Laboratorium ochrony danych

Ćwiczenie nr 5

Temat ćwiczenia: Szyfr Rivesta-Shamira-Adlemana (RSA)

Cel dydaktyczny: Poznanie systemu kryptograficznego z kluczem publicznym na przykładzie algorytmu Rivesta-Shamira-Adlemana. Zastosowanie algorytmu do szyfrowania danych (plików), bezpiecznego przesyłania kluczy kryptograficznych oraz uwierzytelniania kryptograficznego. Analiza czasu szyfrowania plików w zależności od ich rozmiaru. Konstrukcja tuneli SSH i VPN.

Wprowadzenie teoretyczne

Algorytm kryptograficzny RSA jest algorytmem wykładniczym z kluczem publicznym. Bezpieczeństwo szyfru RSA opiera się na trudności rozkładu dużej liczby na czynniki pierwsze *p* i *q*. Tekst szyfrowany przedstawia się w postaci liczb w zakresie od 0 do *n*-1.

W celu wygenerowania kluczy kryptograficznych odbiorca kryptogramu losuje dwie liczby pierwsze *p* i *q* oraz liczbę *e*. Iloczyn *n* = *pq* oraz *e* stanowią publiczny klucz szyfrujący. Liczba *e* musi być względnie pierwsza z funkcją Eulera $\phi(n)$. Spełnione są więc zależności:

$$n = pq,$$
$$\Phi(n) = (p-1)(q-1),$$
$$NWP(e, \Phi(n)) = 1.$$

Elementy szyfru C_i odpowiadające wiadomości $M_i \in [0, n-1]$ otrzymujemy z zależności:

$$C_i = M_i^{\,e} \,(\mathrm{mod}\, n)\,.$$

Tajny klucz deszyfrujący zawiera *n* i liczbę *d* spełniającą zależność:

$$de(\mathrm{mod}\,\Phi(n)) = 1.$$

Liczba *d* jest odwrotnością multyplikatywną liczby *e* obliczaną mod $\phi(n)$. Odwrotność istnieje jeśli liczby *e* i $\phi(n)$ są względnie pierwsze. Możliwe jest również wylosowanie liczby *d* i obliczenie jej odwrotności *e*.

Wiadomość M_i z kryptogramu C_i możemy obliczyć z zależności:

$$M_i = C_i^{\,d} \,(\mathrm{mod} n).$$

Funkcje szyfrująca i deszyfrująca są wzajemnie odwrotne. Poprawność algorytmu można pokazać podstawiając C_i do ostatniego równania i przyjmując $ed = t\Phi(n) + 1$ oraz $M_i^{\,\Phi(n)} (\mathrm{mod} n) = 1$, gdzie *t* jest pewną liczbą całkowitą (arytmetyka wykładników *e* i *d* jest arytmetyką modulo $\Phi(n)$):

$$M_i = M_i^{ed} (\mathrm{mod} n) = M_i .$$

Przy wyprowadzeniu wzoru stosuje się Eulerowskie uogólnienie twierdzenia Fermata, tj. dla *a* i *n* względnie pierwszych:

$$a^{\varphi(n)} \left(\mathrm{mod}\, n\right) \equiv 1.$$

W tym przypadku $a = M_i$, $n = pq$, oraz $M_i \in [0, n-1]$, czyli $M_i < n$. Liczba M_i jest względnie pierwsza z iloczynem liczb pierwszych *p* i *q*, a więc NWD(M_i, *n*) = 1 oraz $M_i^{\Phi(n)} (\mathrm{mod} n) = 1$.

Nie znając czynników *p* i *q* liczby *n*, wyznaczenie wartości d klucza deszyfrującego jest bardzo trudne, co gwarantuje bezpieczeństwo szyfru.

Na bezpieczeństwo szyfru wpływa dobór liczb pierwszych *p* i *q*. Powinny to być liczby o długości powyżej stu cyfr. W celu lepszego zabezpieczenia przed rozkładem *n* na czynniki pierwsze zaleca się aby:

- liczby *p* i *q* różniły się o kilka cyfr,
- zarówno *p*-1 jak i *q*-1 miały duże czynniki pierwsze,
- wartość *NWP*(*p*-1,*q*-1) była niewielka.

