INSTYTUT TELEKOMUNIKACJI
POLITECHNIKA WARSZAWSKA

KRZYSZTOF SZCZYPIORSKI
KONRAD WRONA

PRETTY GOOD PRIVACY
OGÓLNODOSTĘPNY SYSTEM OCHRONY INFORMACJI

NA PRAWACH RĘKOPISU

Warszawa, czerwiec 1996

WSTĘP

Bariery kontaktów międzyludzkich są skrzętnie usuwane przez coraz doskonalsze środki (narzędzia i usługi) telekomunikacyjne. Telefony komórkowe, superszybkie komputery, sieci komputerowe, światłowody - są już teraźniejszością. Modne ostatnio pojęcia "globalna wioska" i "cyfrowy świat" opisują metodę rozchodzenia się informacji we współczesnym świecie. Czy zatem w tak skonstruowanej rzeczywistości jest miejsce na intymność, niezależność, czy my, jako użytkownicy technicznych wynalazków, możemy czuć się bezpieczni?

Otóż miejsce jest, ale skoro jeszcze go nie znamy, nie możemy udawać, że czujemy się bezpiecznie. Świadomość i użycie metod ochrony informacji da nam poszukiwaną gwarancję spokuju.

Przedstawiony w tym opracowaniu system PGP - Pretty Good Privacy - jest zaprojektowany do użycia w sieciach komputerowych i na samodzielnych stanowiskach. Korzysta z technik kryptografii klucza publicznego - każdy użytkownik posiada dwa klucze - publiczny i prywatny. **Klucz publiczny** służy **do szyfrowania wiadomości kierowanych do jego właściciela**, **klucz prywatny** - **do odszyfrowania tych wiadomości przez** wspomnianego **właściciela**. Dzięki systemowi PGP wymiana informacji między dwiema osobami - nawet nie znającymi się osobiście - staje się bezpieczna.

Opracowanie składa się z trzech części:

Część pierwsza - Opis działania systemu PGP - ogólny opis algorytmów:
- IDEA,
- RSA,
- MD5,
- ZIP.

Część druga - Podręcznik użytkownika:
- instalacja systemu pod MS DOS,
- spis komend,
- liczne uwagi ułatwiające pracę.

Część trzecia - Ćwiczenia - zestaw ćwiczeń pomocnych w szybkim opanowaniu programu.

CZĘŚĆ PIERWSZA - Opis działania systemu PGP

1.1. Co to jest PGP?

PGP (Pretty Good Privacy) jest ogólnodostępnym[1] systemem zapewniającym integralność, poufność i uwierzytelnienie, przesyłanej poczty elektronicznej oraz przechowywanych danych.

Pierwsza wersja PGP, opracowana przez Philipa Zimmermanna, pojawiła się w 1991 roku. Od tego czasu program ten zrobił burzliwą karierę, stając się nieoficjalnym standardem wymiany poufnej poczty elektronicznej.

1.2. Co robi PGP?

PGP zapewnia realizację następujących usług kryptograficznych:
- poufności,
- uwierzytelnienia,
- potwierdzenia integralności danych.

Usługa poufności jest realizowna poprzez szyfrowanie pliku (plików). W tej formie mogą być przechowywane lub też przesyłane pocztą elektroniczną. W przypadku transmisji, tylko odbiorca docelowy posiada możliwość odszyfrowania i przeczytania wiadomości. Przechwycenie przesyłanej wiadomości nie pozwala na zapoznanie się z jej treścią.

Dzięki podpisowi cyfrowemu odbiorca może sprawdzić tożsamość nadawcy oraz integralność dokumentu, czyli brak celowych lub przypadkowych przekłamań powstałych w czasie transmisji.

1.3. Istniejące wersje programu.

Najbardziej rozpowszechnionymi na świecie wersjami PGP są wersje: 2.3a oraz 2.6.x. PGP wykorzystuje opatentowany w USA algorytm RSA bez zgody jego właścicieli, stąd jego użytkowanie w tym kraju jest niezgodne z prawem. W celu usunięcia problemów prawnych, w maju 1994 roku na MIT (Massachusetts Institute of Technology), została, za zgodą RSA Data Security Inc. (właściciela patentu), opracowana wersja 2.6 (z późniejszymi poprawkami 2.6.1 i 2.6.2). Wersja ta może być legalnie użytkowana w USA do celów niekomercyjnych. Wadą tej wersji jest celowa niezgodność formatu wiadomości i kluczy przez nią używanych z wcześniejszymi nielegalnymi wersjami. Ograniczono też maksymalną długość generowanych kluczy do 1024 bitów (dla wersji 2.3a wartość ta wynosi 1264 bity, a dla 2.6.3 aż 4096).

[1] program należy do grupy oprogramowania **public domain**

W związku z panującymi w USA przepisami eksportowymi, wywóz nowoczesnego oprogramowania kryptograficznego (takiego jak PGP) podlega ścisłej kontroli państwa. W związku z tym w Europie zostało opracowanych wiele innych wersji PGP oznaczanych poprzez dodanie **ui** lub **i** (Unofficial / International realese). W trakcie ćwiczeń wykorzystywana jest wersja 2.6.3i for DOS, będąca przeróbką PGP 2.6.1 MIT, w której usunięto ograniczenia długości klucza i niekompatybilność z wersją 2.3a.

Dostępna jest również komercyjna (odpłatna) wersja PGP opracowana przez firmę ViaCrypt. Wersja ta jest w pełni kompatybilna z wersjami 2.3, 2.3a, 2.4 i 2.6.x.

Jedną z głównych zalet PGP jest ogólnodostępność jego implementacji dla praktycznie wszystkich platform systemowych (UNIX, Windows, Macintosh, DOS).

