Korzystanie z programu Gnu Privacy Guard

Spis treści

1. Szyfrowanie informacji	2
2. Idea pary kluczy publicznego i prywatnego	2
3. Standard OpenPGP i jego implementacje	3
4. Instalacja GnuPG i narzędzi wspomagających	3
5. Generowanie kluczy	5
6. Eksportowanie oraz importowanie kluczy	8
7. Podpisywanie i weryfikacja	15
8. Szyfrowanie i deszyfrowanie	20
9. Sieć zaufania (web of trust)	22
10. Bibliografia	23

1. Szyfrowanie informacji

Już 100 lat p.n.e. powstały pierwsze pomysły na bezpieczną wymianę informacji. Jeden z pierwszych szyfrów został stworzony przez Juliusza Cezara. Opierał się na prostej idei przesuwania znaków o 3 pozycje. Np. znak 'a' był zapisywany jako 'd', 'b' jako 'e', 'z' jako 'b' itd. Oto jak wyglądałaby zaszyfrowany słynny cytat Juliusza Cezara:

Postać oryginalna: Veni Vidi Vici Postać zaszyfrowana: Yhql Ylgl Ylfl

Aby odszyfrować tak zakodowaną wiadomość należy znać klucz jakim była szyfrowana i wykonać działania odwrotne. Szyfrując używano przesunięcia o 3 znaki w prawo, więc deszyfrowanie odbywałoby się przesuwając każdy znak o 3 pozycje w lewo.

Taki algorytm nazywany jest w kryptografii **szyfrowaniem symetrycznym**. Deszyfrowanie polega na odwróceniu działań wykonanych podczas szyfrowania stosując ten sam klucz. W przypadku szyfru Cezara kluczem była liczba 3. Co ciekawe szyfr Cezara może dawać wyniki zupełnie odmienne od oczekiwanych, przykładowo zaszyfrowanie kluczem o wartości 13 łańcucha "*hejnal urwany*" da w wyniku łańcuch "*urwany hejnal*".

Oczywiście szyfr przesuwający zostanie złamany w ciągu ułamków sekund na każdym współczesnym komputerze. Nie nadaje się więc do zastosowania w dzisiejszych warunkach. Ponadto szyfrowanie symetryczne nie nadaje się do wykorzystania w Internecie, ponieważ w takim przypadku zachodziłaby potrzeba bezpiecznej wymiany klucza. Przechwycenie przez intruza pakietu zawierającego klucz dawałoby mu pełny dostęp do szyfrowanych informacji, więc cała idea straciłaby sens. Jak więc bezpiecznie wymienić dane poprzez sieć? Idealnie byłoby gdyby dane można było szyfrować jednym, a deszyfrować tylko innym kluczem. W takim przypadku przechwycenie klucza umożliwiającego szyfrowanie, nie wystarczyłoby, aby odszyfrować informacje. Jak się okazuje istnieje takie rozwiązanie i nazywa się **szyfrowaniem asymetrycznym**.

2. Idea pary kluczy publicznego i prywatnego

Zasada działania szyfrowania asymetrycznego opiera się na idei dwóch kluczy, publicznego i prywatnego. Kluczem publicznym można tylko szyfrować, a kluczem prywatnym można deszyfrować. Jak wskazują nazwy klucz prywatny powinien być dostępny tylko dla osoby, która może czytać szyfrowane wiadomości, natomiast klucz publiczny może być dostępny dla każdego, kto zechce zaszyfrować informacje. Idea szyfrowania asymetrycznego opiera się na złożoności danych działań matematycznych względem działań do nich odwrotnych. Pomnożenie dwóch dużych liczb pierwszych jest dużo mniej wymagające obliczeniowo, niż uzyskanie czynników z tak uzyskanego iloczynu. Weźmy dwie 128-bitowe liczby pierwsze, przeciętny komputer w ciągu niespełna sekundy dokona mnożenia takich liczb i zwróci liczbę 256-bitową. Odwrócenie działania, czyli uzyskanie z liczby 256-bitowej dwóch liczb 128-bitowych, będzie dużo bardziej wymagające. Klucz publiczny może być właśnie tak powstałą liczbą 256-bitową, natomiast klucz prywatny dwiema liczbami 128-bitowymi, z których powstał klucz publiczny.

Przykładowo rozłożenie na czynniki pierwsze liczby 15 jest proste: 3 5, natomiast liczba 91723948197324 sprawi już trochę więcej kłopotu, tym bardziej liczba 155- czy 200-cyfrowa. Rozłożenie liczby na czynniki pierwsze składa się z pewnej ilości kroków, która to ilość zwiększa się wykładniczo wraz ze wzrostem wielkości liczby. Oznacza to, ze jeśli liczba jest wystarczająco

duża, jej faktoryzacja może zająć lata.