W realizacji praktycznej algorytmu występuje problem wyboru algorytmu generowania liczb pierwszych i realizacji funkcji szyfrującej. W załączonym programie w celu wygenerowania liczb pierwszych stosuje się generator losowy liczb nieparzystych a następnie sprawdza się czy wylosowana liczba *p* jest liczbą pierwszą, stosując twierdzenie Fermata. Liczba pierwsza spełnia wtedy zależność:

$$x^{p-1} \left(\mathrm{mod}\ p\right) = 1.$$

Funkcję szyfrującą:

$$C = M^e \left(\mathrm{mod}\ n\right)$$

realizuje następujący algorytm numeryczny:

- liczbę *e* przedstawiamy w postaci binarnej $e = e_k, e_{k-1}, ..., e_1, e_0$,
- podstawiamy *C*=1,
- powtarzamy w pętli od *i*=*k* do 0 następujące operacje:
  1. $C = C^2 \left(\mathrm{mod}\ n\right)$,
  2. jeżeli $e_i = 1$ to $C = C * M \left(\mathrm{mod}\ n\right)$.

Można również analizować bity od końca. Wówczas algorytm ma postać:

- liczbę *e* przedstawiamy w postaci binarnej $e = e_k, e_{k-1}, ..., e_1, e_0$,
- podstawiamy *C*=1,
- powtarzamy w pętli od *i*=0 do *k* następujące operacje:
  1. jeżeli $e_i = 1$ to $C = C * M \left(\mathrm{mod}\ n\right)$.
  2. $M = M^2 \ (\mathrm{mod}\ n)$.

W 1994 roku szyfr RSA został złamany. Odczytano zdanie zaszyfrowane przez twórców RSA siedemnaście lat wcześniej. Pracowało nad tym 600 osób na pięciu kontynentach, przez osiem miesięcy, za pomocą sieci Internet.

Algorytmy kryptografii asymetrycznej są wykorzystywane do szyfrowania niewielkich plików oraz wiadomości poczty elektronicznej, a także kluczy sesji dla bezpiecznych systemów komunikacji implementujących algorytmy kryptografii symetrycznej. Znajdują one również zastosowanie w konstrukcji certyfikatów kluczy, systemach podpisów cyfrowych oraz systemach uwierzytelniania kryptograficznego typu „wyzwanie – odpowiedź".

Systemy kryptografii asymetrycznej oraz symetrycznej umożliwiają konstrukcję bezpiecznych kanałów komunikacyjnych – tuneli szyfrowanych, w niezabezpieczonej sieci publicznej, np. sieci Intertnet. Do najbardziej popularnych bezpiecznych kanałów komunikacyjnych należą tunele SSH (ang. Secure Shell) oraz tunele w postaci sieci VPN (ang. Virtual Private Network).

SSH jest protokołem warstwy aplikacji, który służy do zdalnego zarządzania hostami. Protokół ten ustanawia tzw. „bezpieczną powłokę" do komunikacji ze zdalnym serwerem, na którym użytkownik posiada konto. Zasadniczo SSH jest używany do bezpiecznego pozyskiwania i używania zdalnej sesji terminalowej. Połączenie klienta z serwerem jest szyfrowane i można ustawić klienta SSH tak, aby działał jako serwer proxy SOCKS Host. Wówczas, można skonfigurować aplikacje na komputerze klienta, np. takie jak przeglądarka internetowa, aby korzystały z serwera proxy SOCKS. Ruch sieciowy wchodzi do proxy SOCKS, który działa w systemie lokalnym, a klient SSH przekazuje go przez szyfrowane połączenie SSH, nazywane tunelem SSH do serwera zdalnego. W takim przypadku działanie tunelu SSH z punktu widzenia klienta jest podobne do działania sieci VPN, w której cały ruch sieciowy odbywa się przez szyfrowany kanał VPN.