Z punktu widzenia użytkownika wszystkie opisane odmiany programu są dla danego systemu identyczne.

1.4. Jak działa PGP?

Przy szyfrowaniu klasycznym PGP używa algorytmu IDEA.

Do usług wymagających użycia klucza publicznego PGP korzysta z ogólnie uznanego algorytmu RSA.

Do generacji podpisów cyfrowych i "odcisków klucza publicznego", wykorzystywana jest również funkcja skrótu MD5, uważana za jedną z najmocniejszych w swoim rodzaju.

1.4.1 Co to jest IDEA?

IDEA (International Data Encryption Algorithm) jest blokowym konwencjonalnym algorytmem kryptograficznym. Został on opracowany przez Xuejia Lai i Jamesa Masseya ze Swiss Federal Institute of Technology. Algorytm używa 128-bitowego klucza do szyfrowania 64-bitowych bloków danych. (Dla porównania DES wykorzystuje również 64-bitowe bloki, ale efektywnie tylko 56-bitowy klucz.)

Zalety algorytmu IDEA to:

- duża przestrzeń kluczy ($2^{128} \cong 3.4 \cdot 10^{38}$) zapewniająca większą odporność na atak metodą prób i błędów.

  Przeszukanie przestrzeni kluczy przy szybkości 10 bilionów prób na sekund, zajęłoby średnio $5.4 \cdot 10^{18}$lat, co przy wieku Wszechświata ocenianym na ok. 15 miliardów lat, czyli $1.5 \cdot 10^{10}$ lat wydaje się czasem dość długim.

- bardziej odporna na kryptoanalizę struktura wewnętrzna algorytmu,

- zoptymalizowanie algorytmu pod kątem implementacji programowych.

Dokładny opis algorytmu można znaleźć w [Schnei94], [Stalli95a], [Stalli95b], [Szczyp95].

1.4.2 Co to jest RSA?

RSA (Rivest, Shamir, Adleman) jest jednym z pierwszych systemów klucza publicznego, opracowanym na MIT w 1977 roku przez Rona Rivesta, Adiego Shamira i Lena Adlemana. RSA wykorzystuje arytmetykę modularną. Dla danego tekstu jawnego M i zaszyfrowanego C przy użyciu klucza e i d, zachodzi:

$$C = M^e \bmod n$$
$$M = C^d \bmod n = \left(M^e\right)^d \bmod n = M^{e \cdot d} \bmod n$$

Zarówno nadawca jak i odbiorca musi znać wartość n oraz e. Tylko odbiorca zna wartość d.

Parę (e,n) przyjęto nazywać kluczem publicznym, a parę (d,n) kluczem prywatnym.

Aby system taki mógł zostać wykorzystany praktycznie, musi spełniać następujące wymagania:

i) Dla każdego M < n muszą istnieć takie wartości e,d i n, że $M = M^{e \cdot d} \bmod n$.

ii) Dla każdego M < n obliczenie M^e i C^d musi by dość proste.

iii) Odtworzenie d na podstawie e i n musi być praktycznie niewykonalne.

System RSA spełnia 2 pierwsze wymagania, a 3. z nich jest prawdziwe dla wystarczająco dużych e i n.

Szczegółowy opis systemów klucza publicznego, a w szczególności RSA, jest przedstawiony w [Schnei94], [Stalli95a], [Stalli95b], [Szczyp95].

1.4.3 Co to jest MD5?

Funkcja skrótu (message-digest function) MD5 została opracowana przez Rona Rivesta z MIT. Algorytm skraca (streszcza) tekst wejściowy o dowolnej długości do postaci unikalnego 128-bitowego ciągu. W funkcji MD5 każdy bit wynikowy zależy od każdego bitu wejściowego. Cecha ta, w połączeniu ze skomplikowanym systemem przekształceń wewnętrznych algorytmu, minimalizuje prawdopodobieństwo, że dwie różne wiadomości zostaną skrócone (streszczone) do tej samej postaci.

Poniżej przedstawiono skróty czterech wyrazów:

```
MD5 ("krawat")    = df85bc0a1a695bb34f29d0327d6b0ad7
MD5 ("Marta")     = 83f9c4eb242966cdcada1d01be5d9b15
MD5 ("marta")     = a763a66f984948ca463b081bf0f0e6d0
MD5 ("plomba")    = b29795ff70be9e892bb7bf82744abdfa
```

Dokładny opis funkcji skrótu, w tym MD5, można znaleźć w [RFC1321], [Schnei94], [Stalli95b] .

1.4.4 Do czego to wszystko służy?

1.4.4.1 Generacja podpisu cyfrowego

Wiadomość jest skracana przy użyciu MD5. Kod wynikowy, jednoznacznie określający jej treść, jest szyfrowany przy użyciu algorytmu RSA oraz prywatnego klucza nadawcy (co pozwala na jego jednoznaczną identyfikację) i zostaje dołączony do wiadomości.

Odbiorca wiadomości, w celu sprawdzenia podpisu nadawcy, używa MD5 do streszczenia otrzymanej wiadomości. Następnie odszyfrowuje załączony podpis cyfrowy przy pomocy klucza publicznego nadawcy i porównuje oba skróty.

1.4.4.2 Szyfrowanie wiadomości

PGP automatycznie generuje unikalny 128-bitowy klucz sesyjny. Wiadomość jest szyfrowana przy użyciu algorytmu IDEA[2], a klucz sesyjny jest szyfrowany przy użyciu RSA kluczem publicznym odbiorcy. System PGP dołącza zaszyfrowany klucz sesyjny do pliku z przesyłaną wiadmością.

Odbiorca odszyfrowuje klucz sesyjny przy użyciu swego klucza prywatnego i przy użyciu algorytmu IDEA odszyfrowuje wiadomość.

