Teoria Liczb i pokrewne algorytmy w Kryptografii

Streszczenie

Ta analiza nowego protokołu dystrybucji Klucza Publicznego służy dwóm celom: zabezpieczeniu przeciw aktywnym przeciwnikom, i możliwości uwierzytelniania klucza. Ten protokół oparty na Problemie Diffiego-Hellmana jest protokołem dwuprzebiegowym i ma wiele pożądanych zabezpieczeń. Protokół zakłada współdzielenie tajnego klucza K pomiędzy dwie jednostki. Protokół jest rozszerzeniem Wymiany Klucza Diffie-Hellmana używającego liczb losowych. Ta analiza podąża śladem MTI ,który jest podstawą w dwuprzebiegowym uzgadnianiu klucza. Generalnie, analiza skupi się na rozwijaniu protokołu ,aby użytkownicy mogli uwierzytelniać jeden drugiego w niektórych sieciach bez centralnego uwierzytelniania.

Zawartość

Rozdział 1 Wprowadzenie

Kryptografia ma długą i fascynującą historię. Najbardziej zadziwiającym wydarzeniem w historii kryptografii było opublikowanie w 1976 roku przez **Diffiego** i **Hellmana** *"New Directions in Cryptography* "[14]. Ten dokument wprowadził rewolucyjną koncepcję kryptografii z kluczem publicznym jak również dostarczył nowej i błyskotliwej metody dla wymiany klucza, bezpieczeństwo, które jest oparte o trudny do rozpracowania problem logarytmy dyskretnego a który jest powszechnie używanym protokołem dla wymiany klucza. Później, kilka publicznych kryptosystemów używało wielu podobnych pomysłów. Wiele z nich okazywało się nie być bezpiecznymi.Jednak, protokół Diffie-Hellmana wydaje się być jednym z najsilniejszych do dzisiaj. Algorytm wymiany klucza Diffie-Hellmana opiera się na założeniu ,że logarytmy dyskretne są trudne do wyliczenia. Ta trudna hipoteza jest również podstawą do wnioskowania bezpieczeństwa różnorodnych innych schematów klucza publicznego. Podczas gdy nastąpił postęp dotyczący algorytmów logarytmu dyskretnego w ostatnich dwóch dekadach, generalnie logarytm dyskretny pozostaje jeszcze dużym problemem. Niestety żadnych dowodów na twardość nie są dostępne w tej dziedzinie, więc jest konieczne poleganie na doświadczeniu i intuicji w ocenianiu, jakich parametrów używa się dla kryptosystemów.

W wielu protokołach kryptograficznych dwie osoby życzą sobie komunikować się ze sobą. Jednak, zakładamy,że początkowo nie posiadają one żadnego wspólnego tajnego klucza a zatem nie mogą użyć tajnego klucza kryptosystemów. Wymiana klucza w protokole Diffie-Hellmana rozwiązuje tą sytuację przez uwzględnienie budowy wspólnego tajnego klucza ponad niepewnym kanałem komunikacyjnym. Bazuje on na problemie związanym z logarytmami dyskretnymi, mianowicie problemie Diffie-Hellmana. Problem ten jest rozpatrywany jako trudny, i jest w pewnych przypadkach tak trudny jak problem logarytmu dyskretnego. Protokół Diffie-Hellmana jest generalnie rozważany jako bezpieczny, kiedy jest stosowana właściwa grupa matematyczna. W szczególności, generator elementów używany przy potęgowaniu powinien mieć duży okres.

Cel dystrybucji klucza lub protokołu uzgadniania klucza jest taki ,że na końcu protokołu, dwie części obejmują zarówno posiadania tego samego klucza K, a wartość K nie jest znana żadnej inne części. Z pewnością jest to o wiele trudniejsze do zaprojektowania protokołu zakładającego ten typ bezpieczeństwa.

Ataki przeciwko Diffie-Hellmanowi to przede wszystkim ataki typu *man-in-the-middle*. Jest to w praktyce bardzo łatwe jeśli protokół nie stosuje środków zaradczych takich jak uwierzytelnianie. Ponieważ

sieć jest niezabezpieczona dla *n* użytkowników, musimy ochronić się przeciw potencjalnym przeciwnikom (zwanych również *adwersarzem, intruzem, wrogiem, atakującym, podsłuchującym,* i *podszywającym* w różnych sytuacjach). Przeciwnik może być *pasywnym adwersarzem, który* próbuje pokonać technikę kryptograficzną przez proste nagrywanie danych a potem ich analizę poprzez *atak pasywny.* Z drugiej strony, *atak aktywny* dotyczy *aktywnego atakującego, który* modyfikuje lub wprowadza wiadomości. Aktywny atakujący może wykonywać różne typy paskudnych rzeczy, takich jak te poniżej

1. zmodyfikować wiadomości, które obserwuje podczas transmisji przez sieć.
2. zachować wiadomości do ponownego użycia w późniejszym czasie
3. spróbować maskarady jako różni użytkownicy w sieci

Cel aktywnego atakującego może być jednym z dwóch:

1. nabrać użytkowników aby zaakceptowali "niepoprawny " klucz jako poprawny
2. sprawić, aby użytkownicy uwierzyli,że wymienili klucz każdy z każdym, kiedy tak jednak nie było.

1.1 Cele analizy

W większości Algorytmów Wymiany Klucza *zaufany autorytet* jest odpowiedzialny za weryfikację tożsamości użytkowników, wybór i transmisję kluczy do użytkowników itp Czasami jest tak,że ta analiza pozwala tego uniknąć. Ponieważ są przypadki, że użytkownicy chcą komunikować się bezpośrednio.

Jako cel podstawowy, celem tej analizy jest rozwiniecie protokołu dystrybucji tajnego klucza tak, aby osiągnąć poniższe cele :

1. zabezpieczenie przeciwko atakom man-in-the-middle
2. zabezpieczenie oparte o trudności problemu Diffiego-Hellmana
3. zdolności do uwierzytelnienia klucza, dostarczenie zabezpieczenia dla odbiorcy ,czy on lub ona wyliczył poprawnie klucz

Ta analiza skupia się na rozwinięciu protokołu, przez który użytkownicy mogą uwierzytelniać się wzajemnie w niezabezpieczonej sieci bez centralnego uwierzytelniającego. Stosując taki protokół, użytkownicy będą mogli poprawnie zidentyfikować początek wiadomości, z pewnością,że tożsamość nie jest sfałszowana.

1.2 Działania pokrewne

Jest wiele algorytmów opartych o trudny do rozwiązania Problem Logarytmu Dyskretnego,które są zabezpieczone przed atakiem pasywnym,ale nie specjalnie przed atakiem man-in-the-middle,który będzie omawiany w tej analizie.

Yamamoto i Akiyama zaproponowali protokół nazwany Method 1 protokołu uzgadniania klucza,który jest rozszerzeniem protokołu Diffiego-Hellmana, używającym liczb losowych dla kluczy sesyjnych. Takie działanie jest zabezpieczeniem przeciwko atakowi pasywnemu ale ma już problem z atakiem man-in-the-middle.

Inne działanie zaproponowane przez Okamoto i Nakamurę to Method 2 protokołu uzgadniania klucza używający liczb losowych dla klucza sesyjnego w inny sposób niż Method 1. Bezpieczeństwo tej metody jest takie samo jak w Method 1.

Matsumoto, Takashima i Imai skonstruowali kilka interesujących protokołów uzgadniania klucza przez modyfikację wymiany klucza Diffiego-Hellmana. Protokoły te są sklasyfikowane jako protokoły uzgadniania klucza MTI. Zaprezentuję kilka z tych protokołów i pokażę atak typu man-in-middle na jeden z nich, aby pokazać jak atakujący może oszukać użytkowników poprzez akceptację "niewłaściwego " klucza jako poprawnego.

Protokół Elgamala jest innym działaniem, które jest dużo trudniejsze niż metody wymienione powyżej. Jest to protokół jednoprzebiegowy dający uwierzytelnianie, ale nie jest on zabezpieczony przeciwko atakowi typu man-in-the-middle.

Uwierzytelnianie jest drugim problemem w tych protokołach. Są one protokołami nieuwierzytelnionymi, w których użytkownicy nie mogą być pewni ,czy wygenerowały one poprawny klucz. W kilku protokołach, takich jak ElGamal lub jednoprzebiegowe protokoły,odbiorca nie ma potwierdzenia z kim współdzieli tajny klucz, ani też nie jest pewny czy klucz jest bezpieczny. Ani nie uzyskuje uwierzytelnienia jednostki ani potwierdzenia klucza.

1.3 Zarys analizy

Ta analiza została zorganizowana w 5 rozdziałach. Rozdział 1 daje wprowadzenie do tła historycznego Wymiany Klucza Diffiego-Hellmana jak również działań pokrewnych. Rozdział 2 przedstawia matematyczne podstawy potrzebne dla zrozumienia protokołów opartych o Diffie-Hellman i typy ataków na jakie są narażone. Protokoły oparte o Diffiego-Hellmana wymagają wiedzy z kilku obszarów matematyki, wliczając w to teorię liczb, grupy, pierścienie i pola. Spróbuję się skupić na najważniejszych i koniecznych definicjach oraz twierdzeniach,które są konieczne do zrozumienia tej analizy. Oczywiście, nie opiszę żadnych dowodów tych twierdzeń,.

W Rozdziale 3 omówię protokoły DiffieHellmana, Elgamala, i MTI. Celem tego rozdziału jest przegląd protokołów opartych o Diffiego-Hellmana, konieczny do zrozumienia kolejnego rozdziału.Ponadto, rozważymy atak typu man-in-the-middle na protokół Diffie-Hellmana i MTI/A0.

Rozdział 4 ,który jest główną częścią tego tekstu, składa się z protokołu, jaki jest proponowany w tej analizie. Ten protokół jest przedłużeniem protokołów Diffiego-Hellmana i MTI/C1

.

Punkty bezpieczeństwa będą omówione w dalszej części tego rozdziału. Omówimy dwa przypadki ataku typu man-in-the-middle na ten protokół. Większość tych ataków jest oparta na sztuczkach matematycznych ,które postaram się rozważyć. Ostatni rozdział składa się z wniosków i przyszłych działań.

Rozdział 2 Podstawy matematyczne

Rozdział ten jest zbiorem podstawowego materiału z Teorii Liczb, Grup, Pierścieni, Pól I Pól Skończonych ,które będą stosowane w tej analizie. Celem tego rozdziału jest przypomnienie sobie elementów Teorii Liczb potrzebnych w dalszej pracy.

2.1 Zagadnienia w Teorii Liczb

2.1.1 Podzielność

Zbiór liczb całkowitych {... , -3, -2, -1, 0,1, 2, 3,... } jest oznaczony symbolem Z.

Definicja 2.1.1 Niech *a, b* będą liczbami całkowitymi. Wtedy *a dzieli się przez b* (równoważnik: a jest *dzielnikiem dla b,* lub a jest *czynnikiem b)* jeśli istnieje liczba całkowita c taka,że $b = ac$. Jeśli *a* dzieli się przez *b,* wtedy oznaczone jest to przez $a|b$.

Przykład 2.1.1 (i) $-3|18$, ponieważ $18 = (-3)(-6)$. (ii) $173|0$, ponieważ $0 = (173)(0)$.

Poniżej jest kilka podstawowych właściwości podzielności.

Twierdzenie 2.1.1 *(właściwości podzielności)*

1. $a|a$.
2. Jeśli $a|b$ i $b|c$,to a|c
3. Jeśli $a|b$ i $a|c$ to $a|(bx + cy)$ dla wszystkich $x, y \in Z$.
4. Jeśli $a|b$ i $b|a$, to $a = \pm b$.

Definicja 2.1.2 *(algorytm dzielenia liczb całkowitych)* jeśli *a* i *b* są liczbami całkowitymi z $b \geq 1$, wtedy zwykłe dzielenie przez liczbę wielocyfrową *a* przez *b* daje liczbą całkowitą *q* (*iloraz)* i r (*reszta)* takie ,że

$a = qb + r$, gdzie $0 < r < b$.

Ponadto q i r są jednorazowe. Reszta z dzielenia jest oznaczona przez *a mod b,* a iloraz jest oznaczony przez *a div b.*

Przykład 2.1.2 jeśli $a = 73$, $b = 17$, wtedy $q = 4$ a $r = 5$. Dlatego ,że 73 *mod* 17 = 5 a 73 *div17* = 4.

Definicja 2.1.3 Liczba całkowita c jest *wspólnym dzielnikiem* *a* i *b* jeśli *c*|*a* i c|b.

Definicja 2.1.4 Nieujemna liczba całkowita d jest *największym wspólnym dzielnikiem* liczb całkowitych *a* i b, oznaczonym *d = gcd(a, b),* jeśli

1. *d* jest wspólnym dzielnikiem *a* i *b;* i
2. kiedy c|a, i *c*|*b,* wtedy c|d

Odpowiednio, *gcd(a, b)* jest największą dodatnią liczbą całkowitą, która dzieli się przez *a* i *b,* z wyjątkiem, kiedy *gcd(0,* 0) = 0.

Przykład 2.1.3 Wspólne dzielniki z 12 i 18 to {±1, ±2, ±3, ±6}, a *gcd(12,*18) = 6.

Definicja 2.1.5 Nieujemna liczba całkowita *d* jest *najmniejszą wspólną wielokrotnością* liczb całkowitych *a* i *b,* oznaczone przez l = *lcm(a, b),* jeśli

1. *a*|*l* i *b*|*l;* i
2. kiedy *a*|*c,* i *b*|*c,* wtedy l|c.

Odpowiednio, *lcm(a, b)* jest najmniejszą nieujemną liczbą całkowitą podzielną przez *a* i *b.*

Twierdzenie 2.1.1 Jeśli *a* i *b* są dodatnimi liczbami całkowitymi , wtedy *lcm(a, b) = a • b/gcd(a, b).*

Przykład 2.1.4 Ponieważ *gcd(12,*18) = 16, wtedy *lcm(12,*18) = 12 • 18/6 = 36.

Definicja 2.1.6 *(liczby pierwsze)* Przy $p > 2$ mówimy,że jest *pierwsza* jeśli dzieli się tylko przez 1 i samą siebie (*p).* W przeciwnym razie, *p* jest nazywana *złożoną.*

Twierdzenie 2.1.2 Jeśli *p* jest liczbą pierwszą i *p*|*ab,* wtedy albo *p*|*a* albo p|b(albo obie).

Twierdzenie 2.1.3 Jest nieskończona liczba liczb pierwszych

Dowód
Załóżmy,że jest skończona liczba liczb pierwszych, wtedy możemy je wypisać wszystkie:

$$P i , P 2 , \ldots , , Pr$$

Niech *P* będzie ich iloczynem, bardzo dużą liczbą ale skończoną:

$$P = p1 \text{ x } p_2 \text{ x } \bullet \bullet \bullet \text{ x } p_r.$$

Teraz rozważmy $P + 1$,które jest liczbą całkowitą I może być rozłożona na czynniki pierwsze. Ale ponieważ wszystkie liczby pierwsze dzielą się przez *P,* żadna z nich nie dzieli się przez $P + 1$, ponieważ jeśli p_i dzieli się przez P i dzieli się przez $P + 1$, wtedy musi dzielić się przez 1. To jest nasza sprzeczność.