Domyślnym algorytmem umożliwiającym nawiązanie bezpiecznego połączenia SSH jest algorytm RSA, istnieje również możliwość szyfrowania danych za pomocą nieco słabszego algorytmu DSA. Podczas instalacji serwera SSH tworzona jest para kluczy – klucz publiczny i klucz prywatny serwera – służą one do szyfrowania i deszyfrowania komunikacji. Podczas pierwszego połączenia z serwerem, klient prosi serwer o klucz publiczny serwera i zapisuje publiczny klucz serwera na swoim dysku. Następnie klient generuje tak zwany klucz sesji, który będzie stosowany do szyfrowania całej komunikacji z wykorzystaniem algorytmów kryptografii symetrycznej (np. IDEA, 3DES, AES). Klucz sesji zostaje zaszyfrowany kluczem publicznym otrzymanym wcześniej od serwera i jest do niego odsyłany. Od tego momentu cała komunikacja szyfrowana jest kluczem

sesji. Standardowo SSH działa na porcie 22. Jednym z najpopularniejszych programów klienckich wykorzystujących SSH, jest program PUTTY. Aby połączyć się zdalnie z hostem, wystarczy go uruchomić, podać nazwę hosta lub jego adres IP, wybrać SSH jeśli domyślnie nie jest zaznaczony i kliknąć OPEN. Jeśli po raz pierwszy łączymy się ze zdalnym hostem potwierdzamy chęć nawiązania połączenia i już możemy zdalnie nim zarządzać. Alternatywanie do bezpiecznej komunikacji ze zdalnym serwerem można wykorzystać program WinSCP.

VPN oznacza „wirtualną sieć prywatną" - jak sama nazwa wskazuje, służy do łączenia się z sieciami prywatnymi za pośrednictwem sieci publicznych, takich jak Internet. Po połączeniu z siecią prywatną firmy za pomocą tunelu VPN klient może działać tak jakby był podłączony bezpośrednio do swojej sieci firmowej, a więc może uzyskiwać dostep do udziałów plikowych i innych zasobów sieciowych tak, jakby faktycznie korzystał z sieci fizycznej. Klient VPN komunikuje się za pośrednictwem publicznego Internetu i wysyła ruch sieciowy komputera za pośrednictwem zaszyfrowanego połączenia z serwerem VPN. Szyfrowanie zapewnia bezpieczne połączenie, co oznacza, że nie jest możliwe odczytanie danych przesyłanych w sieci VPN. W zależności od VPN cały ruch sieciowy komputera może być przesyłany przez VPN - lub tylko niektóre z nich (zazwyczaj jednak cały ruch sieciowy przechodzi przez VPN przez co nie jest on dostępny dla osób nieuprawnionych). Istotne jest to, że VPN działa bardziej na poziomie systemu operacyjnego niż na poziomie aplikacji. Innymi słowy, po skonfigurowaniu połączenia VPN, system operacyjny klienta może skierować cały ruch sieciowy za jego pośrednictwem ze wszystkich aplikacji i nie trzeba konfigurować poszczególnych aplikacji osobno, tak jak to ma miejsce w przypadku tunelu wykorzystującego protokół SSH.

W ramach laboratorium zadanie 3 dotyczy konstrukcji tunelu SSH do bezpiecznych połączeń z wybranym serwerem (fizycznym - znajdującym się we własnej sieci lokalnej lub zainstalowanym na wybranym hostingu, albo serwerem zainstalowanym na maszynie wirtualnej). Do uwierzytelniania połączeń wykorzystać klucze kryptograficzne (publiczny i prywatny) kryptografii asymetrycznej (połączenie bezhasłowe) wygenerowane po stronie klienta. W nowszej wersji SSH w przypadku systemu Linux zostaną one zapisane w plikach ~/.ssh/id_rsa i ~/.ssh/id_rsa.pub. Następnie należy skopiować zawartość klucza publicznego klienta (tj. przesłać klucz publiczny) do pliku ~/.ssh/authorized_keys na zdalnej maszynie. Z kolei klucz prywatny klienta zabezpieczyć hasłem (passphrase) na komputerze klienta. Dzięki temu wszyscy, którzy mają prawo do czytania pliku z kluczem prywatnym nie będą mogli go użyć bez podania hasła klucza, aby dostać się do zdalnej maszyny w ten sam sposób - czyli bez podawania hasła. Dotyczy to również wszystkich osób, które mają prawa roota na lokalnej maszynie. Użycie hasła do klucza prywatnego (passphrase) zmusza do wpisywania hasła podczas wykonywania operacji w ramach sesji. Aby tego uniknąć możemy wykorzystać funkcjonalność o nazwie ssh-agent, która pozwala nam na wprowadzenie passphrase jednokrotnie w obrębie danej sesji. Dzięki użyciu ssh-agent wymagane jest wpisanie hasła tylko jeden raz co nie jest już tak uciążliwe. Po uruchomieniu agenta wprowadzenie hasła do aplikacji odbywa się komendą: ssh-add <key>. Do realizacji zadania w środowisku Windows można

wykorzystać oprogramowanie PUTTY lub WinSCP, które upraszcza procedurę konstrukcji kanału SSH z użyciem logowania bezhasłowego.