1.4.5 Usługi dodatkowe

1.4.5.1 Kompresja danych

Wiadomość może zostać skompresowana przed transmisją (lub w celu przechowywania) przy użyciu algorytmu ZIP.

Zalecana jest kompresja wiadomości przed jej zaszyfrowaniem, w celu eliminacji redundancji językowych i utrudnienia ewentualnej kryptoanalizy. W tym celu należy w pliku config.txt nadać zmiennej compress atrybut on (compress=on).

1.4.5.2 Zapewnienie kompatybilności z różnymi systemami poczty elektronicznej

[2] algorytm pracuje w Trybie Wiązania Bloków Zaszyfrowanych - CBC (Cipher Block Chaining Mode). Każdy z bloków zależy zarówno od bieżącego, jak i od wszystkich poprzedzających go bloków wiadomości

Aby zapewnić bezbłędne przekazywanie wiadomości pomiędzy różnymi systemami sieciowymi, zaszyfrowana wiadomość może zostać przekształcona do postaci ciągu znaków ASCII za pomoc konwersji radix-64 (alfabet 64-znakowy). Opcja ta może być również wykorzystana do przesyłania pocztą elektroniczną plików binarnych (np. programów).

Zaleca się korzystanie z tej opcji poprzez ustawienie w pliku config.txt zmiennej armor=on.

Ponieważ niektóre systemy narzucają ograniczenia na maksymalną długość przekazywanych pocztą elektroniczną wiadomości, PGP pozwala na ich segmentację na dowolną ilość wiadomości o określonej długości (opcja armorlines - domyślnie 720).

CZĘŚĆ DRUGA - Podręcznik użytkownika

2.1 Instalacja pod MS DOS

Pakiet PGP 2.6i jest dostarczany w postaci skompresowanego archiwum o nazwie **PGP26i.ZIP**.

Poniższe wydruki (komenda dir) przedstawiają zawartość utworzonych po dekompresji programem **PKUNZIP.EXE** katalogów.

```
 Volume in drive C is COOL_WATER     Serial number is 1CD8:84A2
 Directory of  c:\security\pgp26i\*.*

.            <DIR>     95-01-16  10:00
..           <DIR>     95-01-16  10:00
DOC          <DIR>     95-01-16  10:00
config.txt       3986  94-05-21  18:06
es.hlp           4379  94-05-06  15:58
fr.hlp           4467  94-05-06  15:58
keys.asc        10573  94-10-16  18:44
language.txt    70744  94-05-23  18:40
pgp.exe        228767  94-10-16  23:47
pgp.hlp          3983  94-06-19  17:15
pgp26ii.asc       293  94-10-16  23:50
readme.1st      13268  94-10-16  23:55
setup.doc       15293  94-10-16  23:55
         355.753 bytes in 13 file(s)          376.832 bytes allocated
   1.985.387.192 bytes free

 Volume in drive C is COOL_WATER     Serial number is 1CD8:84A2
 Directory of  c:\security\pgp26i\doc\*.*

.            <DIR>     95-01-16  10:00
..           <DIR>     95-01-16  10:00
appnote.doc      6368  94-05-20   3:47
blurb.txt         730  94-09-03   0:40
changes.doc     18150  94-09-03   0:17
keyserv.doc      4295  94-05-23  18:21
pgformat.doc    37351  94-05-21  18:10
pgpdoc1.txt     84757  94-09-03   0:28
pgpdoc2.txt    132963  94-09-04  21:46
politic.doc     18215  94-05-07  15:15
readme.doc       6768  94-09-05  16:21
setup.doc       15447  94-09-05  20:10
         325.044 bytes in 12 file(s)          348.160 bytes allocated
   1.985.387.192 bytes free
```

CONFIG.TXT jest plikiem konfiguracyjnym niezbędnym do pracy PGP.

KEYS.ASC zawiera klucze publiczne niektórych twórców programu.

LANGUAGE.TXT zawiera tłumaczenia wyświetlanych informacji - języki: francuski i hiszpański.

PGP.EXE	jest właściwym programem.
PGP.HLP, FR.HLP i ES.HLP	zawierają krótką pomoc w dostępnych językach.
PGP26II.ASC	zawiera cyfrowy podpis jednego z twórców wersji 2.6i -Stale Schumachera.
README.1ST	zawiera wstępny opis wersji 2.6i.
SETUP.DOC	zawiera informacje dotyczące instalacji programu.

Katalog DOC zawiera dokumentację pakietu.

W trakcie użytkowania program PGP generuje pięć dodatkowych plików:
- **PUBRING.PGP** - domyślny publiczny brelok[3],
- **PUBRING.BAK** - kopia (back-up) publicznego breloku,
- **SECRING.PGP** - domyślny prywatny brelok,
- **SECRING.BAK** - kopia (back-up) prywatnego breloku,
- **RANDSEED.BIN** - plik używany przy generacji liczb losowych.

2.2 Krótki spis komend PGP 2.6i

Wyświetlenie pomocy:

pgp -?

lub pgp -h

Zaszyfrowanie pliku zawierającego tekst jawny kluczem publicznym odbiorcy:

pgp -e plik jej_id_użytkownika

Podpisanie pliku zawierającego tekst jawny twoim kluczem prywatnym:

pgp -s plik [-u twój_id_użytkownika]

Podpisanie pliku z tekstem jawnym twoim kluczem prywatnym i pozostawienie tekstu czytelnego bez uruchamiania PGP:

pgp -sta plik [-u twój_id_użytkownika]

Podpisanie pliku zawierającego tekst jawny twoim kluczem prywatnym, a następnie zaszyfrowanie go kluczem publicznym odbiorcy:

pgp -es plik jej_id_użytkownika [-u twój_id_użytkownika]