Zauważmy ,że cały ten dowód świadczy o tym,że jest nieskończenie wiele liczb pierwszych.Bez sensu jest próba generowanie liczb pierwszych .Jeśli znamy pierwsze n liczb pierwszych, pozwoli to nam na podanie nowej liczby pierwszej, ale prawdopodobnie nie kolejnej liczby pierwszej. Również nie daje to gwarancji,że $P + 1$ będzie liczbą pierwszą. Na przykład:

$$(2 \text{ x } 3 \text{ x } 5 \text{ x } 7 \text{ x } 11 \text{ x } 13) + 1 = 30031 = 59 \text{ x } 509.$$

Twierdzenie 2.1.4 *(podstawowe twierdzenie arytmetyczne)* Rozkład na czynniki pierwsze liczb pierwszych jest jednoznaczny w swoim porządku.

Dowód

W rzeczywistości udowodnimy,że każda liczba całkowita z niejednoznacznym rozkładem na czynniki pierwsze ma właściwy podzielnik z niejednoznacznym rozkładem na czynniki pierwsze. Jeśli były liczby całkowite z niejednoznacznym rozkładem na czynniki pierwsze, wtedy ostatecznie moglibyśmy zredukować je do liczby pierwszej z niejednoznacznym rozkładem na czynniki pierwsze, a to zaprzeczałoby faktowi,że jest liczbą pierwszą a zatem nie ma dodatnich podzielników innych niż 1 I ona sama.

Niech n będzie liczbą całkowitą z niejednoznacznym rozkładem na czynniki pierwsze:

$$n = p_1 \text{ x } p_2 \text{ x } \bullet \bullet \bullet \text{ x } p_r$$

$$= q_1 \text{ x } q_2 \text{ x } \bullet \bullet \bullet \text{ x } q_s,$$

gdzie liczby pierwsze nie są koniecznie wyraźne, ale gdzie drugi rozkład na czynniki pierwsze nie jest prostą zmianą uporządkowania pierwszej.Liczba pierwsza q_1 dzieli się przez n a więc dzieli się przez iloczyn p_i. Zatem, jest przynajmniej jedna p_i która jest podzielna przez q_1. Jeśli jest to konieczne, zmienimy uporządkowanie p_i tak aby q_1 dzieliło się przez p_1. Ponieważ p_1 jest liczbą pierwszą, q_1 musi równać p_1. Zatem mówimy,że

$$\frac{n}{q_1} = p_2 \text{ x } p_3 \text{ x } \blacksquare \blacksquare \blacksquare \text{ x } p_r$$

$$= q_2 \text{ x } q_3 \text{ x } \bullet \bullet \bullet \text{ x } q_s$$

Ponieważ rozkład na czynniki pierwsze n był różny, rozkład na czynniki pierwsze n/q_1 musi również by różny. Dlatego też n/q_1 jest właściwym podzielnikiem n z niejednoznacznym rozkładem na czynniki pierwsze.

gdzie p_i są różnymi liczbami pierwszymi, a e_i są dodatnimi liczbami całkowitymi. Co więcej, rozkład jest jednoznaczny aż do kombinacji czynników.

Twierdzenie 2.1.5 Jeśli $a = p_1^{e1} p_2^{e2} \ldots p_k^{ek}$, $b = p_1^{f1} p_2^{f2} \ldots p_k^{fk}$, gdzie każde $e_i > 0$ i $fi > 0$,wtedy

$$gcd(a,b)=p_1^{min(e1f1)}p_2^{min(e2,f2)}....p_k^{min(ek,fk)}$$

a

$$lcm(a,b)=p_1^{max(e1,f1)}p_2^{max(e2,f2)}....p_k^{max(ek,fk)}$$

Przykład 2.1.5 Niech $a = 4864 = 2^8 \bullet 19$, $b = 4358 = 2 \bullet 7 \bullet 13 \bullet 19$. Wtedy *gcd(4864,* 3458) = 2 • 19 = 38 a lcm(4864, 3458) = $2^8 \bullet 7 \bullet 13 \bullet 19 = 442624$.

Definicja 2.1.7 Dla $n \geq 1$, niech $\varphi(n)$ oznacza kilka liczb całkowitych w przedziale [1, *n]* które są odpowiednio liczbami pierwszymi do *n.*Funkcja φ nazywa się *funkcją phi Eulera.*

Twierdzenie 2.1.6 *(właściwości funkcji phi Eulera)*

1. Jeśli *p* jest liczbą pierwszą, wtedy $\varphi(p) = p - 1$.
2. Funkcja phi Eulera jest *multiplikatywna.* To znaczy, jeśli *gcd(m, n)* = 1, wtedy $\varphi(mn) = \varphi m) \bullet \varphi(n)$.
3. Jeśli $p_1^{e1}p_2^{e2}\ldots p_k^{ek}$ jest rozkładem na czynniki pierwsze dla n, wtedy

$$\varphi(n)=p_1^{e1-1}(p_1-1)x\ p_2^{e2-1}(p_2-1)x.....x\ p_k^{ek-1}(p_k-1)$$

$$=n(1-1/p_1)(1-1/p_2)\ldots(1-1/p_k)$$

2.1.2 Algorytm Euklidesa

Niech *a* i b będą nieujemnymi liczbami całkowitymi, każda mniejsza lub równa *n.* Liczba bitów w binarnej reprezentacji n to *[ln n*+ 1], a liczba ta jest aproksymowana przez *ln n.* Kilka operacji bitowych dla czterech podstawowych działań na liczbach całkowitych, dodawania, odejmowania, mnożenia i dzielenia używających klasycznych algorytmów jest przedstawionych w Tabeli 2.1. Zauważ

Działanie		Złożoność bitowa		
Dodawanie	a + b	O(ln a + ln b)	=	O(lnn)
Odejmowanie	a - b	O(ln a-ln b)	=	O(lnn)
Mnożenie	a • b	O((ln a)(ln b))	=	$O((lnn))^2$
Dzielenie	a = qb + r	O((ln q)(ln b))	=	$O((lnn))^2$

Tabela 2.1: *Złożoność bitowa podstawowych działań w Z*

że, powyższe równania na działaniach bitowych są dobrze znanymi instrukcjami. Inny sposób zapisu powyższych równań jest następujący. Notacja *Time(A)* oznacza liczbę bitów operacji dla pracy potrzebnej w *A*.

Time(a + b) = O(log(max(a, b))), działanie bitowe, gdzie $a, b \in Z$.

Time(a x *b)* = O(log a log b), działanie bitowe, gdzie $a, b \in Z$.

Time(a/b) = O(log a log b), działanie bitowe, gdzie $a, b \in Z$.
Time($\sqrt{a}$) = O($\log^3$a), działanie bitowe, gdzie $a \in Z$.
Time(g^a *mod b*) = O(log a $\log^2$ b), gdzie a, b $\in$ Z, dla pewnej stałej liczby całkowitej g.

Dal obliczenia największego wspólnego dzielnika dwóch liczb całkowitych, najbardziej wydajnym algorytmem jest Algorytm Euklidesa, oparty o następujący prosty fakt.

Twierdzenie 2.1.7 Jeśli a i b są dodatnimi liczbami całkowitymi przy $a > b$, wtedy $gcd(a, b) = gcd(b, a \bmod b)$. Algorytm Euklidesa składa się z sekwencji następujących dzieleń Wtedy

$$\begin{aligned}
a &= q_1 b + r_1, & 0 < r_1 , b \\
b &= q_2 r_1 + r_2 & 0 < r_2 , r_1 \\
r_1 &= q_3 r_2 + r_3, & 0 < r_3 < r_2 \\
&\vdots \\
r_{n-2} &= q_n r_{n-1} + r_n, & 0 < rn < r_{n-1} \\
r_{n-1} &= q_{n+1} r_n + 0,
\end{aligned}$$

$$a > b > r_1 > r_2 > \bullet\bullet\bullet > r_n > 0)$$

największy wspólny podzielnik to

$$gcd(a, b) = gcd(b, r_1) = gcd(r_2, r_1) = \blacksquare\blacksquare\blacksquare = gcd(r_n, 0) = r_n.$$

W ten sposób, wynika ,że $gcd(a, b) = r_n$.

Algorytm algorytm, Euklides dla oblcizenia największego wspólnego dzielnika dwóch liczb całkowitych

---

Dane wejściowe: dwie nieujemne liczby całkowite a i b przy czym $a > b$.

Dane wyjściowe: największy wspólny podzielnik dla a i b.

1. Podczas gdy $b \neq 0$ zrób co następuje:

   1.1 Ustaw $r \leftarrow a \bmod b, a \leftarrow b, b \leftarrow r$.

2. Zwraca (a).

Tabela 2.2: *Algorytm Euklidesa*

Twierdzenie 2.1.8 Powyższy algorytm ma czas uruchomienia $O((l\ nn)^2)$ operacji bitowych.

Algorytm Euklidesa może być rozszerzony tak aby nie tylko dostarczał największego wspólnego podzielnika d z dwóch liczb całkowitych a i b, ale również liczb całkowitych x i y spełniających warunek $ax + by = d$.

Algorytm Rozszerzony Algorytm Euklidesa

Dane wejściowe: dwie nie ujemne liczby całkowite a i b takie ,że $a > b$.
Dane wyjściowe: $d = gcd(a, b)$ i liczby całkowite x, y takie ,że $ax + by = d$.

1. Jeśli $b = 0$ wtedy ustaw $d \leftarrow a$, $x \neq 1$, $y \leftarrow 0$ I zwraca(d, x, y).
2. Ustaw $x_2 \leftarrow 1, x_1 \leftarrow 0, y_2 \leftarrow 0, y_1 \leftarrow 1$.
3. Gdy $b > 0$ zrób co poniżej :
   3.1 $q \leftarrow$ [a/b]J,r $\leftarrow a-qb, x \leftarrow x_2 - gx_i, y \leftarrow y_2 - qy_1$.
   3.1 a $\leftarrow$ b,b $\leftarrow$ r, $x_{2 \leftarrow x1,}$ $x_1 \leftarrow$ x , $y_2 \leftarrow y_1$, i $y_1 \leftarrow$ y.
4. Ustaw d $\leftarrow$ a, x $\leftarrow x_2$, y $\leftarrow y_2$, i zwraca(d, x, y).

Tabela 2.3: *Rozszerzony algorytm Euklidesa*

Twierdzenie 2.1.9 Rozszerzony algorytm Euklidesa ma czas działania $O((lnn)^2)$operacji bitowych

Ponieważ algorytm Euklidesa oblicza największy wspólny podzielniki,może być używany do określenia czy dodatnia liczba całkowita a < n ma odwrotność mnożenia (będzie omówiona w sekcji 2) modulo n. Jednak nie oblicza wartości odwrotności mnożenia.

2.1.3 Kongruencje

Niech n będzie dodatnia liczbą całkowitą.

Definicja 2.1.8 Jeśli a i b są liczbami całkowitymi, wtedy mówimy,że a jest *kongruencją dla b modulo n,* zapisane jako $a = b \pmod n$, jeśli n dzieli się przez $(a - b)$. Liczba całkowita n nazywana jest modułem kongruencji.

Twierdzenie 2.1.10 *(właściwości kongruencji)* Dla wszystkich $a, a_1, b, b_1, c \in Z$, wtedy poniższe to prawda.

1. $a = b \pmod n$ jeśli I tylko jeśli a i b mają taką samą resztę kiedy dzielone są przez n.
2. *(zwrotność)* $a \equiv a \pmod n$.
3. *(symetria)* Jeśli $a \equiv b \pmod n$, wtedy $b \equiv a \pmod n$.
4. *(przechodniość)* Jeśli $a \equiv b \pmod n$, i $b \equiv c \pmod n$, wtedy $a \equiv c \pmod n$.
5. Jeśli $a \equiv a_1 \pmod n$, i $b \equiv b_1 \pmod n$, wtedy $a + b \equiv a_1 + b_1 \pmod n$ a $ab \equiv a_1 b_1 \pmod n$

Definicja 2.1.9 *Całkowite modulo n,* oznaczone Z_n, jest to zbiór *(równoważna klasa liczb całkowitych)* $\{0, 1, 2, \ldots, n - 1\}$. Dodawanie, odejmowanie i mnożenie Z_n wykonuje się modulo n.

Przykład 2.1.6 $Z_{25} = \{ 0, 1, 2, \ldots, 24\}$. W Z_{25}, 13 + 16 = 4, ponieważ 13 + 16 = 29 ≡ 4 *(mod* 25). Podobnie, 13 • 16 = 8 w Z_{25}.

Twierdzenie 2.1.11 *(Twierdzenie chińskie o resztach, CRT)* Jeśli liczby całkowite $n1, n_2, \ldots, n_k$ są podwójnie relatywnie pierwsze, wtedy system jest jednocześnie kongruencjami

$$x \equiv a_1 \ (mod\ n_1)$$

$$x \equiv a_2 \ (mod\ n_2)$$

.

.

.

$$x \equiv a_k \ (\text{mod } n_k)$$

ma jednoznaczne rozwiązanie modulo $n = n_1 n_2 \ldots . n_k$.

Przykład 2.1.7 Para kongruencji $x \equiv 3$ *(mod 7), x* ≡ *7 (mod* 13) ma jednoznaczne rozwiązanie $x \equiv 59 \ (\text{mod } 91)$.

Twierdzenie 2.1.12 Jeśli $gcd(n_1,n_2) = 1$, wtedy para kongruencji $x \equiv a \ (\text{mod } n_1), x \equiv a \ (\text{mod } n_2)$ ma jednoznaczne rozwiązanie $x \equiv a \ (mod\ n_1 n_2)$

Definicja 2.1.10 *Grupa multiplikatywna* z Z_n to $Z^*_n = \{a \in Z_n | gcd(a,n) = 1\}$. W szczególności, jeśli *n* jest liczbą pierwszą , wtedy $Z^*_n = \{a | 1 \le a \le n - 1\}$.

Definicja 2.1.11 Porządek Z^*_n jest zdefiniowany jako liczba elementów w Z^*_n , konkretnie $| Z^*_n |$

Twierdzenie 2.1.13 Niech $n \ge 2$ będzie liczbą całkowitą.

1. *(Twierdzenie Eulera)* Jeśli a ∈ e Z^*_n, wtedy $a^{\varphi(n)} = 1 \ (\text{mod } n)$.
2. Jeśli n jest iloczynem różnych liczb pierwszych, i jeśli $r \equiv s \ (mod\ \varphi(n))$, wtedy $a^r \equiv a^s \ (mod\ n)$dla wszystkich liczb całkowitych *a*. Innymi słowy, kiedy działa taki wykładnik n może być zredukowane modulo $\varphi(n)$.

Definicja 2.1.12 Niech $a, b \in Z_n$. *Odwrotność mnożenia a* modulo *n* jest liczba całkowitą $x \in Z_n$ taką ,że $ax \equiv 1 \ (mod\ n)$. Jeśli takie *x* istnieje, wtedy jest jednoznaczne, a mówimy ,że *a odwracalne,* lub a *jedność;* odwrotność *a* jest oznaczona przez a^{-1}.