Najbardziej istotne jest to, że klucz deszyfrujący (prywatny) znacząco różni się od klucza szyfrującego (publicznego) oraz że dla odpowiednio długiego klucza publicznego nie jest możliwe, w rozsądnym czasie, wygenerowanie klucza prywatnego.

3. Standard OpenPGP i jego implementacje

Wraz z rozwojem Internetu oraz związanych z nim niebezpieczeństw i brakiem prywatności, szyfrowanie przesyłanych danych nabrało szczególnego znaczenia.

W 1991 roku Phil Zimmerman zapoczątkował projekt PGP - program umożliwiający szyfrowanie i cyfrowe podpisywanie wszelkiego rodzaju dokumentów elektronicznych. PGP umożliwiał bezpieczną wymianę oraz autentykację wszelkiego rodzaju dokumentów elektronicznych. Początkowo był dostępny wraz kodem źródłowym, począwszy jednak od wersji 5.0 stał się oprogramowaniem zamkniętym i komercyjnym.

Z początkiem 1997 roku została powołana w ramach IETF (Internet Engineering Task Force) grupa, która zdefiniowała otwarty standard szyfrowania poczty elektronicznej z wykorzystaniem kryptografii klucza publicznego, bazując na zamkniętym programie PGP i zapewniając kompatybilność z nim. Tak powstał standard OpenPGP, opisany w dokumencie RFC 2440. Określa on wzorcowe formaty zaszyfrowanych wiadomości i podpisów. Od tej pory praktycznie każdy może stworzyć własną implementację standardu i wymieniać informacje z użytkownikami innych, jak choćby PGP.

Najpopularniejszą otwartą implementacją protokołu OpenPGP jest Gnu Privacy Guard (GPG). Choć oferuje nieco mniejsze możliwości niż komercyjny PGP, znakomicie nadaje się do zastosowań w małych firmach oraz wśród użytkowników domowych.

4. Instalacja GnuPG i narzędzi wspomagających

Gnu Privacy Guard w wersji w przygotowanej na platformę Windows można pobrać ze strony projektu <u>http://www.gpg4win.org/</u>. W pakiecie oprócz GnuPG znajduje się kilka aplikacji ułatwiających korzystanie z niego:

•GPGee – rozszerzenia systemowego menu podręcznego

•WinPT – narzędzie do zarządzania kluczami

Proces instalacji nie sprawia kłopotu. Jednak aby móc korzystać z narzędzi w powłoce systemowej należy dodać ich lokalizację do zmiennej systemowej PATH. W tym celu przejdźmy do Panel sterowania -> System -> Zaawansowane -> Zmienne środowiskowe -> Zmienne systemowe

	2		1000	
Przywracanie systemu	Aktualizacje auto	matyczne	Zdalny	
Ogólne Nazwa kom	putera Sprzę	Zaa	wansowane	
Aby móc przeprowadzić więł Administrator, Wydajność	(szość tych zmian, mu	sisz zalogowa Zmienne śr	ć się jako odowiskowe	2 ×
pamięć wirtualna	e uzycia procesora, v	Zmienneu	użytkownika dla	sensei
		Zmienn	a W	/artość
		TEMP	C:	\Documents and Settings\sensei\Usta
- Profile użytkownika		TMP	C:'	\Documents and Settings\sensei\Usta
Hetawienia pulpitu powiaza	ne z logowaniem użul			Edytowanie zmiennej systemowej
-Uruchamianie i odzyskiwani Informacje o uruchamianiu	e systemu, awariach sy	Zmienne s Zmienn Comspe	ystemowe	Nazwa zmiennej: Path Wartość zmiennej: em32\Wbem;C:\Program Files\GNU\Gnu Vart OK Vart OK
Zmier	ne środowiskowe	NUMBER OS Path	0051_c NC 2_OF_P 1 Wi C!	ý indows_NT \WINDOWS\system32;C:\WINDOWS; ▼ N <u>o</u> wa E <u>d</u> ytuj U <u>s</u> uń
_		<u> </u>		OK Anuluj

Rys. 1 Ustawianie zmiennej systemowej PATH.

Jeśli wszystko wykonaliśmy jak należy, otwórzmy powłokę systemową *Start -> Uruchom -> cmd.exe.* Po wpisaniu:

gpg --version

powinniśmy ujrzeć numer wersji zainstalowanego GPG oraz wspierane algorytmy kryptograficzne. Możemy już więc rozpocząć właściwą pracę z GnuPG.