[3] (ang.) keyring - kółko na klucze - używany w tej publikacji polski odpowiednik - brelok - jest tylko propozycją. Pojęcie breloku wiąże się z omówioną dalej koncepcją zarządzania kluczami stosowaną w PGP

Zaszyfrowanie pliku zawierającego tekst jawny w sposób konwencjonalny:

pgp -c plik

Odszyfrowanie pliku lub sprawdzenie zgodności podpisu w podpisanym pliku:

pgp zaszyfrowany_plik [-o plik_z_tekstem_jawnym]

Zaszyfrowanie informacji dla dowolnej liczby użytkowników:

pgp -e plik id_użytkownika_1 id_użytkownika_2 id_użytkownika_3

Zarządzanie kluczami:

Spis komend dotyczących zarządzania kluczami:

pgp -k

Wygenerowanie pary kluczy (publiczny/prywatny):

pgp -kg

Dodanie publicznego lub prywatnego klucza przechowywanego w pliku plik_z_kluczem do twojego publicznego lub prywatnego breloku:

pgp -ka plik_z_kluczem [brelok]

Skopiowanie klucza z twojego publicznego lub prywatnego breloku:

pgp -kx id_użytkownika plik_z_kluczem [brelok]

lub **pgp -kxa id_użytkownika plik_z_kluczami [brelok]**

Wyświetlenie zawartości twojego publicznego breloku:

pgp -kv[v] [id_użytkownika] [brelok]

Wyświetlenie odcisków klucza publicznego - pomoc w weryfikacji jego właściciela przez telefon:

pgp -kvc [id_użytkownika] [brelok]

Wyświetlenie zawartości i sprawdzenie ważności podpisów w twoim publicznym breloku:

pgp -kc [id_użytkownika] [brelok]

Edytowanie id_użytkownika lub paszportu[4] dla twojego klucza prywatnego:

pgp -ke id_użytkownika [brelok]

[4] (ang.) pass phrase - używany w tej publikacji polski odpowiednik - paszport - jest tylko propozycją. W PGP paszport może być dowolnym zdaniem - może zawierać dowolne znaki możliwe do wydrukowania (printable) - nie jest zatem klasycznie pojmowanym hasłem

Edytowanie parametrów zaufania dla klucza publicznego:

pgp -ke id_użytkownika [brelok]

Usunięcie klucza lub tylko id_użytkownika z publicznego breloku:

pgp -kr id_użytkownika [brelok]

Podpisanie lub sprawdzenie czyjegoś klucza publicznego na twoim publicznym breloku:

pgp -ks id_użytkownika [-u twoje_id_użytkownika] [brelok]

Usunięcie z breloku wybranych podpisów dla danego użytkownika:

pgp -krs id_użytkownika [brelok]

Trwałe unieważnienie twojego klucza, wydanie certyfikatu kompromisu:

pgp -kd twoje_id_użytkownika

Zamrożenie lub uaktywnienie publicznego klucza na twoim publicznym breloku:

pgp -kd id_użytkownika

Tajemne komendy:

Odszyfrowanie pliku i pozostawienie w nim nietkniętego podpisu:

pgp -d zaszyfrowany_plik

Stworzenie podpisu odseparowanego od dokumentu:

pgp -sb plik [-u twoje_id_użytkownika]

Odseparowanie podpisu od podpisanego pliku:

pgp -b zaszyfrowany_plik

Kombinacje komend:

Utworzenie zaszyfrowanego_pliku w formacie ASCII radix-64 - opcja -a (ascii) podczas szyfrowania (lub podpisywania) pliku lub kopiowania klucza:

pgp -sea plik jej_id_użytkownika

lub **pgp -kxa id_użytkownika plik_z_kluczami [brelok]**

Usunięcie pliku z tekstem jawnym po stworzeniu zaszyfrowanego pliku - opcja -w (wipe - wyczyść) podczas szyfrowania lub podpisywania pliku:

pgp -sew plik_z_tekstem_jawnym jej_id_użytkownika

Zaznaczenie tego, że plik_z_tekstem_jawnym zawiera tylko znaki ASCII i powinien być przetworzony u odbiorcy według zasad obsługi plików tekstowych w jego systemie - opcja -t (text - tekst):

pgp -seat plik_z_tekstem_jawnym jej_id_odbiorcy

Wyświetlenie na ekranie odszyfrowanego tekstu jawnego (tak jak unixowa komenda "more") bez zapisu do pliku - opcja -m (more) podczas odszyfrowania:

pgp -m zaszyfrowany_plik

Uniemożliwienie odbiorcy zapisu na dysk odszyfrowanego tekstu jawnego - TYLKO dla jej oczu - tekst będzie wyświetlany na ekranie - opcja -m:

pgp -steam plik_z_tekstem_jawnym jej_id_użytkownika

Odtworzenie oryginalnej nazwy pliku z tekstem jawnym podczas odszyfrowywania - opcja -p:

pgp -p zaszyfrowany_plik

Użycie unixowego trybu - czytanie ze stdin i zapis do stdout - opcja -f (filter):

pgp -feast jej_id_użytkownika <plik_WEjściowy >plik_WYjściowy

2.3 Ustawienie zmiennej środowiskowej TZ (tylko platforma MS-DOS)

Format zmiennej środowiskowej (*environment string*) TZ [Borland]:

TZ = zzz[+/-]d[d][lll]

Składnik	**Znaczenie**
zzz	Ciąg składający się z trzech znaków. Określa strefę czasową. Wszystkie trzy znaki są wymagane.
[+/-]d[d]	Wymagane pole. Znak opcjonalnie. Ilość cyfr: jedna lub więcej. Liczba określa różnicę między daną strefy czasową a GMT wyrażoną w godzinach. - Liczby dodatnie - strefy na zachód od GMT. - Liczby ujemne - strefy na wschód od GMT. Liczba używana w obliczeniach czasu strefy.

lll	Opcjonalne, trzyznakowe pole. Określa dla danej strefy czasowej czas słoneczny (*daylight saving time*). - jeśli pole to występuje, czas ten jest różny od 0. - jeśli pole to nie występuje, czas ten jest równy 0.

dla Polski (wg. [Zimmer93]) TZ=MET-1DST.