Twierdzenie 2.1.14 Niech $a \in Z_n$. Wtedy a jest odwracalne jeśli i tylko jeśli $gcd(a, n) = 1$.

Dowód.
Po pierwsze, jeśli $gcd(a,n)$ było większe niż 1, nie możemy mieć $ab = 1 \ (\text{mod } n)$ dla dowolnego b, ponieważ implikuje to ,że d dzieli się przez $ab - 1$ i w związku tym dzieli się przez 1.

Specjalnym przypadkiem twierdzenia Eulera jest małe twierdzenie Fermata

Twierdzenie 2.1.15 Niech p będzie liczbą pierwszą.

1. *(Twierdzenie Fermata)* Jeśli $gcd(a,p) = 1$, wtedy $a^{p-1} \equiv 1 \ (mod\ p)$.
2. W szczególności, $a^p \equiv a \ (mod\ p)$ dla wszystkich liczb całkowitych *a*.

Definicja 2.1.13 Niech $a \in E\ Z^*_n$. Porządek a jest oznaczony ord(a), jest to najmniejsza dodatnia liczba całkowita *t* taka ,że $a^t \equiv 1 \ (\text{mod } n)$.

Twierdzenie 2.1.16 Jeśli porządek $a \in Z^*_n$ to t, a $a^s \equiv 1 \ (\text{mod } n)$, wtedy t dzieli się przez s. W

szczególności, t | φ(n).

Definicja 2.1.14 Niech a ∈ Z^*_n. Jeśli porządek z *g* to φ(n), wtedy mówimy,że g jest *generatorem* lub *elementem pierwotnym* z Z^*_n. Jeśli Z^*_n ma generator, wtedy mówimy ,że Z^*_n jest *cykliczny.*

Twierdzenie 2.1.17 *(właściwości generatorów* Z^*_n*)*

1. Z^*_n ma generator jeśli I tylko jeśli n= 2, 4,p^k lub $2p^k$, gdzie p jest nieparzystą liczbą pierwszą a $k \geq 1$. W szczególności jeśli p jest liczbą pierwszą, wtedy Z^*_n ma generator.
2. Jeśli g jest generatorem Z^*_n, wtedy Z^*_n = $\{a^i \pmod n | 0 \leq i \leq \varphi(n) — 1\}$.
3. Przypuśćmy ,że g jest generatorem Z^*_n. Wtedy $b = g^i \pmod n$ jest również generatorem Z^*_n jeśli i tylko jeśli $gcd(i, \varphi(n)) = 1$. To oznacza ,że jeśli Z^*_n jest cykliczne, wtedy liczba generatorów to $\varphi(\varphi(n))$.
4. g∈ Z^*_n jest generatorem jeśli i tylko jeśli $g^{\varphi(n)/p} \neq 1 \pmod n$ dla każdego pierwszego dzielnika p z φ(n).

Definicja 2.1.15 Niech $a \in Z^*_n$. Mówimy ,że *a* to *reszta kwadratowa* modulo *n,* lub *kwadratem* modulo *n,* jeśli istnieje x ∈ Z^*_n takie ,że $x^2 \equiv a \pmod n$. Jeśli żadne takie x nie istnieje, wtedy a jest nazywana *nieresztkowym kwadratem* modulo *n.* Zbiór wszystkich reszt kwadratowych modulo *n* jest oznaczony przez Q_n a zbiór wszystkich nieresztkowych kwadratów jest oznaczone przez $\overline{Q}_n$.

Przykład 2.1.8 *g* = 6 jest generatorem Z^*_{13}. Moc *g* jest wylistowany w poniższej tabelce. Q_{13} = {1,3, 4, 9,12} a $\overline{Q}_{13}$ = {2, 5, 6, 7, 8,11}.

i	0	1	2	3	4	5	6	**7**	8	9	10	11
$a^i \bmod 13$	1	6	10	8	9	2	12	**7**	3	5	4	11

2.1.4 Metoda Wielokrotnych Kwadratów

Metoda powtórzonych kwadratów jest podstawowym obliczeniem w arytmetyce modularnej dla znajdowania $b^n \pmod m$ kiedy zarówno m jak i *n* są bardzo duże. Jest zmyślny sposób wykonania tego ,co jest dużo szybsze niż powtarzanie mnożenia b przez samą siebie. Powiązany algorytm jest następujący:

Niech $n_0, n_1, \ldots, n_{k-i}$ oznacza cyfry binarne n, to znaczy

$$n = n_0 + 2n_1 + 4n_2 + \ldots\ldots + 2^{k-1}n_{k-1} \qquad (nj = 0 \text{ lub } 1).$$

Wtedy , obliczamy jak następuje :

$$b^2, (b^2)^2 = b^4, (b^4) = b^8, \ldots..(b^{2k-2})^2 = b^{2k-1}$$

Co daje

$$b^n = b^{n0+2n1+4n2+\ldots.2k-1nk-1}$$
$$= b^{n0} * b^{2n1} * b^{4n2} \ldots. b^{2(k-1)nk-1}$$
$$= b^{n0} * (b^2)^{n1} * (b^4)^{n2} \ldots. (b^{2k-1})^{nk-1}$$

Ponieważ $b^2,(b^2)^2 = b^4,(b^4) = b^8,\ldots..(b^{2k-2})^2 = b^{2k-1}$ jest wyliczane przedtem b^n (mod m) może być łatwo obliczone.

Przykład 2.1.9 Przypuśćmy ,że chcesz obliczyć 432^{678} (mod 987). Podstawowa sztuczka to zacząć pracę z liczbą i jej kwadratami:

$$432^2 = 186624 = 81 \quad 432^4 = 81^2 = 639 \quad 432^8 = 639^2 = 690 \ldots \; 432^{512} = 858$$

Since 678 = 512 + 128 + 32 + 4 + 2,

$$432^{678} = (81)(639)\ldots (858) = 204 \qquad \text{(Mam nadzieję!)}$$

Obliczenia z wykładnikami obejmuje niezbyt wiele mnożeń. Jeśli liczby mają kilkaset cyfr, konieczne jest zaprojektowanie specjalnego podporgramu do wykonania mnożeń.

Pomysł poza szybkim potęgowaniem jest taki,że jeśli wykładnik to 2 ,wtedy możemy potęgować przez sukcesywne podnoszenie do kwadratu:

$$x^8 = ((x^2)^2)^2 ,$$

$$x^{256} = (((((((x^2)^2)^2)^2)^2)^2)^2)^2 .$$

Jeśli wykładnik nie jest 2, wtedy używamy jego binarnej reprezentacji, jaką jest suma potęg 2;

$$x^{291} = x^{256} \text{ x } x^{32} \text{ x } x^2 \text{ x } x^1.$$

Dlatego też podnoszenie x do potęgi n wymaga tylko około log n działań.

2.1.5 Symbole Legendre'a i Jacobiego

Symbol Legendre'a jest użytecznym narzędziem dla śledzenia czy lub nie liczba całkowita *a* jest resztą kwadratową modulo a liczby pierwszej *p.*.

Definicja 2.1.16 Niech *p* będzie nieparzystą liczbą pierwszą a *a* liczbą całkowitą. *Symbol Legendre'a* $\frac{a}{p}$ jest zdefiniowany

$$(\frac{a}{p}) = \begin{cases} 0 & \text{jeśli pla} \\ 1 & \text{jeśli a} \in Q_p \\ -1 & \text{jeśli a} \in \overline{Q}p \end{cases},$$

Twierdzenie 2.1.18 *(właściwości symbolu Legendre'a)* Niech *p* będzie liczbą pierwszą a $a, b \in$ Z. Wtedy symbol Legendre'a ma następujące właściwości:

1. $(\frac{a}{p}) \equiv a^{(p-1)/2} \pmod{\overline{p}}$. W szczególności, $(\frac{1}{p}) = 1$ a $(\frac{-1}{p}) = -1^{(p-1)/2}$ .W związku z tym $-1 \in Q_p$ jeśli $p \equiv 1 \pmod 4$, a $-1 \in Q_p$ jeśli $p \equiv 3 \pmod 4$

2. $(\frac{ab}{p}) = (\frac{a}{p})(\frac{b}{p})$. W związku z tym jeśli $a \in Z^*_p$ wtedy $(\frac{a2}{p}) = 1$

3. Jeśli $a \equiv b \pmod p$, wtedy $(\frac{a}{p}) = (\frac{b}{p})$

4. $(\frac{2}{p}) = (-1)^{(p2-1)/8}$ Wtedy $(\frac{2}{p}) = 1$ jeśli $p \equiv 1$ lub 7 (mod 8), a $(\frac{2}{p}) = -1$ jeśli $p \equiv 3$ lub 5 (mod 8)

5. (*prawo wzajemności kwadratów)* Jeśli q jest nieparzystą liczbą pierwszą różną od p, wtedy

$$(\frac{p}{q}) = (\frac{q}{p})(-1)^{(p-1)(q-1)/4}$$

Innymi słowy $(\frac{p}{q}) = (\frac{q}{p})$ chyba ,że p i q są przystającymi do 3 modulo 4, a wtedy $(\frac{p}{q}) = -(\frac{q}{p})$

Symbol Jacobiego jest uogólnieniem symbolu Legendre'a dla liczb całkowitych *n* które są nieparzyste ale niekoniecznie pierwsze.

Definicja 2.1.17 Niech $n \geq 3$ będzie nieparzysta z rozkładem na czynniki pierwsze $n = p_1^{e1} p_2^{e2} \ldots p_k^{ek}$ Wtedy

Symbol Jacobi'ego $(\frac{a}{n})$ jest zdefiniowany

$$(\frac{a}{n}) = (\frac{a}{p1})^{e1}(\frac{a}{p2})^{e2} \ldots (\frac{a}{pk})^{ek}$$

Twierdzenie 2.1.19 *(właściwości symbolu Jacobi'ego)* Niech $m \geq 3$, lub $n \geq 3$, będą nieparzystymi liczbami całkowitymi, a $a, b \in Z$. Wtedy symbol Jacobi'ego ma następujące właściwości:

1. $(\frac{a}{n}) = 0, 1$ lub -1 . Ponadto $(\frac{a}{n}) = 0$ jeśli i tylko jeśli gcd(a,n) $\neq 1$

2. $(\frac{ab}{n}) = (\frac{a}{n})(\frac{b}{n})$. W takim wypadku jeśli Z^*_n , wtedy $(\frac{a2}{n}) = 1$

3. $(\frac{a}{mn}) = (\frac{a}{m})(\frac{b}{n})$

4. Jeśli a ≡ b (mod n), wtedy $(\frac{a}{n}) = (\frac{b}{n})$

5. $(\frac{1}{n}) = 1$

6, $(\frac{-1}{n}) = (-1)^{(n-1)/2}$. Zatem $(\frac{-1}{n}) = 1$ jeśli n ≡ 1 (mod 4), a $(\frac{-1}{n}) = -1$ jeśli n ≡ 3 (mod 4).

7. $(\frac{2}{n}) = (-1)^{(n2-1)/8}$. Zatem () = 1 jeśli n ≡ 1 lub 7 (mod 8), a $(\frac{2}{n}) = -1$ jeśli n ≡ 3 lub 5 (mod 8)

8. $(\frac{m}{n}) = (\frac{n}{m})(-1)^{(m-1)(n-1)/4}$. Innymi słowy $(\frac{m}{n}) = (\frac{n}{m})$ chyba ,że zarówno m jak i n są kongruencjami dla 3 modulo 4. w tym przypadku $(\frac{m}{n}) = -(\frac{n}{m})$

2.2 Grupy

Ta sekcja dostarcza omówienia podstawowej algebry obiektów i ich właściwości.

Definicja 2.2.1 *Działanie binarne* * na zbiorze *S* jest odwzorowaniem *S* x *S* do *S*. To znaczy ,że * jest zasadą która przyporządkowuje każdą uporządkowaną parę elementów z *S* elementowi *S*.

Definicja 2.2.2 *Działanie grupowe (G,* *) składa się ze zbioru *G* z działaniem binarnym * na *G* spełniającym następujące trzy aksjomaty.

1. Grupa jest *łączna.* To znaczy,że *a* * *(b* * c) = (a * *b)* * c dla wszystkich $a, b, c \in G$.
2. Jest element 1 ∈ G, nazywany *elementem neutralnym,* takim,że a * 1 = 1 * a = a dla wszystkich $a \in G$.
3. Dla każdego $a \in G$ istnieje element $a^{-1} \in G$, nazywany *odwrotnością a,* taki ,że $a * a^{-1} = a^{-1} * a = 1$.

   Grupa G jest *abelowa* (lub p*rzemienna)*jeśli
4. a * b = *b* * a dla wszystkich $a, b \in G$.

Definicja 2.2.3 Grupa G jest *skończona* jeśli |*G*| jest skończone. Liczba elementów w skończonej grupie jest nazywana *następstwem.*

Definicja 2.2.4 Grupa G jest *cykliczna* jeśli jest element $g \in G$ takie ,że dla każdego $b \in G$ jest liczba całkowita i z $b = g^i$ Taki element g nazywamy *generatorem* G.

Twierdzenie 2.2.1 Jeśli *G* jest grupą i $a \in G$, wtedy zbiór wszystkich potęg *a* formuje cykliczną podgrupę *G,* nazywaną podgrupą wygenerowaną przez a i oznaczoną (a).

Twierdzenie 2.2.2 Niech *G* będzie grupą , i niech $a \in G$ będzie elementem skończonego następstwa t. Wtedy | (a) |, rozmiar podgrupy wygenerowanej przez *a,* jest równy t.

Przykład 2.2.1 Rozważmy grupę multiplikatywną $a \in Z^*_{19} = \{1, 2,\ldots , 18\}$ o następstwie 18. Grupa jest cykliczna, a generator $g = 2$. Podgrupy z $a \in Z^*_{19}$, i generatory są wylistowane w poniższej tabelce

Podgrupa	Generatory	Porządek
{1}	1	1
{1,18}	18	2
{1,7,11}	7,11	3
{1,7,8,11,12,18}	8,12	6
{1,4,5,6,7,9,11,16,17}	4,5,6,9,16,17	9
{1,2,3,… ,18}	2,3,10,13,14,15	18

Tabela 2.4: *Podgrupa* Z^*_{19}.

2.3 Pierścienie

Definicja 2.3.1 Pierścień *(R,* +, x) skład się ze zbioru *R* z dwoma działaniami binarnymi arbitralnie oznaczonymi + (dodawanie) i x (mnożenie) w *R,* spełniającymi poniższe aksjomaty.

1. *(R,* +) jest grupą abelową z elementem neutralnym oznaczonym 0.
2. Działanie x jest łączne. To znaczy *a* x *(b* x c) = *(a* x *b)* x c dla wszystkich $a, b, c \in R$.
3. Jest multiplikatywny element neutralny oznaczony 1, $1 \neq 0$, taki ,że $1 \times a = a \times 1 = a$ dla wszystkich $a \in R$.
4. Działanie x jest *rozdzielne* w stosunku do +. To znaczy, *a* x *(b* + c) = *(a* x *b)* + *(a* x c) a *(b* + c) x a = (b x a) + (c x a) dla wszystkich $a, b, c \in$ R.