C:\WINDOWS\system32\cmd.exe	- 🗆 🗙
Microsoft Windows XP [Wersja 5.1.2600] (C) Copyright 1985-2001 Microsoft Corp.	*
C:\Documents and Settings\sensei>gpgversion gpg: NOTE: old default options file `C:/Documents and Settings/sensei/Dane a acji/gnupg\options' ignored gpg (GnuPG) 1.4.5 Copyright (C) 2006 Free Software Foundation, Inc. This program comes with ABSOLUTELY NO WARRANTY. This is free software, and you are welcome to redistribute it under certain conditions. See the file COPYING for details.	aplik
Home: C:/Documents and Settings/sensei/Dane aplikacji/gnupg Supported algorithms: Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224 Compression: Uncompressed, ZIP, ZLIB, BZIP2	
C:\Documents and Settings\sensei>	
	-

Rys. 2 Wynik wywołania polecenia gpg –version.

5. Generowanie kluczy

Kiedy utworzymy parę kluczy, zarówno publiczny jak i prywatny będą przechowywane na dysku naszego komputera. Stwarza to pewne ryzyko – każdy kto uzyska dostęp do naszego komputera może zdobyć nasz klucz prywatny i odszyfrować nasze wiadomości lub podszyć się pod nas. Próba zapamiętania klucza w pamięci również skazana jest na niepowodzenie – jest on po prostu bardzo długi. Praktycznym rozwiązaniem tego problemu jest zaszyfrowanie klucza prywatnego hasłem. Przy każdej próbie użycia klucza prywatnego gpg zapyta o nie, a po poprawnym podaniu zdeszyfruje klucz w pamięci i skorzysta z niego. Należy dbać by hasło miało odpowiednią długość i złożoność – na pewno złym pomysłem będzie używanie haseł słownikowych. Zaleca się hasła zawierające małe i duże litery, cyfry oraz znaki specjalne o długości conajmniej ośmiu znaków.

Aby utworzyć parę kluczy, w powłoce systemowej wpiszmy:

gpg --gen-key

Zostaniemy poproszeni o wybranie rodzaju klucza. Wybierzmy domyślny DSA and Elgamal.

Następne pytanie dotyczy długości klucza. W tym miejscu również zaleca się wybranie wartości domyślnej. Wartość ta powinna być kompromisem – krótszy klucz łatwiej złamać, dłuższy klucz będzie wymagał więcej czasu na każdą operację, która go wykorzystuje.

Wzrost mocy obliczeniowej komputerów oraz rozwój badań w dziedzinie kryptoanalizy powoduje, że wiele algorytmów, uważanych niegdyś za niemożliwe do złamania w krótkim czasie, zostało skompromitowanych w ostatnich latach. Na przykład w ostatnim czasie złamano algorytm RSA o długości klucza 576 bitów (100 maszyn w 3 miesiące).

Kolejne pytanie dotyczy długości czasu ważności klucza. Ustawienie wygaśnięcia klucza po pewnym czasie zwiększa nieco bezpieczeństwo – po przekroczeniu daty ważności nie będzie można z niego korzystać. Wiąże się z tym pewna niedogodność – będziemy musieli rozesłać nasz nowy klucz publiczny wszystkim, którzy dotychczas korzystali ze starego. Wybierzmy więc opcję $0 = key \ does \ not \ expire - klucz \ nie \ wygasa \ nigdy.$

C:\WINDOWS\system32\cmd.exe

```
- 🗆 🗙
 C:\Documents and Settings\sensei>gpg --gen-key
gpg: NOTE: old default options file `C:/Documents and Settings/sensei/Dane aplik
acji/gnupg\options' ignored
gpg (GnuPG) 1.4.5; Copyright (C) 2006 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.
Please select what kind of key you want:

(1) DSA and Elgamal (default)

(2) DSA (sign only)

(5) RSA (sign only)

Your selection? 1

DSA keypair will have 1024 bits.

ELG-E keys may be between 1024 and 4096 bits long.

What keysize do you want? (2048)

Requested keysize is 2048 bits

Please specify how long the key should be valid.

0 = key does not expire

<n> = key expires in n days

<n> = key expires in n weeks

<n> = key expires in n weeks

<n> = key expires in n years

Key is valid for? (0)

Key does not expire at all

Is this correct? (y/N) y
 You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"
 Real name: F
Email address: p
                                                                    Comment:
 You are using the 'CP852' character set.
You selected this USER-ID:
                                                                                                                       __>"
                                                                                       a Cu
  Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.
 passphrase not correctly repeated; try again.
passphrase not correctly repeated; try again.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
 generator a better chance to gain enough entropy.
  +++++...>+++++.............++++++
```

Rys. 3 Generowanie pary kluczy.