2.4 Polska wersja językowa (LANGUAGE.TXT)

Przy tworzeniu polskiej wersji językowej pojawia się problem wyboru standardu polskich znaków. Przedstawiona poniżej propozycja zakłada wykorzystanie strony kodowej 852.

Dodając napisy w języku polskim do pliku LANGUAGE.TXT należy przestrzegać konwencji wyznaczonej przez autorów PGP - należy korzystać z 7-bitowego alfabetu ASCII. Kody polskich znaków - jako wyższe od 127 - uzyskuje się w następujący sposób:

ć	**\206**	**ą**	**\245**
ł	**\210**	**Ę**	**\250**
Ź	**\215**	**ę**	**\251**
Ć	**\217**	**ź**	**\253**
Ś	**\227**	**Ż**	**\275**
ś	**\230**	**ż**	**\276**
Ł	**\235**	**Ó**	**\340**
ó	**\242**	**Ń**	**\343**
Ą	**\244**	**ń**	**\344**

Przyjęta konwencja jest sformalizowana przez składnię funkcji printf języka C.

CZĘŚĆ TRZECIA - Ćwiczenia

Ćwiczenia zamieszczone w tej części publikacji dotyczą następujących zagadnień:

- generacja kluczy,
- szyfrowanie, podpisywanie wiadomości,
- odszyfrowanie wiadomości,
- zastosowanie kompresji.

Ćwiczenie 1 - generacja kluczy

1.1 Wygenerować parę kluczy (publiczny i prywatny):

pgp -kg

Co się stanie ?

1.1.1 Program poprosi o podanie pożądanej długości klucza:

```
Pick your RSA key size:
    1)   512 bits- Low commercial grade, fast but less secure
    2)   768 bits- High commercial grade, medium speed, good security
    3)  1024 bits- "Military" grade, slow, highest security
Choose 1, 2, or 3, or enter desired number of bits:
```

Uwaga: Wyświetlane opcje nie odzwierciedlają wszystkich możliwości programu, a jedynie propozycje (i komentarze) jego autorów. **Nie udokumentowana opcja: pgp -kg długość_klucza -** 384 ≤ długość_klucza ≤ 2048

Proces szyfrowania i odszyfrowania praktycznie nie zależy od wielkości klucza, dlatego zalecane jest używanie klucza o długości 1024 bity.

Jednak czas tworzenia klucza rośnie wykładniczo wraz z jego długością[5]. Z tego powodu w ćwiczeniu należy korzystać z klucza 512-bitowego.

1.1.2.Użytkownik jest proszony o wprowadzenie swojego identyfikatora:

```
Generating an RSA key with a 512-bit modulus.

You need a user ID for your public key.  The desired form for this
user ID is your name, followed by your E-mail address enclosed in
<angle brackets>, if you have an E-mail address.
```

[5] Orientacyjne czasy generacji klucza o długości 2048 trwa:
- 6 minut i 40 sekund na komputerze z procesorem i486 DX2 (zegar 66 MHz, 32 MB RAM),
- 24 minuty i 50 sekund na komputerze z procesorem AMD 386 DX (zegar 40 MHz, koprocesor, 4 MB RAM).

```
For example:  John Q. Smith <12345.6789@compuserve.com>
Enter a user ID for your public key:
```

Uwaga: Zalecane jest podanie swojego imienia, nazwiska oraz adresu poczty elektronicznej (ewentualnie numeru telefonu).

1.1.3. Następnie użytkownik jest proszony o wprowadzenie swojego paszportu:

```
You need a pass phrase to protect your RSA secret key.
Your pass phrase can be any sentence or phrase and may have many
words, spaces, punctuation, or any other printable characters.

Enter pass phrase:
Enter same pass phrase again:
```

Uwaga:

1. **Nie wybieraj jako frazy**: swojego imienia, imion swoich bliskich, imion znanych postaci rzeczywistych i fikcyjnych, słów które są w słowniku dostępnym w twoim systemie, nazw towarów, tytułów powieści, sztuk i filmów oraz programów telewizyjnych, radiowych, ciągów znakowych w stylu QWERTY, 1234, ogólnie znanych skrótów (np. NBP, SPEC czy EB) itd.
2. **Nie używaj** fraz krótszych niż 7 znaków.
3. **Wybierz** frazę, którą będziesz mógł zapamiętać.
4. **Nie używaj** frazy, którą już gdzieś stosujesz (w szczególności hasła swego konta sieciowego).
5. **Nie mrucz** swego hasła w czasie wpisywania.
6. **Umieść** w swojej frazie cyfry, znaki pomocnicze (np. oddziel nimi sylaby - **KA-TA**7**STRO**&**FA**).
7. **Nie przestrzegaj** reguł ortograficznych.

1.1.4. Użytkownik musi napisać na klawiaturze dowolny, losowy tekst o zadanej długości. (Program mierzy odstępy pomiędzy kolejnymi uderzeniami klawiatury - są one traktowane jak generator liczb losowych.)

1.1.5. Rozpoczyna się proces tworzenia kluczy.

```
Note that key generation is a lengthy process.
..................++++ .......++++
Key generation completed.
```

1.1.5. Klucz zostaje automatycznie podpisany

1. Twój podpis gwarantuje, że nikt nie będzie mógł zmienić jego wartości ani identyfikatora.
2. Podpisując czyjś klucz publiczny, uwiarygodniasz go w oczach innych. Twój podpis jest rodzajem gwarancji, że sygnowany klucz nie został sfałszowany. Nie należy podpisywać klucza, którego nie dostało się bezpośrednio od jego właściciela lub co do którego autentyczności istnieją jakiekolwiek wątpliwości.