Pierścień jest *pierścieniem przemiennym* jeśli a x b = b x *a* dla a, b ∈ R.

Definicja 2.3.2 Element a z pierścienia R jest nazywany *jednostkowym* lub *elementem odwracalnym* jeśli jest element b ∈ *R* taki ,że $a \times b = 1$.

2.4 Pola

Definicja 2.4.1 *Pole* jest to pierścień przemienny w którym wszystkie nie zerowe elementy mają odwrotności multiplikatywne.

Twierdzenie 2.4.1 Z_n jest polem (zgodnie ze zwykłymi operacjami dodawania i mnożenia modulo *n)* jeśli i tylko jeśli n *n* jest liczbą pierwszą. Jeśli *n* jest liczbą pierwszą, wtedy Z_n ma charakterystykę *n.*

Twierdzenie 2.4.2 Jeśli charakterystyka *n* pola nie jest 0, wtedy *n* jest liczbą pierwszą.

2.4.1 Pola skończone

Definicja 2.4.2 *Pole skończone* jest to pole *F* które zawiera skończoną liczbę elementów. *Następstwo F* jest to liczba elementów w F.

Twierdzenie 2.4.3 *(istnienie i jednoznaczność pól skończonych)*

1. Jeśli *F* jest polem skończonym, wtedy *F* zawiera p^m elementów dla pewnej liczby pierwszej *p* i liczby całkowitej m≥ 1.

2. Dla każdej liczby pierwszej o potędze następstwa p^m, jest jednoznaczne (aż do izomorfizmu) pole skończone o porządku p^m . Pole to jest oznaczone F_{pm}, lub czasami $GF(p^m)$.

Twierdzenie 2.4.4 Jeśli F_q jest polem skończonym o następstwie $q = p^m$,*p* jest liczbą pierwszą, wtedy charakterystyka F_q to *p.* Ponadto, F_q zawiera kopię Z_p jako podpola. W takim razie F_q może być rozpatrywane jako rozszerzone pole Z_p stopnia m.

Twierdzenie 2.4.5 *(podpola pola skończonego)* Niech F_q będzie polem skończonym o następstwie $q = p^m$. Wtedy każde podpole F_q ma następstwo p^n, dla każdego *n* które jest dodatnim dzielnikiem m. I odwrotnie , jeśli *n* jest dodatnim podzielnikiem m, wtedy jest dokładnie jedno podpole F_q o następstwie p^n; element a ∈ F_q jest w podpolu F_{pn} jeśli i tylko jeśli $a^{pn} = a$.

Definicja 2.4.3 Nie zerowe elementy F_q z grupy zgodnej z mnożeniem nazywana jest *grupą multiplikatywną* F_q, oznaczona F^*_q.

Twierdzenie 2.4.6 F^*_q jest grupą cykliczną o następstwie *q* — 1. W zawiązku z tym $a^q = a$ dla wszystkich $a \in F_q$.

Propozycja 2.4.1 Następstwo dowolnego $a \in F^*_q$ dzieli się przez q — 1.

Dowód.
Dla $a^{q-1} = 1$ niech d będzie następstwem *a,* tzn. najmniejszą dodatnią potęgą która daje 1. Jeśli *d* nie dzieli się przez *q* — 1, możemy znaleźć mniejszą dodatnią liczbę r, reszta kiedy

$$q - 1 = bd + r, \quad gdzie\ 1 \le r < d$$

jest dzielona przez *d*- takie ,że

$$a^r \bullet a^{bd} = a^{q-1} = 1.$$

Ale to przeciwstawia się minimalizacji *d.* To konkluzja dowodu.

Definicja 2.4.4 Generator grupy cyklicznej F^*_q jest nazywany *elementem pierwotnym* lub *generatorem* F_q.

Twierdzenie 2.4.7 Jeśli $a, b \in F_q$, pole skończone o charakterystyce p, wtedy

$$(a + b)^{pt} = a^{pt} + b^{pt} \text{ dla wszystkich } t \geq 0.$$

2.5 Znajdowanie elementów pierwotnych w Z_p

W wielu protokołach dystrybucji klucza publicznego opartych o Diffie-Hellmana, konieczne jest znalezienie elementu elementarnego $g \in Z_p$, gdzie *p* jest liczbą pierwszą. Nie jest to zbyt trudne do zrobienia jeśli znany jest rozkład na czynniki pierwsze *p — 1*. Dla celów pozostałej części sekcji załóżmy,że rozkład na czynniki pierwsze *p* — 1 to

$$p\text{-}l = p_i^{ei} {}_{P2}^{e2} \text{---} p_k^{ek}$$

gdzie $p_1, p_2, \bullet\bullet\bullet, p_k$ są różnymi liczbami pierwszymi.

Najpierw, przedstawię lemat ,który dostarcza metody określania czy dany elelement $g \in Z^*_p$ pierwotny.

Lemat 2.5.1 Przypuśćmy,że p jest liczbą pierwszą, a rozkład na czynniki pierwsze *p— 1* jest podany powyżej. Wtedy $g \in Z^*_p$ jest pierwotny.

$$g^{(p-1)/pj} \not\equiv 1 \pmod p$$

for $1 \leq j \leq k$.

Dowód.

Niech *d* oznacza prządek *g.* Wiemy,że d jest dzielnikiem p -1 i *g* jest pierwotne jeśli i tylko jeśli *d = p — 1.*

Najpierw ,przypuśćmy,że $g^{(p-1)/pj} \equiv 1$ *(mod p)* dla *j.* Wtedy $d \leq (p — 1)/pj$, więc z pewnością $d \neq p - 1$.

I odwortnie, przypuśćmy,że $g^{(p-1)/pj} \not\equiv 1$ *(mod p)* for $1 \leq j \leq k$. Przypuśćmy ,że $d \neq p - 1$. Ponieważ *d* jest dzielnikiem *p — 1* i *d < p-1,* istnieje liczba pierwsza p_j *(1 ≤ j ≤ k)* takich ,że p_j jest dzielnikiem (p - *1)/d.* Ale implikuje to,że d jest dzielnikiem (p — *1)/*p_j. Dlatego wygląda to tak

$g^{(p-1)/pj} \equiv g^d \equiv 1 \pmod p$

co jest przeciwstawnością. To dowodzi ,że *d = p* — 1, jak żądaliśmy.

Teraz, kiedy mamy efektywną metodę określania czy dany element g jest pierwotny, jak przejść do znajdowania elementów pierwotnych? Może to być zrobione całkiem łatwo za pomocą algorytmu LasVegas, przez wybór wartości losowych dla *g* i ich testowanie, dopóki pierwiastek pierwotny nie zostanie znaleziony. Skuteczność takiego podejścia zależy od prawdopodobieństwa ,ze element losowy $g \in Z^*_p$ jest pierwotny. Dokładnie, jest dokładnie $\varphi(p—l)$ elementów pierwotnych w Z^*_p, więc prawdopodobieństwo ,że element losowy g jest elementem pierwotnym to $\varphi(p-1)/(p-1)$.

Specjalnym przypadkiem jest to kiedy $p = 2q + 1$, gdzie *q* jest liczbą pierwszą. W tym przypadku, uzyskamy następujący wniosek.

Wniosek 2.5.1 Przypuśćmy,że *p* i *q* są liczbami pierwszymi, a $p = 2q + 1$. Przypuśćmy,że $g \in Z^*_p$ a $g \not\equiv \pm 1$ *(mod p).* Wtedy *g* jest elementem pierwotnym jeśli i tylko jeśli $g^{(p-1)/2} \equiv Y \pmod p$.

Dowód.
Zauważmy ,że $g^{(p-1)/q} \equiv g^2$ *(mod p),* i $g^2 \equiv 1$ *(mod p)* jeśli i tylko jeśli $g \equiv \pm 1 \pmod p$. W związku z tym, wyniki są takie jak w ostatnim lemacie.

Faktycznie, Jeśli $g \not\equiv \pm 1 \pmod p$ a *g* nie jest pierwotne, wtedy $g^{(p-1)/2} \equiv 1 \pmod p$.

$$\begin{aligned}(-g)^{(p-1)/2} &\equiv (-1)^{(p-1)/2} g^{(p-1)/2} \pmod p \\ &\equiv (-1)^{(p-1)/2} \pmod p \\ &\equiv -1 \pmod p\end{aligned}$$

Ten wynik zapiszemy jako:

Wniosek 2.5.2 Przypuśćmy ,że p i *q* są liczbami pierwszymi, a $p = 2q + 1$. Przypuśćmy ,że $g \in Z^*_p$ nie jest elementem pierwotnym, a $g \not\equiv \pm 1$ *(mod p).* Wtedy *(—g)* jest elementem pierwotnym.

Oznacza to,że mamy skuteczny deterministyczny algorytm do znajdowania elementu pierwotnego kiedy p i (p — 1)/2 są liczbami pierwszymi.

Nie jest łatwo zweryfikować czy elementy są pierwotne, jeśli nie jest znany rozkład na czynniki pierwsze p-1. Z tego powodu, projektanci kryptosystemu często konstruują p w taki sposób ,że rozkład na czynniki pierwsze p-1 jest znany. Na przykład, często jest pożądane zaimplementowanie kryptosystemu w Z_p, gdzie $p = 2q + 1$ a p i *q* są liczbami pierwszymi. Jedne z powodów zrobienia tego jest taki ,że zapewnia to ,że system nie będzie nieodporny na ataki Pohliga-Hellmana problemem logarytmu dyskretnego. Aby znaleźć takie p,projektant systemu wybierze losową wartość nieparzystą *q,* i przetestuj zarówno p i $p = 2q + 1$ dla pierwszości używając jednego z probabilistycznych testów pierwszości. Jeśli albo p albo q okażą się liczbami złożonymi, wtedy wybierana jest kolejna losowa wartość *q* i proces jest powtarzany.

Inny przykład, kilka protokołów zaimplementowanych w Z_p gdzie p-1 ma pierwszy dzielnik *q* określonego rozmiaru. Dogodną realizacją takiego systemu będzie wzięcie $p = 2qr + 1$, gdzie p, *q* i r są liczbami pierwszymi. Jeśli *q* jest 160 bitową liczbą pierwszą a p jest 512 bitową liczbą pierwszą, wtedy r będzie liczbą pierwszą w przybliżeniu 352 bitową. Tu, projektant systemu zacząłby od wyboru wartości losowych q i r o właściwym rozmiarze, a potem

definiuje $p = 2qr + 1$. Trzy liczby całkowite p, q i r wszystkie będą testowane na pierwszość przy zastosowaniu algorytmu testu pierwszości.

2.6 Algorytm Pohliga-Hellmana

W tej sekcji , skupimy się na rozpracowaniu algorytmu Pohliga-Hellmana. Jest konieczne zrozumienie tego algorytmu ponieważ atak Pohlig-Hellmana jest atakiem poważnym opartym na matematycznych podstawach. Będziemy używać takiej samej notacji jak w ostatniej sekcji: p jest liczbą pierwszą, g jest elementem pierwotnym w Z_p, a $h \in Z^*_p$. Naszym celem jest określenie $a = \log_g h$, gdzie, $0 \leq a \leq p-2$.

Rozkład na czynniki pierwsze potęgi pierwszej p - 1 to

$$p-1 = p_1^{e1} p_2^{e2} \ldots p_k^{ek},$$

gdzie $p_1, p_2, \ldots, p_k$ są różnymi liczbami pierwszymi. Głównym krokiem jest obliczenie $g \pmod{p_i^{ei}}$, $1 \leq i \leq k$ .Więc przypuśćmy,że $q = p_i$ a $e = e_i$ dla i, $1 \leq i \leq k$. Tu pokaże jak obliczyć $x = a \pmod{q^e}$.

Najpierw x jest jest wyrażone jako

$$x = \sum_{i=0}^{e-1} a_i q_i$$

gdzie $0 \leq ai \leq q - 1$ $(0 \leq i \leq e - 1)$. Z tego wynika ,że

$$a = a_0 + a_1 q = \ldots + a_{e-1} q^{e-1} + s q^e,$$

dla pewnej liczby całkowitej s.

Wyliczenie a_0 wynika z następującego faktu

$$h^{\frac{p-1}{q}} \equiv g^{\frac{a_0(p-1)}{q}} \pmod{p} \qquad (2.1)$$

Tu mamy dowód równania (2.1):

$$h^{\frac{p-1}{q}} \equiv (g^a)^{\frac{p-1}{q}} \pmod{p}$$

$$\equiv (g^{ao+a1q+\ldots ae-1qe-1+sqe})^{\frac{p-1}{q}} \pmod{p}$$

$$\equiv (g^{ao+Kq})^{\frac{p-1}{q}} \pmod{p} \qquad \text{(gdzie K jest liczbą całkowitą)}$$

$$\equiv g^{\frac{ao(p-1)}{q}} g^{K(p-1)} \pmod{p}$$

$$\equiv g^{\frac{ao(p-1)}{q}} \pmod{p}$$

Od tego miejsca już prosty sposób na określenie a_0.

Kolejnym krokiem będzie obliczenie $a_1, a_2, \ldots, a_e$-i (if e > 1). Te obliczenia mogą być wykonane z dogodnego uogólnienia Równania (2.1).

Najpierw oznaczamy $h_0 = h$ i

$$h_j = h_g^{-(ao+aiq+-+aj-iqj-1)} (modp),$$

dla $0 \leq j \leq$ e - 1. Użyjemy uogólnionego Równania (2.1):

$$(hj)^{\frac{(p-1)}{qj+1}} \equiv g^{\frac{aj(p-1)}{q}} \pmod{p} \qquad (2.2)$$

(Zauważ ,że kiedy $j = 0$, Równanie (2.2) redukuje Równanie (2.1).) Dowód Równania (2.2) jest taki sam jak dla Równania (2.1):

$$(hj)^{\frac{(p-1)}{qj+1}} \equiv (g^{a-(ao+a1q+...+aj-1qj-1)})^{\frac{(p-1)}{qj+1}} \pmod{p}$$

$$\equiv (g^{ajqj+...+ae-1qe-1+sqe})^{\frac{(p-1)}{qj+1}} \pmod{p}$$

$$\equiv (g^{ajqj+Kjqj+1})^{\frac{(p-1)}{qj+1}} \pmod{p} \qquad \text{(Gdzie } K_j \text{ jest liczbą całkowitą)}$$

$$\equiv g^{\frac{aj(p-1)}{q}} \; g^{Kj(p-1)} \pmod{p}$$

$$\equiv g^{\frac{aj(p-1)}{q}} \pmod{p}$$

Dlatego też, przy danym h_j jest proste do wyliczenia a_j z Równania (2.2).

Dla całkowitego opisu algorytmu, wystarczy zaobserwować ,że h_{j+1} może być wyliczone z h_j co oznacza prostą relację rekurencji od momentu poznania a_j . To wynika z następującej łatwej relacji:

$$h_{j+1} = hjg^{-ajq}j \; (mod \; p). \qquad (2.3)$$

Teraz możemy obliczyć a_0, h_1, a_1, h_2, … , h_{e-1}, a_{e-1} poprzez zastosowanie Równania (2.2) i (2.3).

