DSA

Patryk Lisik 210254

Barbara Morawska 210279

Adam Sadowski 210310

1. Generacja parametrów

Na potrzeby działania algorytmu generowane są 3 parametry: **p**, **q** i **g**.

* 1. **q**: liczba pierwsza o określonej liczbie bitów
  2. **p**: liczba pierwsza o określonej liczbie bitów taka, że **p**-1 jest wielokrotnością **q**

Zamiast generować liczbę pierwszą p(1024 bity) i sprawdzać, czy (**p** - 1) % **q** = 0, generujemy dodatkową liczbę 864 bitową, mnożymy przez **q**, dodajemy 1 i sprawdzamy czy jest pierwsza.

* 1. **g**: liczba, dla której q jest najmniejszą dodatnią liczbą taką, że

**gq**=1 mod **p**

Możliwe kombinacje ilości bitów liczb **p** i **q**: (1024, 160), (2048, 224), (2048, 256), (3072, 256).

Parametry (**p**, **q**, **g**) mogą być współdzielone przez różnych użytkowników systemu

1. Generacja kluczy użytkownika
   1. Generowany jest klucz prywatny **x**, mniejszy od **q**
   2. Generowany jest klucz publiczny  **y** = **gx** mod **p**
2. Podpisywanie

Niech **H** będzie funkcją hashującą(w naszym przypadku SHA-1), a **m** wiadomością

1. Generowana jest losowa liczba **k**, po jednej na wiadomość, gdzie **k**<**q**
2. Liczone jest **r** = (**gk**mod **p**) mod **q**
   1. Jeżeli **r** = 0, wróć do punktu 3.a
3. Liczone jest **s** = **k-1**(**H**(**m**) + **xr**) mod **q**
   1. Jeżeli **s** = 0, wróć do punktu 3.a

Otrzymana para liczb (**r**, **s**) jest podpisem danej wiadomości

1. Weryfikacja
   1. Podpis jest odrzucony, jeżeli 0 < **r** < **q** lub 0 < **s** < **q**
   2. Liczone jest **w** = **s**-1 mod **q**
   3. Liczone jest **u1** = **H**(**m**)**w** mod **q**
   4. Liczone jest **u2** = **rw** mod **q**
   5. Liczone jest **v** = (**gu1 yu2**mod **p**) mod **q**
   6. Podpis jest prawidłowy wtedy i tylko wtedy, gdy **v** = **r**
2. Przykład
   1. Generowana jest liczba pierwsza **q** = 845931091825522617070507672575421950618458198729
   2. Generowana jest liczba pierwsza **p** = 66202895492720086530338948820761796246952254793463041494976120559039646803573740915759453027521675564856531417649466426618605856721328508313041313911311044071430666792033539491336865588235831744113649222938561915481787339276539839512286876907278330641374343963556625603895045194787277747823949381297367904727
   3. Liczona jest liczba **g** = 45315104054051436951672232035565392686774795735612463340921678548703265661716192545320134464416526386765253996749796870393937587187377648664237652641705491776978215205511360081868276021608293585243875214265029887495756268643683825411263908460617478573460381646391938512164367208202625574837417183877569226414
   4. Generowany jest klucz prywatny **x** = 744528309652987671773592399775334842576166850773
   5. Liczony jest klucz publiczny **y** = 30663538380450904473984253237787210166741612467356820702095799723780396214130738477006451222661785104720004379483713038213001997786392957128034276321788973811350144589051319510427143636391838838699402492963516223670317954295757037751001268309233932622730870561943759798441316580678880875877262881100117871175
   6. Liczone jest SHA-1 wiadomości: 1249035667134964717426977061132101334259232579545
   7. Generowana jest liczba **k** = 98289567908473854794179832472637629393641523116
   8. Liczona jest liczba **r** = 53942008434170076533538769557997747395741985824
   9. Liczona jest liczba **s** = 500683257809261814259873158546413074642513481685

Która wraz z liczbą **r** tworzą podpis

* 1. Liczona jest liczba **w** = 669318764025854241273940630509827969931746263451
  2. Liczona jest liczba **u1** = 836003008950982875349698322328360318876602988126512189561296860105781700981989825094817883709795
  3. Liczona jest liczba **u2** = 36104398414230920692071547381749443479478073836643489058355164977332352969734272935190611318624
  4. Liczona jest liczba **v** = 53942008434170076533538769557997747395741985824 = **r**

Więc podpis jest prawidłowy

1. Pomocnicze algorytmy:
   1. Generacja liczb pierwszych wykonana jest za pomocą wbudowanej funkcji generującej liczbę n-bitową, a następnie przepuszczana przez probabilistyczny algorytm Millera-Rabina(implementacja własna) z prawdopodobieństwem wystąpienia błędu 7.458341 \* 10-157%
   2. Algorytm szybkiego potęgowania z modułem
   3. Liczenie inwersji modularnej a-1 mod b jako ab-2 mod b
   4. Do Hashowania wykorzystano algorytm SHA-1