1.3 Pobrać klucz:

pgp -kxa [id_użytkownika [nazwa_pliku_z_kluczem]]

(np. **pgp -kxa**)

Uwaga: Aby rozpowszechnić swój klucz, musisz najpierw go pobrać z publicznego breloku.

Co się dzieje?

1.3.1 Program pobiera klucz i zapisuje go w formacie radix-64 do pliku o podanej nazwie (np. **usr2_key.asc**):

```
Extracting from key ring: 'pubring.pgp', userid "USER 2".

Key for user ID: USER 2
512-bit key, Key ID 3256783D, created 1995/01/11

Transport armor file: usr2_key.asc

Key extracted to file 'usr2_key.asc'.
```

1.3.2 Obejrzyj przy pomocy edytora utworzony plik.

1.4 Dodać klucz(-e) do publicznego breloka:

pgp -ka nazwa_pliku_z_kluczem(-ami)

(np. **pgp -ka keys.asc**[6])

Uwaga: Aby móc wysyłać do innych zaszyfrowane informacje i sprawdzać podpisy na otrzymywanych dokumentach musisz dodać klucze publiczne korespondentów do swojego publicznego breloku.

1.4.1 Obejrzyj rezultat wykonanej operacji przy pomocy polecenia:

pgp -kv

1.4.2 Powtórz obserwacje za pomocą opcji:

pgp -kvv

Zagadnienie:

1.Czym różnią się obie listy?

1.4.3 Sprawdź klucz Stale Schumachera używając polecenia:

[6] klucze używane w ćwiczeniu 1.4, 1.5, 1.6 - Stale'a Schumachera i Phila Zimmermanna - pochodzą z pliku KEYS.ASC

pgp -kvc stale[7]

Zagadnienie:

1. Czy zauważyłeś nowe informacje?

Uwaga:

Odcisk klucza (key fingerprint) jest bardzo pożyteczną metodą weryfikacji prawdziwości otrzymanego klucza publicznego. Jeżeli chcemy się upewnić o jego poprawności wystarczy zadzwonić do jego właściciela, poprosić o przedyktowanie odcisku i porównać go z odciskiem klucza posiadanego. Dobrym zwyczajem jest również zapisać odcisk swojego klucza publicznego w pliku .signature, dołączanym w systemie UNIX (przy odpowiedniej konfiguracji) do wszystkich wysyłanych wiadomości - ułatwi to ewentualną weryfikację klucza naszym korespondentom.

1.5 Przeprowadzić edycję klucza:

pgp -ke [id_użytkownika [brelok[8]**]]**

(np. **pgp -ke**)

1.5.1 Przeprowadź edycje swojego klucza publicznego - zmień swój identyfikator,

1.5.2 Przeprowadź edycje swojego klucza prywatnego -zmień paszport.

1.5.3 Przeprowadź edycję parametrów zaufania dla klucza publicznego Stale Schumachera:
- najpierw nadaj mu pełną wiarygodność,
- później zmień na ograniczoną

Każdorazowo obejrzyj dokonane zmiany przy pomocy opcji **-kc**.

1.6 Unieważnić klucz(-e):

pgp -kd [id_użytkownika]

(np. **pgp -kd**)

1.6.1 Unieważnij klucz Phila Zimmermanna. Obejrzyj zawartość publicznego breloku.

1.6.2 Wykonaj powyższe polecenie powtórnie. Co się stało ?

[7] stale jest imieniem Schumachera

[8] domyślnie przyjmowany jest PUBRING.PGP

1.6.3 Unieważnij[9] swój własny klucz, który wygenerowałeś na początku ćwiczenia. Utwórz certyfikat unieważniający.

Uwaga:

Jeżeli podejrzewasz, że twój klucz prywatny został zdradzony, musisz unieważnić klucz publiczny. W tym celu wystawiasz certyfikat unieważniający i rozsyłasz go do wszystkich zainteresowanych. Dobrym pomysłem jest utworzyć taki certyfikat zaraz po generacji klucza i przechowywać go w oddzielnym, bezpiecznym miejscu. Umożliwi to unieważnienie klucza nawet w przypadku fizycznej utraty wszystkich kopii breloku prywatnego (zawsze powinieneś posiadać przynajmniej jedną taką kopię na oddzielnym, zabezpieczonym nośniku). Ponieważ **unieważnienie** własnego **klucza jest nieodwołalne**, w celu wystawienia "przyszłościowego" unieważnienia, musisz najpierw sporządzić kopię swego nowego klucza, którą po uzyskaniu certyfikatu należy ponownie dodać do breloku.

Ćwiczenie 2 - szyfrowanie, podpisywanie wiadomości

2.0 Utworzyć przy pomocy zewnętrznego edytora (np. EDIT w MS DOS) plik zawierający tekst jawny np. o nazwie (USER1.TXT):

2.1 Zaszyfrować utworzony plik z tekstem jawnym:

pgp -e plik_z tekstem_jawnym

(np. **pgp -e USER1.TXT**)

Co się stanie?

2.1.1 Program wyświetli komunikat - prośbę o nazwę odbiorcy:

```
Recipients' public key(s) will be used to encrypt.
A user ID is required to select the recipient's public key.
Enter the recipient's user ID:
```

Należy podać odbiorcę wiadomości.

W aktualnym katalogu zostanie utworzony plik USER1.PGP.

2.2 Podpisać plik zawierający tekst jawny swoim kluczem prywatnym:

pgp -s plik_z tekstem_jawnym

(np. **pgp -s USER1.TXT**)

Co się stanie?