Rozdział 3 System dystrybucji klucza publicznego

W komunikacji sieciowej, dwóch użytkowników życzy sobie komunikować się wzajemnie poprzez wspólny schemat klucza szyfrowania, mając do podziału tajny klucz kryptograficzny nieznany innym użytkownikom. Jest to co nazywamy problemem dystrybucji klucza.

Ten rozdział rozważa protokoły ustalania klucza i powiązane techniki kryptograficzne ,które dostarczają współdzielenie sekretów pomiędzy dwiema jednostkami,zazwyczaj dla późniejszego użycia jako kluczy symetrycznych dla różnych celów kryptograficznych wliczając w to szyfrowanie, deszyfrowanie i uwierzytelnianie wiadomości .Reszta tego rozdziału jest zorganizowana następująco. Sekcja 1 dostarcza podstawowych materiałów wliczając w to podstawowe definicje i pojęcia , i omawia cele. Sekcja 2 i Sekcja 3 omawiają funkcje jednokierunkowe i protokoły uzgadniania klucza, w oparciu o Protokoły Wymiany Klucza Diffie-Hellmana.

3.1 Definicje

Celem *kryptosystemu* jest *zaszyfrowanie* zrozumiałego *tekstu otwartego* (zwanego również *tekstem niezakodowanym),* zatem stworzenie niezrozumiałego *tekstu zaszyfrowanego* (zwanego również *kryptogramem).* Docelowy odbiorca musi *odszyfrować* tekst zaszyfrowany, zatem odzyskanie tekstu niezakodowanego. Jednak, *podsłuchujący* (zwani *kryptanalitykami)* nie mogą *odszyfrować* tekstu zaszyfrowanego.

Jest kilka sposobów w jaki kryptosystemy mogą być kwalifikowane. Generalnie, kryptosystemy są dzielone na dwie klasy; Kryptosystemy *Klucza Prywatnego* które są poza tymi rozważaniami, I kryptosystemy *Klucza Publicznego,* które będą omawiane w tych rozważaniach.

Kryptosystem jest pięciostopniowy *(M, C,* K, *E, D),* gdzie są spełnione następujące warunki:

- *M* oznacza zbiór nazywany *przestrzenią wiadomości.* Element z *M* nazywany jest *wiadomością tekstu niezakodowanego* lub po prostu *tekstem niezakodowanym.*
- *C* oznacza zbiór nazywany *przestrzenią tekstu zaszyfrowanego.* Element z *C* jest nazywany *tekstem zaszyfrowanym.*
- *K* oznacza zbiór nazywany *przestrzenią klucza.* Element z *K* jest nazywany *kluczem.*
- *E* oznacza zbiór nazywany *transformacją szyfrowania,*

$$C = \{E_k \mid k \in K\}$$
$$E_k : M \rightarrow C, \quad (k \in K)$$

• D oznacza zbiór nazywany *transformacją deszyfrowania,*

$$D = \{D_k \mid k \in K\}$$

$$D_k : C \rightarrow M, \qquad (k \in K).$$

Schemat szyfrowania składa się ze zbioru $\{E_e \mid e \in K\}$ transformacji szyfrowania takiego ,że $E_e(m) = c$ dla wszystkich $m \in M$, i odpowiedniego zbioru $\{D_d \mid d \in K\}$ transformacji deszyfrowania takiego ,że $D_d(c) = m$ z taką właściwością ,że dla każdego $e \in K$ istnieje jednoznaczne $d \in K$ takie ,że $D_e = E_e^{-1}$ co znaczy ,że $D_d(E_e(m)) = m$ dla wszystkich $m \in M$. Do kluczy e i d z poprzednich definicji,odnosimy się jako do *pary kluczy* i czasami oznaczone są przez (e, d). Zauważ,że , e i d mogą być takie same.

3.2 Funkcje jednokierunkowe

Definicja 3.2.1 Funkcja f ze zbioru X do zbioru Y jest nazywana *funkcją jednokierunkową* jeśli $f(x)$ jest łatwo wyliczalna dla wszystkich $x \in X$ ale w gruncie rzeczy wszystkie elementy $y \in Im(f)$ są obliczeniowo niewykonalne dla znalezienia dowolnego $x \in X$ takiego,że $f(x) = y$.

Pojęcie funkcji jednokierunkowej jest podstawą kryptografii klucza publicznego, która ma właściwość, której ktoś kto wie tylko jak szyfrować nie może użyć do zaszyfrowania klucza aby znaleźć funkcję deszyfrującą,ponieważ $f : M. \rightarrow C$ jest łatwo wyliczalny kiedy tylko klucz szyfrujący E_k jest znany, ale jest bardzo trudno w praktyce obliczyć funkcję odwrotną $f^{-1} : C \rightarrow M$. To znaczy, z punkty widzenia realistycznej obliczalności, funkcja f nie jest odwracalna (bez żadnych dodatkowych informacji – klucz deszyfrujący D_k). Taka funkcja f jest nazywana *funkcją pułapkową.* To znaczy, funkcja pułapkowa f jest funkcją bez jakiś dodatkowych pomocniczych informacji poza tym co konieczne do obliczenia f. Inwersja f^{-1} jest łatwa do obliczenia, jednakże ,dla kogoś kto ma tą informację D_k (klucz deszyfrujący). To pojęcie nie powinno być mieszane z funkcjami, które są matematycznie nie-odwracalne z funkcjami jeden do jednego.

Przykład 3.2.1 Prostym przykładem kandydata na funkcję jednokierunkową jest *mnożenie całkowite.* Łatwo jest mnożyć bardzo duże liczby całkowite, podczas gdy nawet najmocniejszy komputer z najlepszymi algorytmami jest niezdolny do rozłożenia na czynniki pierwsze zwykłej dwustucyfrowej liczby, która jest iloczynem dwóch w przybliżeniu równych rozmiarem w rozsądnym czasie.

Przykład 3.2.2 *Liczba pierwsza* jest dodatnią liczbą całkowitą większą niż 1 której dodatnimi dzielnikami całkowitymi są 1 i ona sama. Wybierz liczby pierwsze $p = 48611$, $q = 53993$, z $n = pq = 2624653723$, i niech $X = \{1, 2, 3, \ldots, n - 1\}$. Definiujemy funkcję f w X przez $f(x) = r_x$ dla kazdego $x \in X$, gdzie r_x jest resztą kiedy x^3 jest dzielone przez n. Na przykład, $f(2489991) = 1981394214$ ponieważ $2489991^3 = 5881949859 \cdot n + 1981394214$. Obliczenie

f(x) jest relatywnie proste do zrobienia, ale odwrócenie procedury dużo bardziej trudniejsze; to znaczy,dana reszta do znalezienia wartości *x* która została pierwotnie podniesiona do sześcianu. Ta procedura odnosi się do obliczenia sześcianu modularnego z modulo *n*. Jeśli czynniki z *n* są nieznane I duże, jest to trudny problem; jednkaże jeśli czynniki *p* i *q* z *n* są znane wtedy jest wydajny algorytm dla obliczenia modularnego sześcianu.

3.3 Uwierzytelnianie i Identyfikacja

Protokoły uwierzytelniania zaprojektowano aby dwie lub więcej jednostek komunikowało się przez otwartą sieć uzyskując cele kryptograficzne takie jak poufność ,poprawność danych, uwierzytelnienie jednostki, uwierzytelnienie wiadomości ,określanie tożsamości i uwierzytelnianie klucza.W tej sekcji mamy krótkie wprowadzenie do uwierzytelniania.

Uwierzytelnianie jest jednym z najważniejszych celów bezpieczeństwa informacji. Potrzeba jest łatwego rozpoznawania podpisu jako autentycznego, ale niemożliwe dla kogoś kto nie jest podpisującym.

W krytpografii z kluczem publicznym jest specjalny prosty sposób identyfikacji, w taki sposób ,że nikt nie będzie pretendował do bycia Tobą. Niech A (Alicja) i B (Bob) będą dwoma użytkownikami systemu. Nich f_A będzie transformacją szyfrowania, którą każdy inny użytkownik wysyła wiadomość do Alicji, i niech f_B będzie tym samym dla Boba. Dla uproszczenia, załóżmy,że zbiór *M* wszystkich możliwych wiadomości niezakodowanych i zbiór *C* wszystkich możliwych wiadomości zaszyfrowanych są równe, są takie same dla wszystkich użytkowników. Niech *M* będzie podpisem Alicji (wliczając w to numer identyfikacyjny, czas wysłania wiadomości itp). Nie wystarcza to aby Alicja wysłała do Boba zaszyfrowaną wiadomość $f_B(M)$, ponieważ *każdy* wie jak to zrobić, wiec nie ma możliwości aby rozpoznać czy podpis nie został sfałszowany. Raczej, na początku(lub końcu) wiadomości Alicja wysyła $f_B f_A^{-1}(M)$ .Wtedy Bob odszyfrowuje całą wiadomość, łącznie z tą częścią, przez zastosowanie f_B^{-1}, odkrywając , cały tekst niezakodowany, z wyjątkiem tej małej sekcji, którym jest $f_A^{-1}(M)$. Ponieważ Bob wie,że wiadomość pochodzi od Alicji, stosuje f_A (ponieważ wie,że klucz szyfrujący Alicji jest publiczny), i uzyskuje *M*. Ponieważ nikt oprócz Alicji nie mógł zastosować funkcji f_A^{-1} która jest odwrotnością f_A,wie,że wiadomość pochodzi od Alicji.

3.3.1 Identyfikacja

Definicja 3.3.1 Technika *identyfikacji* lub *uwierzytelniania jednostki* zakłada identyfikację jednej części przez część drugą, która byłą aktywna w czasie tworzenia potwierdzenia.

Zazwyczaj do identyfikacji konieczna jest transmisja danych podczas komunikowania się podmiotów. Obie jednostki są aktywne w komunikacji, dając gwarancję poprawności czasowej.

3.3.2 Uwierzytelnianie źródła danych

Definicja 3.3.2 Technika *uwierzytelniania źródła danych* lub *uwierzytelniania wiadomości* zapewnia jedną część , która odbiera wiadomość o poprawności tożsamości części wysyłającej wiadomość.

Często wiadomość dociera do odbiorcy wraz z dodatkowymi informacjami, tak aby odbiorca mógł określić tożsamość jednostki wysyłającej wiadomość. Taka forma uwierzytelniania zazwyczaj nie gwarantuje poprawności czasowej, ale jest użyteczna w sytuacjach gdzie jedna z części nie jest aktywna w komunikacji.

3.4 Diffie-Hellman a pokrewne Protokoły Uzgadniania Klucza

Protokoły uzgadniania klucza mają różne zalety. W protokołach *przekazania klucza*, klucz jest tworzony przez jeden podmiot i bezpiecznie przekazywany do drugiego podmiotu, podczas gdy w protokołach *uzgadniania klucza* oba podmioty przekazują informację, która jest użyta dla uzyskania współdzielonego tajnego klucza. W protokołach *symetrycznych* te dwa podmioty posiadają a priori wspólne tajne informacje, podczas gdy w protokołach *asymetrycznych* te dwa podmioty współdzielą tylko informacje publiczne, które zostały uwierzytelnione. Ten tekst skupia się na dwukierunkowych protokołach uzgadniania klucza w ustawieniach asymetrycznych.

Stworzenie asymetrycznego protokołu uzgadniania klucza ma burzliwą historię.Przed laty, proponowano liczne protokoły pod kątem pożądanego bezpieczeństwa i wymagań wydajnościowych. Wiele z tych protokołów zostało później uznanych za niedoskonałe , więc zostały zmodyfikowane pod kątem odporności na nowe ataki, lub całkowicie zapomniane. Po serii ataków i modyfikacji, przetrwały tylko te protokoły,które zapewniły potężne bezpieczeństwo publiczne i zapewniały ochronę w realnym życiu.

Ta sekcja skupia się na asymetrycznych protokołach uzgadniania klucza, których bezpieczeństwo jest oparte o trudny problem Diffiego-Hellmana. Następna sekcja mówi o tym dlaczego są stosowane logarytmy dyskretne.

3.4.1 Dlaczego logarytmy dyskretne i Diffie-Hellman?

Prawie wszytko co zapewnia kryptografia z kluczem publicznym,czyli podpisy cyfrowe I wymiana klucza, może być dokonane z RSA i jego wariantami. Jednak, krytposystemy oparte o dyskretne potęgowanie pozostaje interesujące z trzech powodów:

1. **Sprawa patentu** Patent na Diffie-Hellmana wygasł w 1997. Dlatego też każdy zainteresowany zastosowaniem kryptografii z kluczem publiczny, w USA (które są jedynym miejscem gdzie został zastosowany patent) może zaoszczędzić pieniędzy jak również uniknąć negocjacji licencyjnych.

2. **Zalety techniczne** W wielu przypadkach gdzie istnieją porównywalnie funkcjonujące algorytmy, mówimy o polach skończonych I liczbach całkowitych modulo liczby pierwszej *p,* i innych

zastosowaniach złożonej liczby całkowitej *n* o tym samym rozmiarze, przy czym logarytm dyskretny modulo *p* okazuje się czymś trudniejszym niż rozkład na czynniki pierwsze liczby całkowitej *n*.

Inne zalety kryptosystemów logarytmu dyskretnego pochodzą z ich ograniczenia. Głęboko wierzono ,że amerykański Digital Signature Algorithm jest oparty o logarytm dyskretny ponieważ jest trudny do zastosowania do szyfrowania niż oparcie się o RSA (a zatem na rozkładzie na czynniki pierwsze liczb całkowitych). Pomogło to egzekwować regulacje kontroli eksportu silnego szyfrowania bez słabości metod podpisu cyfrowego, które są w mniejszym stopniu kontrolowane. Z drugiej strony wielu ludzi woli algorytm Diffie-Hellman, ponieważ klucz sesyjny generowany jest ulotnie. W najprostszej aplikacji RSA generowania klucza, Alicja tworzy klucz sesyjny i przekazuje go do Boba używając klucza publicznego Boba. Podsłuchujący, który może zmusić Boba do wydania jego klucza prywatnego, może odtworzyć pełny tekst komunikacji między Alicją a Bobem. W odróżnieniu , jeśli Alicja i Bob zastosują Diffie-Hellmana do wygenerowania klucza sesyjnego, zostanie on ziszczony po zakończeniu sesji, i nie przechowa ich komunikacji, i żaden podsłuchujący ani kryptoanalityk nie będą w stanie rozpracować wymienianej przez nich informacji.

