Zajęcia 11 Łamanie Diffiego-Hellmana, czyli odwracanie potęgowania modulo

Bezpieczeństwo protokołu D-H opiera się na tym że nie jest łatwo odwrócić potęgowanie modulo, tj. wyznaczyć x, taki że a^x mod n = d.

Pierwsza metoda: próbne mnożenie
Najprostszą metodą jest próbowanie jednej po drugiej wykładników x i sprawdzanie kiedy powyższe równanie będzie spełnione.

Przykład 3^x mod 17 = 2
zaczynamy od 3^1 mod 17 =3 (nie spełnione)
w kolejnych krokach wykorzystujemy poprzednie:
3^2 mod 17 = (3^1 * 3) mod 17 = (3 * 3) mod 17 = 9
3^3 mod 17 = (3^2 * 3) mod 17 = (9 * 3) mod 17 = 27 mod 17 = 10
3^4 mod 17 = (3^3 * 3) mod 17 = (10 * 3) mod 17 = 30 mod 17 = 13
3^5 mod 17 = 5
…
3^{14} mod 17 = 2, czyli x = 14

Metoda ta jest bardzo wolna, bo musimy sprawdzić wszystkie liczby aż do x (które jest rzędu n)– złożoność liniowa

Metoda baby step – giant step
W tej metodzie mamy mniejszą złożoność obliczeniową kosztem zapotrzebowania na pamięć (obie są teoretycznie sqrt(x).

równanie a^x mod n = d możemy zapisać jako $a^{i\,m+j}$ mod n = d (czyli x = i * m +j), gdzie m jest sqrt(n), zaokrąglony w górę, a 'i' i 'j' są z zakresu 0 do m-1.
Równanie możemy zapisać w postaci
a^j mod n = (d * $(a^{-m})^i$) mod n
Ujemnej potęgi możemy się pozbyć zastepując a^{-m} przez v^m, gdzie (a * v) mod n = 1. Ostatni warunek to odwracanie modulo które już przerabialiśmy.

Istotą algorytmu jest że liczymy osobno lewą stronę, zapisujemy w tablicy dla i od 0 do m-1 i potem liczymy prawą stronę za każdym razem sprawdzając czy zgodzi nam się z którymś z elementów tablicy.

Przykład 3^x mod 17 = 2, m =sqrt(17) czyli zaokrąglone w górę m=5
liczymy 3j mod 17 dla j=0 do 4 (najlepiej tak jak w metodzie próbnego mnożenia)
j=0: 3^0 mod 17 =1
j=1: 3^1 mod 17 =3
j=2: 3^2 mod 17 =9
j=3: 3^3 mod 17 =10
j=4: 3^4 mod 17 =13
teraz liczymy prawą stronę, ale zanim to zrobimy wyznaczmy v z równania (3*v)mod 17 =1 (w ten sam sposób jak na poprzednich zajęciach), no i dostajemy v = 6, czyli v^m mod n =6^5 mod 17 = 7.
i=0: d mod n = 2 mod 17 = 2: nie zgadza się z żadną wartości a^j mod n
i=1: d * v mod n = (2*7) mod 17 = 14: nie zgadza się
i=2: (d * v^2) mod n = ((d * v mod n) * v) mod n = (14 * 7) mod 17 = 13 – zgadza się z j=4

Ponieważ zgodziły nam się wartości dla j=4 z i=2 to nasze x = 2 * 5 + 4 = 14.
No i faktycznie po sprawdzeniu 3^{14} mod 17 = 2

Uwaga – złożoność obliczeniowa jest rzędu pierwiastka z n, bo i po lewej i po prawej stronie liczymy do m~sqrt(n). Ale trzeba zauważyć, że mamy też krok sprawdzania czy lewa strona równa się prawej. W najprostszym przypadku jeżeli a^j mod n są zapisane w tablicy (albo słowniku!) i mamy przeszukiwanie w tablicy to jego złożoność jest znowu m~sqrt(n), czyli łączna złożoność wynosi ponownie n i jest niewiele lepsza od pierwszej metody.

Żeby wykorzystać pełną moc tej metody musimy mieć sposób szybkiego przeszukiwania a_j. Najprościej to zrobić wykorzystując prostą funkcję mieszającą.
Oznaczmy $a_j=a^j$ mod n i wybierzmy liczbę pierwsza p która jest bliska m.
Funkcję mieszającą będziemy liczyć przez:
hash = a_j mod p.
Tworzymy wtedy tablicę rozmiaru p w której każdy element będzie dynamiczną tablicą wszystkich a_j które mają taka samą funkcję mieszającą. Wtedy zamiast szukać w pełnej tablicy a_j wystarczy policzyć funkcję mieszającą tego co chcemy znaleźć i przeszukać tylko małą dynamiczną tablicę znajdująca się na pozycji tej funkcji mieszającej. Reszta algorytmu wygląda tak samo

Dla przykładu powyżej:
m =5 więc możemy wziąć również p=5,
a_0=1; a_0 mod 5=1
a_1=3; a_1 mod 5=3
a_2=9; a_2 mod 5=4
a_3=10; a_3 mod 5=0
a_4=13; a_4 mod 5=3

czyli nasza tablica funkcji mieszających wygląda tak:
hash = 0: a_3=10
hash = 1: a_0=1
hash = 2: <puste>
hash = 3: a_1=3, a_4=13
hash = 4: a_2=9

liczymy teraz prawą stronę (tak jak poprzednio)
i=0: d mod n = 2 mod 17 = 2; 2 mod 5 = 2; tablica dla hash = 2 jest pusta – brak rozwiązań
i=1: d * v mod n = (2*7) mod 17 = 14; 14 mod 5 = 4, dla hash =4 mamy jedynie a_2=9, ale nie ma 14, więc ciągle nie ma rozwiązań
i=2: (d * v^2) mod n = (14 * 7) mod 17 = 13; 13 mod 5 = 3, dla hash = 3 mamy a_1=3, a_4=13 i druga pozycja z listy się zgadza, więc ponownie dostajemy j=4

UWAGA: nie należy mylić odwracania potęgowania modulo (które jak wyjaśniam wyżej jest skomplikowane obliczeniowo), czyli a^x mod n =c, x = ? z odwracaniem modulo (które wykorzystywaliśmy np. w RSA i jest szybkie obliczeniowo), czyli a * x mod n = 1, x = ?.