din grup, aflate la distanța $t = \lfloor \sqrt{q} \rfloor$ (*giant-steps*)

$$g^0, g^t, g^{2t}, ..., g^{\lfloor q/t \rfloor \cdot t}$$

- știm că $h = g^x$ se află în unul din aceste intervale
- calculand, cu *baby-steps*, valorile

$$h \cdot g^1, h \cdot g^2, ..., h \cdot g^t$$

- una din ele va fi egala cu unul din punctele marcate i.e. $h \cdot g^i = g^{k \cdot t}$

Metoda Baby-step/giant-step

- marcam și memorăm anumite puncte din grup, aflate la distanța $t = \lfloor \sqrt{q} \rfloor$ (*giant-steps*)

$$g^0, g^t, g^{2t}, ..., g^{\lfloor q/t \rfloor \cdot t}$$

- știm că $h = g^x$ se află în unul din aceste intervale
- calculand, cu *baby-steps*, valorile

$$h \cdot g^1, h \cdot g^2, ..., h \cdot g^t$$

- una din ele va fi egala cu unul din punctele marcate i.e. $h \cdot g^i = g^{k \cdot t}$
- Complexitatea timp este $\mathcal{O}(\sqrt{q} \cdot polylog(q))$ iar complexitatea spațiu este $\mathcal{O}(\sqrt{q})$

Algoritmi generici pentru calculul logaritmului discret

- Metoda Baby-Step/Giant-Step este optimă ca timp de rulare, însă există alți algoritmi mai eficienți d.p.d.v. al complexității spațiu;

Algoritmi generici pentru calculul logaritmului discret

- Metoda Baby-Step/Giant-Step este optimă ca timp de rulare, însă există alți algoritmi mai eficienți d.p.d.v. al complexității spațiu;
- Algoritmul Pohlig-Hellman poate fi folosit atunci când se cunoaște factorizarea ordinului q al grupului iar timpul de rulare depinde de factorii primi ai lui q;

Algoritmi generici pentru calculul logaritmului discret

- Metoda Baby-Step/Giant-Step este optimă ca timp de rulare, însă există alți algoritmi mai eficienți d.p.d.v. al complexității spațiu;
- Algoritmul Pohlig-Hellman poate fi folosit atunci când se cunoaște factorizarea ordinului q al grupului iar timpul de rulare depinde de factorii primi ai lui q;
- Pentru ca algoritmul să nu fie eficient, trebuie ca cel mai mare factor prim al lui q să fie de ordinul 2^{160}.

Algoritmi non-generici pentru calculul logaritmului discret

- Algoritmii non-generici sunt potențial mai puternici decât cei generici;

Algoritmi non-generici pentru calculul logaritmului discret

- Algoritmii non-generici sunt potențial mai puternici decât cei generici;
- Cel mai cunoscut algoritm pentru PLD în $\mathbb{Z}_p^*$ cu p prim este algoritmul GNFS (General Number Field Sieve) cu complexitate timp $2^{\mathcal{O}(n^{1/3} \cdot (\log n)^{2/3})}$ unde $|p| = \mathcal{O}(n)$;

Algoritmi non-generici pentru calculul logaritmului discret

- Algoritmii non-generici sunt potențial mai puternici decât cei generici;
- Cel mai cunoscut algoritm pentru PLD în $\mathbb{Z}_p^*$ cu p prim este algoritmul GNFS (General Number Field Sieve) cu complexitate timp $2^{\mathcal{O}(n^{1/3} \cdot (\log n)^{2/3})}$ unde $|p| = \mathcal{O}(n)$;
- Există și un alt algoritm non-generic numit *metoda de calcul a indicelui* care rezolvă DLP în grupuri ciclice $\mathbb{Z}_p^*$ cu p prim în timp sub-expoențial în lungimea lui p.

Algoritmi non-generici pentru calculul logaritmului discret

- Algoritmii non-generici sunt potențial mai puternici decât cei generici;
- Cel mai cunoscut algoritm pentru PLD în $\mathbb{Z}_p^*$ cu p prim este algoritmul GNFS (General Number Field Sieve) cu complexitate timp $2^{\mathcal{O}(n^{1/3}\cdot(\log n)^{2/3})}$ unde $|p| = \mathcal{O}(n)$;
- Există și un alt algoritm non-generic numit *metoda de calcul a indicelui* care rezolvă DLP în grupuri ciclice $\mathbb{Z}_p^*$ cu p prim în timp sub-expoențial în lungimea lui p.
- Această metodă seamănă cu algoritmul sitei pătratice pentru factorizare;

Important de reținut!

- Cel mai bun algoritm pentru DLP este sub-exponențial;
- Se pot construi funcții hash rezistente la coliziuni bazate pe dificultatea DLP;