3. **Są różnice**. Kryptografowie nauczeni przez doświadczenie, wiedzą ,że nierozsądnie jest wkładać wszystkie jajka do jednego koszyka. Pożądane jest mieć zróżnicowane kryptosystemy, w przypadku złamania jednego. Nieszczęśliwym faktem jest,że logarytmy dyskretne i rozkład na czynniki pierwsze liczb całkowitych są dość bliskie i wiele algorytmów stworzonych dla jednego problemu może być zmodyfikowanych do zastosowania z innym.Dla bezpieczeństwa będzie lepiej aby mieć większą dywersyfikację. Jednak po dwóch dekadach po opublikowaniu pierwszego praktycznego systemu z kluczem publicznym, algorytmy Diffee-Hellmana i RSA, są jedynymi kryptosystemami z kluczem publicznym, cieszącymi się zaufaniem i szeroko stosownymi,opartymi o założone trudności tych samych dwóch problemów tych schematów. Trafiały się próby szukania schematów klucza publicznego opartego o inne zasady lecz zwykle kończyły się niepowodzeniem.

Poniższa notacja będzie używana w całym tym tekście

Alicja, Bob	Uczciwe podmioty.
Lucy	Atakujący (znane również jako atak man-in-the-middle).
p	Liczba pierwsza.
g	pierwiastek pierwotny Z^*_p.
x, y	statyczne klucze prywatne Alicji i Boba.
X, Y	statyczne klucze publiczne Alicji i Boba; $X \equiv g^x \pmod p$, $Y \equiv g^y \pmod p$
r, r'	ulotne klucze prywatne Alicji i Boba

Parametry domeny (p,g) są wspólne dla wszystkich jednostek

3.4.5 Uzgadnianie klucza Diffie - Hellmana

Uzgadnianie klucza Diffie – Hellaman było pierwszym rozwiązaniem problemu dystrybucji klucza, zezwalając dwóm częściom, na współdzielenie tajemnicy poprzez wymianę wiadomości ponad otwartym kanałem. Bezpieczeństwo trudnego do rozwiązania problemu Diffie-Hellmana (DHP) zostanie omówione trochę później

Protokół 1 Ulotna wymiana klucza Diffie-Hellmana

Oboje użytkownicy Alicja i Bob zgadzają się na liczbę pierwszą p I bazowe elementarne

$g \in Z^*_p$

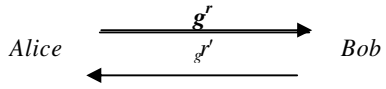
1. Alicja wybiera r losowo $1 \leq r \leq p - 2$.
2. Alicja wylicza $X \equiv g^r \pmod p$ i wysyła go do Boba.
3. Bob wybiera r' losowo, $1 \leq r' \leq p - 2$.
4. Bob wylicza $Y \equiv g^r \pmod p$ i wysyła go do Alicji.
5. Alicja wylicza

$$K = Y^r = (g^{r'})^r = g^{rr'} \pmod p.$$

6. Bob wylicza

$$K' = X^{r'} = (g^r)^{r'} = g^{rr'} \pmod p.$$

Podczas gdy protokołów Diffie-Hellmana zapewnia ukrycie uwierzytelniania klucza przy obecności pasywnego przeciwnika, nie może zapewnić użytecznej usługi przy obecności przeciwnika aktywnego. Ulotna Wymiana Klucza Diffie-Hellmana przypuszczalnie wygląda tak:



Protokół 2 Statyczna wymiana klucza Diffie-Hellmana

Tu zakładamy ,że statyczne klucze publiczne są wymieniane poprzez certyfikaty. $Cert_{Alice}$ oznacza certyfikat klucza publicznego Alicji, zawierający ciąg informacji, które jednoznacznie identyfikują statyczny klucz publiczny Alicji X.

1. Alicja wysyła $Cert_{Alice}$ do Boba.

2. Bob wysyła $Cert_{Bob}$ do Alicji.

3. Alicja oblicza $K = Y^x = (g^y)^x = g^{xy}$ *(mod p).*

4. Bob oblicza $K' = X^y = (g^x)^y = g^{xy}$ *(mod p).*

Ponieważ każda jednostka zakłada ,że posiada autentyczną kopię klucza publicznego innej jednostki, statyczny protokół Diffie-Hellmana zapewnia utajnienie uwierzytelnienia klucza. Główną wadą jest to,że Alicja i Bob obliczają to samo współdzielone $K = K' = g^{xy}$ dla każdego uruchomienia protokołu. Statyczna Wymiana Klucza Diffie-Hellmana wygląda tak:

$$Alice \xrightarrow{g^x eCert_{AHce}} Bob$$
$$Alice \xleftarrow{{}_g y_{eC} ert_{Bob}} Bob$$

Nota 3.4.1 *(sterowanie kluczem Diffie-Hellmana*) Uzgadniania klucza Diffie-Hellmana pozwala każdej części zagwarantować świeżość klucza i wykluczenie sterowania kluczem, używając potęgowania z małym multiplikatywnym porządkiem ograniczony ogólnym porządkiem klucza. Największym zniekształceniem Z_p byłoby wybranie 0 jako pierwotnego wykładnika , dostarczając potęgowania o porządku 1 i multiplikatywnej tożsamości jako klucza wynikowego. Zatem, albo uczestnik może wymusić klucz wynikowy z podzbioru pierwotnego zakresu zbioru.Niektóre warianty Diffie-Hellmana zawierają nieuwierzytelnione wykładniki, które są nieodporne na następujące ataki. Załóż,że $g \in Z^*_p$ gdzie $p = Rq + 1$ (zakładając ,że $R = 2$ a q jest liczbą pierwszą). Wtedy $\beta = g^q = g^{(p-1)/R}$ ma porządek R (β = -I dla $R = 2$). Jeśli Alicja i Bob wymieniają nieuwierzytelnione krótkoterminowe wykładnik g^r i g^r , aktywny przeciwnik może zastąpić je przez $(g^r)^q$ i $(g^{r'})^q$, wymuszając współdzielenia klucza $K = g^{rr'q} = \beta^{rr'}$, który przyjmuje jedną z wartości R (+1 lub -1 dla $R = 2$). K może być zatem stworzony przez wyczerpujące próbowanie wartości R. Bardziej bezpośredni atak wykorzystuje po prostu zastąpienie wymienianego wykładnika przez +1 lub p-1 = -1. Jest to ogólna klasa ataków, której można się przeciwstawić przez uwierzytelnianie wymiany wykładników.

Oboje Alicja i Bob mogą szyfrować wiadomości używając następującej transformacji szyfrowania,

$$c = m^K \ (mod\ p).$$

Żeby odszyfrować, odbiorca musi najpierw znaleźć klucz deszyfrujący $\overline{K}$ przez kongruencję,

$$K \bullet \overline{K} = 1 \ mod\ (p - 1).$$

A potem wyliczyć wiadomość,

$$m = c^K \ (mod\ p).$$

Zauważ ,że , $\overline{K}$ istnieje jeśli i tylko jeśli $gcd(K,p-1) = 1$.

Przykład 3.4.1 Załóżmy ,że modulo $p = 47$ a pierwiastek pierwotny $g = 23$. Przypuśćmy ,że Alicja i Bob wybrali swoje tajne klucze jako $x = 12$ i $y = 33$ Żeby ustalić wspólny tajny klucz K, obliczają oni swoje klucze częściowe :

$$X = g^x = 23^{12} = 27 \ (mod\ 47).$$

$$Y = g^y = 23^{33} = 33 \ (mod\ 47).$$

Po wymianie kluczy częściowych, Alicja i Bob obliczają wspólny tajny klucz,

$$K = Y^x = X^y = 27^{33} = 25 \ (mod\ 47).$$

Znajdują również tajny klucz deszyfrujący $\overline{K}$ używając kongruencji :

$$K \bullet \overline{K} \equiv 1 \quad (mod\ p - 1) \rightarrow \overline{K} \equiv 35 \ (mod\ 46).$$

Teraz jeśli wiadomość $m = 16$, wtedy kryptogram :

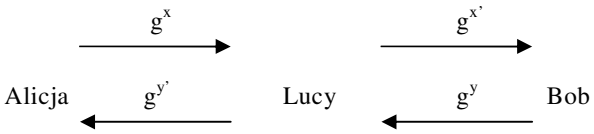
$$c \equiv m^K = 16^{25} = 21 \ (mod\ 47).$$

Odbiorca odtwarza wiadomość tak:

$$m = c^{\overline{K}} = 21^{35} = 16 \ (mod\ 47).$$

Niestety, protokół jest nieodporny na aktywnego przeciwnika używającego *ataku man-in-the-midle.*Atak

man-in-the-middle na protokół Diffie-Hellmana działa następująco. Lucy próbuje przechwycić wiadomość między Alicją a Bobem i zastąpić ją swoją własną wiadomością, jak pokazuje poniższy diagram:



Na końcu protokołu, Alicja przyjęła rzeczywisty tajny klucz $g^{xy'}$ od Lucy, a Bob przyjął tajny klucz $g^{x'y}$ od Lucy. Kiedy Alicja próbuje zaszyfrować wiadomość i wysłać do Boba, Lucy będzie mogła ją odszyfrować ale Bob nie. (Podobna sytuacja wystąpi jeśli Bob wyśle wiadomość do Alice)

Oczywiście, podstawą dla Alicji i Boba jest upewnienie się ,ż wymienili wiadomości między sobą a nie z Lucy. Przed wymianą kluczy, Alicja i Bob mogą wykonać oddzielne protokoły dla upewnienia się co do tożsamości każdego z nich. Ale nie zabezpiecza to przed atakiem aktywnego przeciwnika typu man-in-the-middle jeśli Lucy po prostu pozostanie nieaktywna dopóki Alicja i Bob nie dokonają uwiarygodnienia swoich tożsamości. Powiemy więcej o tym przypadku po wprowadzeniu kilku protokołów opartych ma problemie Diffie-Hellmana.

Problem Diffiego-Hellmana

Problem Diffiego-Hellmana jest blisko związany z *Problemem Logarytmu Dyskretnego.* Jest ważny dla kryptosystemu z kluczem publicznym ponieważ jego oczywista trudność stanowi podstawę bezpieczeństwa wielu schematów kryptograficznych zapewniających Wymianę Klucza Diffie-Hellmana.

Definicja 3.4.1 Problem Diffiego-Hellmana jest następujący : dana jest liczba pierwsza *p,* generator Z^*_p, i elementy g^x *(mod p)* i g^y *(mod p),*znajdź g^{xy} *(mod p).*

Jeśli aktywny przeciwnik w ataku man-in-the-middle ,taki jak Lucy może określić *x* z*X*, lub jeśli może określić y z Y, wtedy może obliczyć *K* dokładnie tak jak Alicja (lub Bob). Oba te wyliczenia są przypadkiem Problemu Logarytmu Dyskretnego.Więc zakładając ,że Problem Logarytmu Dyskretnego w Z^*_p jest trudny do rozwiązania, Wymiana Klucza Diffie-Hellmana jest zabezpieczeniem przed tego typu przypadkiem ataku. Jednak, jest niesprawdzoną hipotezą czy dowolny algorytm,który rozwiązuje protokół Diffiego-Hellmana może również być użyty do rozwiązania Problemu Logarytmu Dyskretnego.

W związku z powyższą uwagą, Problem Diffiego-Hellmana nie jest trudniejszy niż Problem Logarytmu Dyskretnego. Chociaż nie możemy powiedzieć dokładnie jak trudny jest to problem.

3.4.3 Uzgadnianie klucza ElGamala w jednym przebiegu

Uzgadnianie klucza ElGamala jest to wariant Diffie-Hellmana w protokole jednoprzebiegowym z jednostronnym uwierzytelnianiem klucza, zakładającym ,że klucz publiczny odbiorcy jest znany inicjatorowi *a priori.* Ten protokół jest dużo prostszy niż uzgadnianie klucza Diffie-Hellmana , przy czym wykładnik publiczny odbiorcy jest stały I ma weryfikowalne uwierzytelnienie. **Protokół uzgadniania klucza ElGamala (pół certyfikowany Diffie-Hellman)** Obaj użytkownicy Alicja i Bob najpierw uzgadniają liczbę pierwszą *p* i pierwotny element główny $g \in G Z^*_p$.

1. Alicja uzyskuje prawdziwą kopię klucza publicznego Boba (p,g,g^y).
2. Alicja wybiera losowe r, $1 \leq r \leq p - 2$.
3. Alicja oblicza g^r *(mod p)* I wysyła go do Boba.

4. Alice oblicza $K \equiv (g^y)^r \pmod p$.

5. Bob oblicza $K' \equiv (g^r)^y \pmod p$.

Uwaga 3.4.1 *bezpieczeństwo jednoprzebiegowego ElGamala)* Odbiorca w powyższym protokole nie ma potwierdzenia z kim on lub ona współdzielą tajny klucz, ani z kim bezpieczne odświeżanie klucza.Żadna część nie uzyskuj uwierzytelniania podmiotu lub potwierdzenia klucza. Oznacza to,że nawet jeśli przeciwnik w ataku man-in-the-middle zmieni wartość wysłaną przez Alicję, Bob nie może potwierdzić ,że odebrana wartość pochodzi od Alicji.

W kolejnej sekcji omówimy protokoły, które mogą złagodzić wady ulotnego I statycznego protokołu Diffie-Hellmana przez zastosowanie ulotnego i statycznego materiału kluczowego w formowaniu współdzielenia tajemnicy.

3.4.4 Dwu przebiegowe Protokołu Uzgadniania Klucza MTI

W kryptosystemie niezbędna jest zmiana klucza od czasu do czasu. W schemacie Wymiany Klucza Diffiego-Hellmana nie jest tak łatwo zmienić klucz publiczny aby zmienić wspólny klucz, ponieważ kiedy klucz publiczny został zarejestrowany w pliku publicznym ,jest go trudno zmienić z powodu czasochłonności udowodnienia ,że jest się właściwa osobą. Aby zrobić to, jest kilka schematów, które umożliwiają zmianę wspólnych kluczy bez zmiany kluczy publicznych. Te schematy są zaklasyfikowane jako **MTI** stworzone przez Matsumoto, Takashimę i Imai'ego, które są ciekawymi protokołami uzgadniania klucza zmodyfikowanej wymiany klucza Diffie-Hellmana. Przedstawię klika z tych protokołów i rozważę atak man-in-middle na jeden z nich. Zauważ ,że protokoły MTI zaprojektowano do zapewnienia ukrycia uwierzytelniania klucza a nie zapewnienia potwierdzenia klucza a protokół MTI/C0 nie zapewnia także i ukrytego uwierzytelniania klucza.

Przed opisaniem schematów MTI, rozważymy następujące uogólnienia. Każdy protokół MIT składa się z trzech faz :

(a) Faza Rejestracji

Każdy użytkownik [i] wybiera tajną daną X_i I oblicza $Y_i \equiv g^{Xi} \pmod p$ i rejestruje Y_i w pliku publicznym.

(b) Faza Przesłania

Jeśli użytkownik [i] chce współdzielić wspólne dane z innym użytkownikiem [j], [i] przesyła do[j] daną *Zij* wygenerowaną z tajnej liczby losowej R_i i rejestruje daną Y_j i/lub tajną daną X_i i/lub pierwiastek pierwotny *g* z *GF(p)*. Wtedy użytkownik [j] wysyła z powrotem do [i] podobną daną Z_{ji}. Te dane Z_{ij} i Z_{ji} są nazywane " danymi transferowanymi".