Z kolei zadanie 4 dotyczy konstrukcji bezpiecznego kanału komunikacyjnego z wykorzystaniem sieci VPN typu klient-to-site. System można skonfigurować używając maszyn fizycznych lub wirtualnych, np. za pomocą oprogramowania OpenVPN.

Opis oprogramowania

W ćwiczeniu wykorzystywany jest program RSA_KEY.CPP zaprojektowany w języku C++. Program ten generuje klucze kryptograficzne dla algorytmu RSA i zmiennych typu long. Zawiera on następujące funkcje:

- POTEGA_MOD - funkcja obliczająca wartość $x^e \pmod n$,
- LICZBA_PIERWSZA - funkcja sprawdzająca czy wylosowana liczba nieparzysta *p* jest liczbą pierwszą, a jeśli nie jest wyznaczająca liczbę pierwszą większą od liczby wylosowanej; funkcja realizuje test Fermata dla ustalonego *p* oraz *ns* wartości $x < p$, gdzie $x = 0, 1, \ldots, ns-1$ (np. $ns = 9$).
- MULTI_INV - funkcja obliczająca odwrotność multiplikatywną liczby *x* według zależności multi_inv ∗ x (mod n) = 1.
- GCD_EUCLID - funkcja obliczająca największy wspólny podzielnik *gcd*(x,y) za pomocą algorytmu Euclidesa.

Przebieg ćwiczenia

Przed ćwiczeniem należy zapoznać się z opisem algorytmu i oprogramowaniem ćwiczenia. Uruchomić i przeanalizować pracę programu RSA_KEY.CPP. Wygenerować kilka zestawów kluczy kryptograficznych.

1. Wykorzystując funkcje z programu RSA_KEY napisać i uruchomić własny program, który będzie wykonywał następujące czynności:
   - generował klucze kryptograficzne,
   - szyfrował jeden znak w kodzie ASCII i drukował wynik szyfrowania (wykorzystać funkcję POTEGA_MOD);
   - deszyfrował utworzony kryptogram i drukował wynik deszyfrowania (wykorzystać funkcję POTEGA_MOD).

   W wyniku szyfrowania znaków kodu ASCII można otrzymać liczby złożone z kilku bajtów (w zależności od rozmiaru typu całkowitego). W niniejszym ćwiczeniu generowane wartości parametru *n* oraz kryptogramów C nie przekraczają 65535, czyli mieszczą się na 16 bitach. Wydrukować kryptogram (szyfrogram) w postaci szesnastkowej (zobacz przykładowy program RSA_KEY_N).

2. Zastosować opracowany program do szyfrowania i deszyfrowania plików bajt po bajcie. Plik jawny po zaszyfrowaniu zapisac w pliku o nazwie zaszyfr, a plik odszyfrowany w pliku o nazwie odszyfr. Pliki jawny i odszyfrowany powinny zawierać identyczne dane i posiadać taką samą długość. Wyznaczyć zależność czasu szyfrowania pliku w zależności od jego rozmiaru (użyć co najmniej trzech różnych rozmiarów plików).
3. Skonfigurować tunel SSH w wybranym środowisku (Windows, Linux). Zapewnić zdalne uwierzytelnianie (logowanie) z wykorzystaniem kluczy kryptograficznych RSA. Zadanie zrealizować na maszynach fizycznych lub wirtualnych. Przetestować połączenie oraz bezpieczną wymianę plików pomiędzy klientem i serwerem.
4. Skonfigurować połączenie VPN typu klient-to-site w wybranym środowisku (Windows, Linux), np. wykorzystać oprogramowanie OpenVPN. W systemie można zastosować klucz statyczny lub generowany automatycznie (algorytm DH) do realizacji szyfrowania symetrycznego w kanale. Do uwierzytelniania podmiotów (serwera, klienta) można wykorzystać certyfikaty kluczy i/lub hasło. Zadanie zrealizować na maszynach fizycznych lub wirtualnych. Przetestować połączenie oraz bezpieczną wymianę plików pomiędzy klientem i siecią prywatną.