[9] **unieważnienie klucza jest nieodwołalne!** Wykonaj ten podpunkt dopiero pod koniec ćwiczeń

2.2.1 Program zapyta o paszport podpisującego.

```
A secret key is required to make a signature.
You need a pass phrase to unlock your RSA secret key.
Key for user ID "USER 1"

Enter pass phrase:
```

2.2.2 Po podaniu właściwego hasła - program wyświetli komunikat:

```
Output file 'user1.pgp' already exists.  Overwrite (y/N)?
```

Należy dać odpowiedź przeczącą (Enter lub N + Enter) i podać nową nazwę pliku:

```
Enter new file name: user1.pg2
```

Uwaga:

W celu zachowania porządku ostatni znak rozszerzenia powinien być równy numerowi wykonywanego podpunktu ćwiczenia.

Zagadnienia:

1. Jaka metoda szyfrowania zostata zastosowana przy tworzeniu powyższego pliku?
2. Co zawiera powyższy plik?

2.3 Podpisać plik z tekstem jawnym swoim kluczem prywatnym, tak aby tekst pozostał czytelny bez uruchamiania PGP:

pgp -sta plik_z_tekstem_jawnym

(np. **pgp -sta USER1.TXT**)

Co się stanie?

2.3.1 Program zapyta o paszport nadawcy (patrz punkt 2.2.1).

2.3.2 W aktualnym katalogu zostanie utworzony plik USER1.ASC.

Uwaga:

Zmienić nazwę utworzonego pliku:

```
ren USER1.ASC USER1.AS3
```

Zagadnienie:

1. Jakie praktyczne zastosowanie może mieć powyższa opcja?

2.4 Podpisać plik zawierający tekst jawny swoim kluczem prywatnym, a następnie zaszyfrować go kluczem publicznym odbiorcy:

pgp -es plik_z_tekstem_jawnym

(np. **pgp -es USER1.TXT**)

Co się stanie?

2.4.1 Program zapyta o paszport nadawcy (patrz p. 2.2.1), a następnie nazwę odbiorcy (patrz p. 2.1.1):

2.4.2 Patrz punkt 2.2.2

2.5 Zaszyfrować plik zawierający tekst jawny w sposób konwencjonalny (używając algorytmu IDEA):

pgp -c plik_z_tekstem_jawnym

(np. **pgp -c USER1.TXT**)

Co się stanie?

2.5.1. Program zapyta o **paszport dla szyfru IDEA**:

```
You need a pass phrase to encrypt the file.
Enter pass phrase:
```

2.5.2. Patrz punkt 2.2.2

Uwaga:

Jako paszportu dla algorytmu IDEA, nigdy nie podawaj paszportu chroniącego twój klucz prywatny.

Zagadnienie:

1. Spróbować wykonać to ćwiczenie jeszcze raz - tym razem nie podając żadnego paszportu. Co się wydarzy?

2.6 Zaszyfrować informację dla dwóch użytkowników, tak aby była to informacja "tylko dla ich oczu" (niemożliwy zapis do pliku), dostosowana do 7-bitowego kanału E-mail (konwersja radix-64) i aby po zaszyfrowaniu plik_z_tekstem_jawnym został wymazany z dysku:

pgp -mewa plik_z_tekstem_jawnym id_użytkownika_1 id_użytkownika_2

(np. **pgp -mewa USER1.TXT "USER 2" "USER 3"**)

Co się stanie?

2.6.1 W aktualnym katalogu zostanie utworzony plik USER1.ASC.

Uwaga:
Zmienić nazwę utworzonego pliku:

```
ren USER1.ASC USER1.AS6
```

Zagadnienia:

1. (Jeśli możliwe) korzystając z narzędzia odtwarzającego skasowane pliki[10] odzyskać plik_z tekstem_jawnym.
2. Pod edytorem obejrzeć jego zawartość. Skomentować obserwację.

Ćwiczenie 3 - odszyfrowanie wiadomości

Ćwiczenie to jest symetryczne do poprzedniego ćwiczenia - szyfrowanie, podpisywanie wiadomości.

3.0 Odebrać od innego użytkownika zaszyfrowane przez niego pliki np.:

USER2.PGP
USER2.PG2
USER2.AS3
USER2.PG4
USER2.PG5
USER2.AS6

3.1 Odszyfrować plik:

pgp zaszyfrowany_plik

(np. **pgp USER2.PGP**)

Co się stanie?

3.1.1 Program wyświetli komunikat - prośbę o paszport odbiorcy:

```
File is encrypted.  Secret key is required to read it.
Key for user ID: USER 2
512-bit key, Key ID 3256783D, created 1995/01/11

You need a pass phrase to unlock your RSA secret key.
Enter pass phrase:
```

3.1.2 Zostanie utworzony plik o nazwie USER2 zawierający tekst jawny. Korzystając z edytora przeczytać jego zawartość.

[10] np. UNDELETE (MS DOS), UNERASE (Norton Utilities), SALVAGE (Novell)

Zagadnienia:

1. Spróbować wykonać to ćwiczenie jeszcze raz - tym razem podając nieprawdziwy paszport. Co się wydarzy?

2. Co stanie się, gdy zawartość zaszyfrowanego_pliku zostanie naruszona (uszkodzona)?

Rozważyć kombinacje poniższych przypadków:

- zostaje uszkodzony:
  - i) początek pliku,
  - ii) środek lub koniec pliku,
- plik przed zaszyfrowaniem,
  - i) zostaje skompresowany
  - ii) nie zostaje skompresowany

3.2 Odszyfrować plik:

pgp zaszyfrowany_plik

(np. **pgp USER2.PG2**)

Co się stanie?