(c) Faza Generowania Klucza

Użytkownik [i] składa daną K_{ij} z zaakceptowanego Z_{ji}, uprzednio wygenerowanego X_i i R_i, i rejestruje daną Y_j. Użytkownik [j] składa daną K_{ji} w ten sam sposób

Dane K_{ij} i K_{ji} są takie same i oznaczone przez K,mówimy,że są to "dane współdzielone ". To K będzie używane jako działający klucz.

Tu,zarówno Alicja jak i Bob przechowują w tajemnicy tajny klucz x i y, i rejestrują $X \equiv g^x \ (mod p)$, $Y \equiv g^y \ (mod\ p)$ w pliku publicznym. Liczba pierwsza p i jej główny pierwiastek pierwotny g z Z^*_p są publiczne.

Protokół 1 MTI/Uzgadnianie Klucza A0

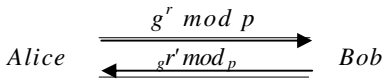
1. Alicja wybiera losową poufność r, $1 \le r \le p - 2$.
2. Alicja oblicza $Z \equiv g^r \ (mod\ p)$ i wysyła ją do Boba.
3. Bob wybiera losową tajność r', $1 \le r' \le p - 2$.
4. Bob coblicza $Z' \equiv g^{r'} \ (mod\ p)$ i wysyła ją do Alicji.
5. Alicja oblicza

$$K = Z'^{x} \bullet Y^r = g^{xr'+yr} \ (mod\ p).$$

6. Bob oblicza

$$K' = Z^y \bullet X^{r'} = g^{xr'+yr} \ (mod\ p).$$

Informacja przekazywana podczas protokołu jest przedstawiona następująco



Protokół 2 MTI/Uzgadnianie Klucza B0

1. Alicja wybiera losową poufność r, $1 \le r \le p - 2$.
2. Alicja oblicza $Z \equiv Y^r = g^{yr} \ (mod\ p)$ i wysyła ją do Boba.
3. Bob wybiera losową tajność r', $1 \le r' \le p - 2$.
4. Bob oblicza $Z' \equiv X^{r'} = g^{xr'} \ (mod\ p)$ i wysyła ją do Alicji.
5. Alicja oblicza

$$K \equiv Z'^{x} \bullet g^r = g^{r+r'} \ (mod\ p).$$

6. Bob oblicza

$$K' \equiv Z^y \bullet g^{r'} = g^{r+r'} \ (mod\ p).$$

Protokół 3 MTI/Uzgadnianie Klucza C0

1. Alice wybiera losową poufność r, $1 \le r \le p - 2$.
2. Alice oblicza $Z = Y^r = g^{yr} \ (mod\ p)$ I wysyła ją do Boba.
3. Bob wybiera losową tajność r', $1 \le r' \le p - 2$.

4. Bob oblicza $Z' \equiv X^{r'} = g^{xr'} \ (mod\ p)$ iwysyła ją do Alicji.

5. Alice oblicza

$$K \equiv Z'^{mr} = g^{rr'} \ (mod\ p).$$

6. Bob oblicza

$$K' \equiv Z^{yr'} = g^{rr'} \ (mod\ p).$$

Protokół 4 MTI/Uzgadnianie Klucza C1

1. Alice wybiera losową tajność r, $1 \leq r \leq p - 2$.

2. Alice oblicza $Z \equiv Y^{rx} = g^{rxy} \ (mod\ p)$ I wysyła go do Boba.

3. Bob wybiera losową tajność r', $1 \leq r' \leq p - 2$.

4. Bob oblicza $Z' \equiv X^{r'} = g^{r'xy} \ (mod\ p)$ I wysyła ją do Alicji.

5. Alicja oblicza

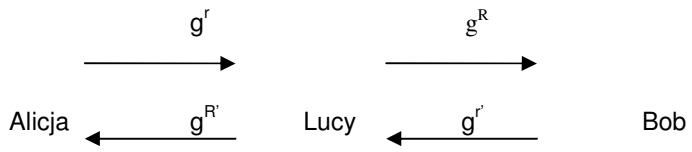
$$K \equiv Z'^{r} = g^{xyrr'} \ (mod\ p).$$

6. Bob oblicza

$$K' \equiv Z^{r'} = g^{xyrr'} \ (mod\ p).$$

Jak omawialiśmy wcześniej, nie jest to ochrona przeciwko aktywnemu atakującemu w ataku typu man-in-the-middle. Oczywiście ,nawet przy tych protokołach ani Alicja ani Bob nie mogą potwierdzić tego z kim wymienili klucze. Jedyną zaletą tych protokołów jest to,że aktywny przeciwnik, taki jak Lucy nie może podsłuchać wiadomości między Alicją a Bobem. Ale również ani Alicja ani Bob nie mogą generować po[prawnych kluczy do szyfrowania lub deszyfrowania wiadomości.

Spójrzmy na bezpieczeństwo protokołu MTI/A0. W ataku man-in-the-middle możliwe jest dla aktywnego przeciwnika ,takiego jak Lucy zmodyfikować wartości jakie Alicja i Bob wysyłają do siebie. Przedstawię jeden typowy scenariusz ,jaki może się wydarzyć:



W tej sytuacji, Alicja i Bob wyliczą różne klucze: Alicja obliczy

$$K \equiv g^{ry+R'x} \pmod{p}$$

podczas gdy Bob obliczy

$$K' = g^{Ry+r'x} \quad (mod\ p).$$

Uwaga 3.4.2 Żaden ze współdzielonych kluczy obliczonych przez Alicję lub Boba nie może być przechwycony przez Lucy, ponieważ wymagają one znajomości o tajnych wykładnikach x i y. Więc chociaż Alicja i Bob obliczają różne klucze (które są użyteczne dla nich oczywiście), żaden z tych kluczy nie może być wyliczony przez Lucy. Innymi słowy, zarówno Alicja jak i Bob zapewniają innych ,że są jedynymi w sieci,którzy mogą obliczyć te klucze. Powiedzmy,że jest to dobre ale niezbyt mocne w wielu przypadkach. Jeśli rozszerzymy ten schemat omówiony powyżej, Alicja lub Bob mogą potwierdzić czy wygenerowany klucz jest prawdziwy czy nie.

Uwaga 3.4.3 *(złożoność obliczeniowa protokołów MTI)* Protokoły A0 i B0 wymagają 3 wykładników dla każdej części, podczas gdy protokoły C0 i C1 wymagają tylko 2. C1 ma dodatkowa zaletę nad B0 i C0 ,ze nie ma konieczności inwersji; jednak te stałe długoterminowe wartości mogą być wyliczone wcześniej.

Rozdział 4 Proponowany protokół dystrybucji klucza publicznego

Przez lata proponowano liczne protokoły oparte o protokół Diffiego-Hellmana, jednak wiele z nich okazywało się nieodpornych na ataki zewnętrzne. Protokół bezpieczny powinien móc stawić opór atakom pasywnym (gdzie przeciwnik próbuje rozpracować protokół jedynie przez obserwcję jednostek korzystających z tego protokołu) i atakom aktywnym (gdzie przeciwnik dodatkowo niszczy komunikację przez podmianę,usunięcie, modyfikację wiadomości), dając ukrytą autoryzację klucza i jego potwierdzenie.

W tym rozdziale skupimy się na projekcie nowego dwuprzebiegowego protokołu. Bezpieczeństwo protokołu jest oparte o trudny do rozwiązania Problem Diffiego-Hellmana. Jak zauważysz jest to rozszerzenie metod omówionych w rozdziale 3 przy MTI. W ostatniej sekcji tego rozdziału rozważymy problemy bezpieczeństwa dla różnych aspektów. Zanim zaczniemy tę sekcję omówimy kilka matematycznych faktów na temat przemienności funkcji potęgowych, które są oparte o trudny obliczeniowo problem logarytmu dyskretnego.

Niech h oznacza nie zerowy element pola skończonego Z_p, gdzie p jest liczbą pierwszą. Zauważ,że funkcja potęgowa dla $GF(p)$ ma następujące właściwości :

1. $(h^x)^y = (h^y)^x = h^{xy}$ (przemienność),
2. $h^x * h^y = h^{x+y}$ (właściwość homomorficzna),
3. $h^x = h^{x(\mathrm{mod}\, p-1)}$

gdzie x i y oznaczają arbitralne liczby całkowite. Przy (3), obserwujemy,że wykładniki należą do zbioru $\{1, 2,\ldots ,p-1\}$. Dla kazdego x, $\bar{x}$ oznacza odwrotność multiplikatywną x modulo $(p-1)$, i istnieje jeśli $gcd(x,p-1) = 1$.

4.1 Opis Protokołu

Ten protokół składa się z 3 faz; Fazy rejestracji, Fazy Przekazywania , i Fazy Generowania Klucza z Fazą Weryfikacji Klucza dla odbiorcy wiadomości Każdy użytkownik,jak Alicja i Bob

wybierają poufne dane x x i y takie ,że $2 \le x, y \le p - 2$, wyliczają $X \equiv g^x \pmod p$, i $Y \equiv g^y \pmod p$ i rejestrują X, i Y w pliku publicznym. Liczba pierwsza p i jej pierwiastek pierwotny g są publiczne. Oczywiście, X i Y również są publiczne. Dla transferu danych i generowania klucza każdy użytkownik , Alicja i Bob powinni wykonać co następuje:

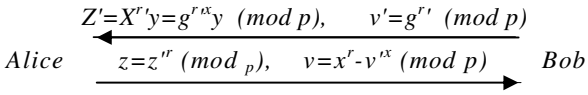
1. Alicja wybiera losową poufność r, $2 \le r \le p - 2$.
2. Alicja oblicza $K \equiv Y^{rx} = g^{rxy} \pmod p$ jako klucz współdzielony.
3. Bob wybiera losową poufność r', $2 \le r' \le p - 2$,taką,żet $gcd(r', p - 1) = 1$. Ponownie Bob znajduje $\overline{r}'$ które jest odwrotnością elementu r' z r'r̄' ≡ 1I $\pmod{p - 1}$.
4. Bob wylicza $Z' \equiv X^{r'y} = g^{r'xy} \pmod p$, ponownie $v' \equiv g^{r'} \pmod p$ i wysyła (Z', v') do Alicji.
5. Alicja wylicza $Z \equiv Z'^{r} = g^{rr'xy} \pmod p$ i ponownie $v = X^r \bullet v'^{x} = g^{x(r+r')} \pmod p$ i wysyła (Z, v) do Boba.
6. Bob wylicza

$$K' \equiv \overline{Z^{r}} = g^{r(r'r'\overline{r}xy} = g^{rxy} \pmod p.$$

i

$$K' = v^y \bullet X^{-r'y} = g^{rxy+r'xy} \cdot g^{-r'xy} = g^{rxy} \pmod p.$$

aby zweryfikować czy $\overline{Z^{r}} = v^y \bullet X^{-r'y}$ Jeśli powyższe równanie jest dobre, wygenerowany klucz jest akceptowany a to oznacza ,że Bob może być pewny,że ma poprawny klucz.Metoda ta jest zilustrowana na poniższym diagramie :



Jak można zauważyć, kierunek strzałek jest odwrócony, co oznacza ,ze kiedy Bob chce odszyfrować wiadomość zaszyfrowaną przez Alicję wysyła jej wartość (Z', v') a otrzymuje (Z, v). Jak wyjaśnię za chwilę, nikt za wyjątkiem Boba nie może odszyfrować tej wiadomości wysłanej przez Alicję

4.2 Właściwości protokołu

Jak mówiłem wcześniej nie ma tu zabezpieczenia przeciwko aktywnemu przeciwnikowi w ataku man-in-the-middle. Ale ten protokół ma następujące zalety :

- Przynajmniej Alicja zapewnia,że ma poprawny klucz.
- Bob może zweryfikować czy ma poprawny klucz czy nie.

- Ten schemat jest nie deterministyczny,ponieważ używa randomizacji w procesie szyfrowania.

Inną właściwością tego schematu jest to,że przeciwnik aktywny nie może oszukać Alicji przy akceptowaniu niepoprawnego klucza jako poprawnego, więc próbuje oszukać Boba. Ale Bob może zweryfikować czy ma poprawny klucz czy nie.

4.3 Aspekty bezpieczeństwa : Ataki

Głównym celem tej sekcji jest naświetlenie delikatnej natury wyżej omówionych protokołów. Problem bezpieczeństwa zostanie przebadany we wszystkich możliwych aspektach. Ataki na protokół Diffiego – Hellmana pochodzą z różnych źródeł, ale generalnie są klasyfikowane następująco:

- **Ataki Denial of Service:** Tu, atakujący będzie próbował powstrzymać Alicję i Boba przed skutecznym wykorzystaniem protokołu. Atakujący może to osiągnąć na kilka sposobów, na przykład przez usunięcie wiadomości jakie Alicja i Bob wysyłają sobie wzajemnie, lub przez zablokowanie tych jednostek niepotrzebnymi obliczeniami lub komunikacji. Prawdopodobieństwo takiego ataku zależy od tego co my wiemy o możliwościach atakującego.Na przykład, jeśli przeciwnik może usuwać i zamieniać wiadomości z publicznego pliku komunikacji, atak denial of service jest niemożliwy do zablokowania.

- **Atak zewnętrzny:** W tym ataku, atakujący próbuje zakłócić protokół (przez, na przykład, dodanie, usuniecie, zastąpienie wiadomości) tak aby uzyskał interesującą go wiedzę (tj. informacje jakich nie może uzyskać przez podgląd wartości publicznych).Przykład takiego ataku to atak typu man-in-the-middle.

- **Atak wewnętrzny:** Jest możliwe,że odbiorca wiadomości w tym protokole tworzy łamliwy protokół uruchomiony w celu spróbowania uzyskania wiedzy uczestnika o tajnym kluczu swojego partnera. Jest to ważny atak jeśli jedna ze stron przechowuje tajny klucz statyczny używany w wielu działających protokołach uzgadniania klucza. Ten atak jest zneutralizowany w tym protokole kiedy Bob chce wygenerować klucz współdzielony.Nawet jeśli Bob używa statycznego klucza tajnego, może usunąć go przez wyliczenie Z^r dla wygenerowania klucza współdzielonego.

4.3.1 Atak man-in-the-middle

(Przypadek 1)

W tym protokole nie ma sposobu aby oszukać Alicję ponieważ sam Alicja jest generatorem klucza współdzielonego. Więc jedyny sposób dla aktywnego przeciwnika w ataku man-in-the-middle to oszukanie Boba jak następuje:

1. Kiedy Bob wysyła swoje (Z', v') do Alicji,aktywny przeciwnik może znać (Z', v') i zastąpić na (Z'', v'') te (Z', v')i wysłać je do Alicji.

$$Z'' \equiv X^{r''w} = g^{r''xw}, \quad v'' \equiv g^{r''} \pmod{p}.$$

2. Alicja oblicza

$$Z = Z''^{r} = g^{rr''xw}, \quad v = X^{r} \blacksquare v''^{x} = {}_{g}<^{r+r''}) \ (mod\ p).$$ i wysyła go do

Boba.

3. Aktywny atakujący oblicza

$$K'' \equiv Z^{\overline{r''}} = g^{rxw} \ (mod\ p).$$

lub

$$K'' = v^{w} \bullet X^{-r''w} = g^{rxw} + {}^{r''w} . g\text{-}^{r''xw} = g^{rxw} \quad (mod\ p).$$

które różni się od rzeczywistej wartości $K \equiv g^{rxy}$. Zatem, nie może wygenerować poprawnego klucza. Tymczasem, Bob oblicza,

$$K' \equiv Z^{\overline{r'}} = g^{\overline{rr'r''xw}} \ (mod\ p),$$

lub

$$K' \equiv V^{y} \bullet X^{-r'y} = g^{rxy+r''xy} . g^{-r'xy} \quad (mod\ p).$$

Ponieważ wartości kluczy są różne Bob rozumie ,że nie ma poprawnego klucza.

(Przypadek 2)

Ten scenariusz ma miejsce kiedy Alicja wysyła *(Z, v)* do Boba

1. Kiedy Alicja wysyła swoje *(Z, v)* do Boba, aktywny przeciwnik może poznać go i podmienić na *(Z", v").*

$$Z" \equiv Z'^{r"} = g^{r'r"xy}, \ v" \equiv W^{r"} \bullet v^{rw} = g^{w(r'+r")} \ (\text{mod p}),$$

gdzie $W \equiv g^{w} \ (mod\ p).$

2. Bob oblicza

$$K' = (Z")^{\overline{r'}} = g^{r"xy} \ (\text{mod p}),$$

lub

$$K' \equiv v"^{y} \bullet X^{-r'y} = {}_{g}{}^{r"wy+r'wy} . g^{-r'xy} \ (\text{mod p}).$$

ponieważ wartość tych kluczy jest inna Bob rozumie ,że nie ma poprawnego klucza.

Nawet w tym przypadku przeciwnik nie może wyodrębnić tajnych informacji z przechwyconej wartości. Ponadto nie może oszukać Boba poprzez podanie mu do akceptacji "niewłaściwego " klucza jako poprawnego. Aby tego dokonać potrzebuje dodatkowych informacji takich jak poufna wartość y. Są różne przypadki, kiedy aktywny przeciwnik próbuje wyodrębnić poufne wartości Alicji i Boba, lub oszukać Boba przez podstawienie mu niepoprawnego klucza,niestety to nie wypali ponieważ sposób w jaki Bob generuje klucz zależy od poufnego klucza y.

4.3.2 Ataki oparte o Teorię Liczb

Poprzedni atak man-in-the-middle , chociaż całkowicie złamał protokół, wymaga od atakującego dużych możliwości. Poniżej podam kilka przypadków, które mogą wystąpić z matematycznego punktu widzenia, połączonych z teorią liczb.

Atak zniekształconą wiadomością

Występują przypadki zniekształceń, przy których nie działa protokół..Na przykład kiedy g^x lub g^y równa się jeden, transferowana dana i klucz współdzielony przyjmują wartość 1.Ponieważ kanał komunikacyjny jest publiczny, każdy może wykryć tę anomalię.na szczęście, taka sytuacja jest niemożliwa ponieważ zarówno x i y są wybierane z $\{2,\ldots,p-2\}$. Zauważ ,że jeśli program komputerowy nie zrozumie ,że g^x, g^y i g^{xy} nie mogą równać się 1, protokół jest nieodporny na złamanie. Te same argumenty dotyczą wartości w postaci $g^{\alpha-(p-1)-x}$ lub $g^{\alpha-(p-1)-y}$, gdzie $a \geq 1$. Wiec łatwo jest zweryfikować ,że g^x i g^y są dodatnimi liczbami całkowitymi mniejszymi niż p - 1 a większymi niż 1.

Jest inny przypadek kiedy przeciwnik może wykryć klucz współdzielony, kiedy $r' = 1$. W tym przypadku $Z' \equiv X^{r'y} = g^{xy}$, a Alicja oblicza $Z \equiv Z^{'r} = g^{rxy} = K$ które jest kluczem współdzielonym.Zauważ ,że taki sam scenariusz zdarzy się kiedy r = 1 ,które w tym przypadku spowoduje wygenerowanie klucza $Z \equiv Z^{'r} = g^{r'xy}$, a atakujący może uzyskać informacje x i naturalnie może znaleźć klucz. Oczywiście,kiedy zarówno r = 1 jak I $r' = 1$, klucz współdzielony to $g^{xy} \equiv K$ który będzie widziany na kanale publicznym. Na szczęście, tak sytuacja jest niemożliwa ponieważ zarówno r jaki r' są wybierane z zakresu $\{2,\ldots,p-2\}$.

Ataki oparte o złożony porządek podgrup

W tym ataku, atakujący może wykorzystać podgrupy, które nie mają dużego porządku liczb pierwszych. Najlepiej zilustrować to przykładem. Przypuśćmy ,że Alicja i Bob wybiorą liczbę pierwszą $p = 2q+1$, gdzie q jest liczbą pierwszą a generator g porządku $p-1 = 2q$. Atakujący może podsłuchać wiadomości g^x i g^y spotęgować je przez q (zastąpi g^x przez g^{xq} i g^y przez g^{yq}.)W protokole Diffie-Hellmana, poufny klucz to g^{xyq} który pozwala atakującemu znaleźć tą wartość przez przeszukiwanie pełne.

Rozważmy atak wewnętrzny Boba używając powyższy atak. Jak wyżej wspomniałem, nawet jeśli Bob użyje q zamiast r dla poznania wartości poufnej Alicji, jest niemożliwe dla niego zrobić to ponieważ po odebraniu $Z \equiv Z^{'r} = g^{rqxy}$ musi wyliczyć $Z^{\overline{q}} \equiv g^{rqqxy} = g^{rxy}$, co oznacza,że wygenerowany klucz nie zależy od klucza sesyjnego Boba .Mimo,że protokół chroni poufny klucz Alicji w ataku wewnętrznym, nauczyliśmy się jednak,że przy takim ataku powinniśmy wybierać g ,które generuje dużą liczbę pierwszą porządku podgrupy lub upewnić się ,że złożony porządek podgrupy nie jest łatwy do złamania. Ponadto, wybierz liczbę pierwszą p takż,że p - 1 zawiera duże współczynniki.

4.3.3 Pozostałe typy ataków

W poprzedniej sekcji rozpatrywaliśmy ataki oparte o strukturę matematyczną. W tej sekcji omówimy inne typy ataków.

- **Przekierowanie wiadomości:** Możliwe jest dla przeciwnika w ataku typu man-in-the-middle przechwycenie i wysłanie wiadomości do kogoś innego niż właściwy odbiorca.Zauważ ,że ten atak może być zastosowany z ulotnym Diffie-Hellmanem a nie innymi protokołami ponieważ atakujący przechwytuje każdy klucz Alicji lub Boba .nawet jeśli atakujący wyśle wiadomość do kogoś, niemożliwe będzie dla niego odszyfrowanie jej.

- **Atak zaszyfrowanym tekstem:** W tym ataku, kryptoalityk przy danych $c_1 = E_1(m_1)$, $c_2 = E_2(m_2)$,….. $c_i = E_i(m_i)$, szyfruje *i* różnych, nieznanych wiadomości tekstu otwartego tym samym nieznanym kluczem. Wywnioskowuje klucz *K* lub, nie mając takiej możliwości wywnioskowuje $m_1, m_2, \ldots, m_i$ tak długo jak to możliwe. Zaproponowany protokół jest zabezpieczony przed tego typu atakami ponieważ liczba losowa r jest używana we współdzielonym kluczu. Ponieważ liczba losowa odświeża klucz dla każdej wiadomości będzie nieosiągalna dla krytpoanalityka dla znalezienia klucza Zauważ ,że nawet jeśli może on znaleźć współdzielony klucz, będzie on nieużyteczny dla kolejnego tekstu.

- **Atak tekstem jawnym:** Kryptoanalityk ma dane $c_1, c_2, \ldots, c_i$ jak powyżej, ale również odpowiednie $m_1, m_2, \ldots, m_i$. Wywnioskowuje on *K* lub stara się wywnioskowywać m_i+1 z nowego tekstu jawnego $c_{i+1} = E_{i+1}(m_{i+1})$.Ten sam scenariusz jak powyżej zdarzy się kryptoanalitykowi ponieważ liczba losowa używana jest w odświeżaniu klucza współdzielonego.Ponadto, robiąc $c_{i+1} = E_{i+1}(m_{i+1})$,potrzebuje pewnych informacji o x i y, które są poufne i nikt z wyjątkiem właściciela ,który je zna.

- **Atak za pomocą wybranego tekstu jawnego:** Kryptoanalityk uzyskał wybraną wiadomość tekstem jawnym $m_1, m_2, \ldots, m_i$, i ma odpowiadającą $c_1 = E_1(m_1)$, $c_2 = E_2(m_2)$,….. $c_i = E_i(m_i)$. Wywnioskowuje on *K* lub, wywnioskowuje m_{i+1} z pewnego nowego tekstu zaszyfrowanego $c_{i+1} = E_{i+1}(m_{i+1})$. W tym ataku kryptoanalityk stawia czoło temu samemu problemowi co powyżej. Jeśli klucz szyfrujący byłby ten sam dla każdego protokołu, uruchomienie go mogłoby być możliwe dla kryptoanalityka znającego pewne informacje o kluczu współdzielonym.Ale niestety, klucz współdzielony odnosi się dla każdego uruchomienia protokołu.

- **Atak za pomocą wybranego tekstu zaszyfrowanego:** Kryptoanalityk uzyskuje wybraną zaszyfrowaną wiadomość $c_1, c_2, \ldots, c_i$, i dany odpowiadającą jej $m_1 = D_1(c_1)$, $m_2 = D_2(c_2), \ldots, m_1 = Di(c_i)$, potwierdzającą ich istnienie. Wywnioskowuje on *K* lub dowolny wydajny algorytm dla obliczenia D_i lub wnioskuje m_{i+1} z pewnego nowego tekstu zaszyfrowanego $c_{i+1} = E_{i+1}(m_{i+1})$ Ten atak jest podobny do ataku wybranym tekstem jawnym z rozważaniem ,że $D_i = K$, i $E_i = K$, gdzie $K \bullet \overline{K} = 1 \pmod{p - 1}$. Zatem , ten sam problem do rozwiązania dla kryptoanalityka.

4.3.4 Względy bezpieczeństwa

W tej sekcji podam pewne zalecenia, które powinny być brane pod uwagę podczas implementacji protokołu Diffie-Hellmana. Większość z tych zaleceń jest opartych na atakach omówionych powyżej.

- **Uwierzytelnianie Parametrów:** Generalna zasada,wszystkie parametry używane w protokołach kryptograficznych powinny być uwierzytelnione. Na przykład przypuśćmy ,że nie uwierzytelniłeś swoich wybranych parametrów, atakujący może oszukać je używając słabych parametrów. Ten typ ataków może być bardzo subtelny i nawet może być pominięty przez najlepszych kryptografów i ekspertów od bezpieczeństwa.

- **Usuwanie poufnych wykładników:** Ważne jest aby usuwać poufne wykładniki, chronić pamięć zapisaną na dysk i zabezpieczyć się przed niepożądanym dostępem do tych wartości.

- **Odświeżanie klucza i poufność doskonała:** W wielu sytuacjach współdzielenie poufnego klucza powinno być często zmieniane, podobnie jak proponowany protokół. Tu są główne powody dlaczego możemy chcieć uzyskać nowe współdzielone klucze.

  1. *Redukcja Ujawnienia* Prawdopodobieństwo ,że dany klucz jest złamany jest mniejsze jeśli nie jest używany zbyt często.
  2. *Poufność* Jeśli starsze klucze szyfrujące zostały usunięte, zaszyfrowane wiadomości nie mogą być dalej deszyfrowane. Dlatego, trzecia część nie może zmontować tylko tekstu zaszyfrowanego (atak za pomocą wybranego tekstu jawnego, i atak wybranym tekstem zaszyfrowanym).

- **Niezależność klucza:** Generalna zasada, zawsze chcemy mieć niezależne klucze. Dokładnie uzyskując jeden poufny klucz nie powinniśmy pomagać atakującemu odkryć inne klucze. Ta właściwość jest nazywana *znane bezpieczeństwo klucza.*

- **Protokół matematyczny**

  1. Sprawa niekonwencjonalnych wiadomości
     - — Upewnij się,że g^x, g^y i g^{xy} nie są równe 1.
     - — Upewnij się,że g^x i g^y są mniejsze niż p- 1 i większe niż 1.
     - — Wybierz $x, y,$ ze zbioru $\{2,\ldots, p - 2\}$.
  2. Bądź ostrożny z porządkiem g
     - — Pierwszy czynnik rozkładu porządku g nie powinien być całkowicie z małych liczb pierwszych.
     - — Podgrupa wygenerowana przez g nie powinna mieć małego porządku podgrupy. Jeśli to możliwe, zbuduj i zastosuj generator, który ma porządek dużych liczb pierwszych.

- **Skuteczność** Generator g powinien być tak mały jak to możliwe aby zredukować koszt modularnego potęgowania.

Rozdział 5 Wnioski

Wyniki naszych rozważań są następujące:

1. Wygenerowany klucz jest zabezpieczony przed atakiem man in the middle.
2. Protokół jest oparty o trudny do rozwiązania problem Diffiego-Hellmana.
3. Jest zdolny do utajnienia uwierzytelniania klucza I potwierdzenia klucza, dostarcza zapewnienia odbiorcy,że on czy ona mogą wyliczyć poprawny klucz.
4. Stosowanie liczb losowych jest nie deterministyczne.
5. Jest zabezpieczony przed atakami omówionymi w tych rozważaniach.
6. Może być chroniony również przeciwko atakom opartych o teorię liczb

Algorytm wymiany klucza Diffiego-Hellmana jest oparty o założenie,że logarytmy dyskretne są trudne do obliczenia. Ta hipotetyczna trudność również jest podstawą dla bezpieczeństwa innych schematów z kluczem publicznym. Podczas gdy było to poważną zaletą w algorytmie logarytmu dyskretnego w ostatnich dwóch dekadach, generalnie logarytm dyskretny jawi się jako trudny. Niestety brak dowodów na jego trudność, więc konieczne są doświadczenia i intuicja jakiego rodzaju parametrów używać dla kryptosystemów.

Omawiany ,proponowany protokół jest bezpieczny a jego bezpieczeństwo opiera się na problemie Diffiego-Hellmana i złożoności jego samego. Protokół ten jest nie detrministyczny, ponieważ stosuje randomizację w procesie szyfrowania.Inną właściwością tego protokołu jest weryfikacja, która może być zrobiona łatwo przez odbiorcę wiadomości. Ten protokół jest rozszerzeniem metod opisanych w Rozdziale 3.

Jeden otwarty problem z tym protokołem jest taki ,że jest jeszcze nieznany, i nie wiadomo czy może być uogólniony do zastosowania z 3 lub więcej użytkownikami. Ten protokół może być użyty również jako protokół jednoprzebiegowy.W tej sytuacji nie będzie zabezpieczony przed większością ataków rozpatrywanych w Rozdziale 4.

Proponowany protokół w tych rozważaniach posiada wiele pożądanych atrybutów bezpieczeństwa.