3.2.1. Program oznajmi, kto podpisał plik.

3.2.2. Program zgłosi znany komunikat:

```
Output file 'user2' already exists.  Overwrite (y/N)?
```

Należy dać odpowiedź przeczącą (Enter lub N + Enter) i podać nową nazwę pliku:

```
Enter new file name: user2.tx2
```

Uwaga:

W celu zachowania porządku ostatni znak rozszerzenia powinien być równy numerowi wykonywanego podpunktu ćwiczenia.

Korzystając z edytora przejrzeć zawartość utworzonego pliku.

3.3 Obejrzeć pod edytorem zewnętrznym plik

edit zaszyfrowany_plik

(np. **edit USER2.AS3**)

Następnie odszyfrować ten plik, tak aby została odtworzona oryginalna nazwa pliku z tekstem jawnym:

pgp -p zaszyfrowany_plik

(np. **pgp -p USER2.AS3**)

Co się stanie?

3.3.1 Program oznajmi, kto podpisał plik.

3.3.2 Otrzymany w ten sposób plik (np. USER2.TXT) nie będzie zawierał podpisu (tylko tekst jawny). Korzystając z edytora przejrzeć zawartość tego pliku.

3.4 Odszyfrować plik, tak aby zawarty w nim podpis pozostał nietknięty:

pgp -d zaszyfrowany_plik

(np. **pgp -d USER2.PG4**)

Co się stanie?

3.4.1 Program poprosi o paszport odbiorcy.

3.4.2 Zgłosi komunikat, że pozostawia w pliku podpis:

```
This file has a signature, which will be left in place.
```

3.4.3 Patrz punkt 3.2.2.

Korzystając z edytora przejrzeć zawartość utworzonego pliku.

Zagadnienia:

1. Co stanie się, gdy część odtworzonego pliku - zawierająca wiadomość - zostanie naruszona i plik będzie ponownie szyfrowany-(przesyłany)-odszyfrowywany?
2. Co stanie się, gdy część odtworzonego pliku - zawierająca podpis - zostanie naruszona i plik będzie ponownie szyfrowany-(przesyłany)-odszyfrowywany?
3. Co determinuje powyższe zjawiska i czym one różnią się od siebie?

3.5 Odszyfrować plik:

pgp zaszyfrowany_plik

(np. **pgp USER2.PG5**)

Co się stanie?

3.5.1 Program zapyta o paszport dla IDEA ("Hej, USER 2, jaki dałeś paszport w punkcie 5?").

3.5.2 Patrz punkt 3.2.2.

Korzystając z edytora przejrzeć zawartość utworzonego pliku.

3.6 Korzystając z edytora przejrzeć zawartość dostarczonego pliku.
Następnie odszyfrować informację:

pgp zaszyfrowany_plik

(np. **pgp USER2.AS6**)

1. Patrz punkt 3.1.1.

Ćwiczenie 4 - zastosowanie kompresji

4.1 Używając programu BUDUJ.EXE wygenerować plik BUDUJ.TMP wielkości 128 kB składający się z dowolnej frazy.

Składnia:

BUDUJ ["fraza"]

fraza domyślna - "To jest test! "

4.2 Zaszyfrować plik metodą konwencjonalną (algorytm IDEA) PRZY WŁĄCZONEJ kompresji:

pgp -c buduj.tmp +compress=on

4.3 Zanotować wielkość utworzonego pliku (zawierającego zaszyfrowany tekst).

4.4 Zaszyfrować plik metodą konwencjonalną (algorytm IDEA) PRZY WYŁĄCZONEJ kompresji:

pgp -c buduj.tmp +compress=off

4.5 Zanotować wielkość utworzonego pliku (zawierającego zaszyfrowany tekst)**.**

Zagadnienia:

1. Porównać zanotowane wielkości. Określić stopień kompresji - ile razy wielkość pliku skompresowanego przed zaszyfrowaniem jest większa od pliku nieskompresowanego przed zaszyfrowaniem?
2. Czy są zauważalne różnice w czasie trwania operacji szyfrowania z kompresją i bez kompresji? Jeśli tak, to jaki jest powód tej różnicy?
3. Jakie praktyczne korzyści płyną z zastosowania kompresji?
4. Czy zaszyfrowany plik można skompresować?

Literatura

[Borland] - Borland C++ 3.1 Help,

[RFC1321] - Ron Rivest - The MD5 Message-Digest Algorithm - RFC 1321 - MIT and RSA Data Security, Inc. - April 1992

[Schnei94] - Bruce Schneier - Applied Cryptography - Protocols, Algorithms, and Source Code in C - John Wiley & Sons, Inc. - New York 1994

[Sttali95a] - William Stallings - Protect Your Privacy - A Guide for PGP Users - Prentice Hall PTR - 1995

[Sttali95b] - William Stallings - Network and Internetwork Security - Prentice Hall PTR - 1995

[Szczyp95] - Krzysztof Szczypiorski, Konrad Wrona - Vivaldi -system ochrony informacji - CITCOM - IT PW - Warszawa, czerwiec 1995

[Zimmer93] - Philip Zimmermann - PGP 2.3a User's Guide -14 June 1993

Kontakt z autorami

Wszelkie pytania i uwagi prosimy kierować pod poniższe adresy poczty elektronicznej:

Krzysztof Szczypiorski **<kszczypi@tele.pw.edu.pl>**
http://www.tele.pw.edu.pl/~kszczypi
PGP key fingerprint = 17 3B CB 12 79 C9 5E C8 A6 56 67 CB 6B 8A 6E 24

Konrad Wrona **<kwrona@tele.pw.edu.pl>**
http://www.tele.pw.edu.pl/~kwrona
PGP key fingerprint = 67 E4 E2 C4 0B 4E 9A FB C3 2F 9E E8 5C 21 70 03

Spis treści