Teraz program zapyta nas o naszą tożsamość i adres email. Są to informacje, które pozwolą innym zidentyfikować do kogo należy klucz. Jeśli troszczymy się o naszą prywatność, nie musimy podawać naszych prawdziwych danych, zamiast tego możemy wybrać nazwę unikalną na tyle, by nasz klucz nie został pomylony z należącym do kogoś innego. Możemy również uzupełnić pole komentarz lub pozostawić je puste.

Ostatnia rzecz o jaką zapyta proces generowania klucza jest hasło, którym będziemy go chronić. Podczas podawania hasła nie drukuje się ono na ekranie (ani żadne inne znaki sygnalizujące wpisywanie).

Po podaniu danych rozpoczyna się właściwy proces generowania klucza. Aby był on wygenerowany w sposób losowy potrzebna jest duża ilość entropii. Podczas trwania tego procesu zaleca się by intensywnie wykorzystywać dysk twardy lub aktywnie przeglądać strony internetowe. Aby dostarczyć odpowiednia entropie można włączyć kopiowanie dużego pliku, z różnymi

odstępami czasowymi naciskać klawisze i losowo przesuwać kursor myszy.

Po wygenerowaniu nasza para kluczy znajduje się w katalogu:

C:\Documents and Settings\%USERNAME%\Dane aplikacji\GnuPG

Klucz publiczny to *pubring.gpg* a prywatny *secring.gpg*. Ten drugi należy szczególnie chronić. Skopiujmy go już teraz na zewnętrzny nośnik i umieśćmy w bezpiecznym miejscu.

Wygenerowane klucze należy po zakończonych zajęciach bezwzględnie usunąć z komputerów laboratoryjnych!

Alternatywnie, tą samą operację generowania klucza możemy wykonać korzystając z graficznej nakładki do zarządzania kluczami, która instaluje się wraz z pakietem *Gpg4win*, to jest *WinPT*. Odnajdźmy ją w menu *Start* i uruchommy. Aplikacja uruchamia się w zasobniku systemowym. Przy pierwszym użyciu, jeśli nie istnieją jeszcze żadne klucze zostaniemy zapytani czy chcemy taką parę stworzyć. Wybierzmy wtedy *Generate a GnuPG key pair*. Tak jak poprzednio zostaniemy poproszeni o podanie naszej tożsamości, adresu email oraz hasła chroniącego klucz prywatny. Ponadto zostaniemy zapytani czy chcemy utworzyć kopię zapasową świeżo wygenerowanych kluczy, a jeśli tak, program pyta o ich docelową lokalizację. Nie mamy wpływu na ważność, długość oraz rodzaj klucza – zostaną ustawione sensowne wartości domyślne.

W razie potrzeb możemy wygenerować klucz w trybie *Expert* (menu w WinPT, *Key* -> *New* -> *Expert*) i uzyskać dostęp do wspomnianych wyżej opcji klucza.

🔍 Key Manager								_ = ×
File Edit View	Key Groups Keyserver ?	8	е.					
	New	Normal						
User ID	Edit Sign Delete Revoke Cert List Signatures List Trust Path	Expert Smartcard UXF 13606000	Type pub/sec	Size 1024/2048	Cipher DSA/ELG	Validity Ultimate	Trust Ultimate	Creation 2006-10-15
	Import via HTTP Import Export Export Secret Key							
	Properties Refresh Keys (Keyserver) Reload Key Cache Reverify Signatures							
Default Key: F1960	BDD	1 secret keys			1 keys			

Rys. 4 Generowanie klucza z nakładki WinPT w trybie Expert.

6. Eksportowanie oraz importowanie kluczy

Wygenerowany klucz publiczny jest bezużyteczny jeśli nie dostarczymy go osobom, z którymi chcemy wymieniać szyfrowane informacje. Możemy więc umieścić go na naszej stronie internetowej lub wysłać pocztą elektroniczną. Jeszcze innym, a prawdopodobnie najpraktyczniejszym, sposobem jest umieszczenie klucza na specjalnym serwerze, który służy do ich przechowywania.

By wyeksportować klucz do formatu tekstowego użyjemy w konsoli systemowej polecenia:

gpg --armor --output "klucz.txt" --export "NAZWA"

gdzie w miejsce nazwa wpisujemy adres email wpisany przy generacji klucza lub naszą 'nazwę' która identyfikuje klucz (np. imię i nazwisko). Możemy także podać część nazwy jeśli jednoznacznie wskazuje ona klucz (np. jedynie nazwa domeny adresu email jeśli jest unikatowa w zbiorze kluczy).

Tak spreparowany klucz jest gotowy do publikowania. Ważne jest by umieszczać go w całości (razem z komentarzami, od początkowego do końcowego):

-----BEGIN PGP PUBLIC KEY BLOCK-----

-----END PGP PUBLIC KEY BLOCK-----



Rys. 5 Wyeksportowany klucz w postaci znaków ASCII.

By dokonać tego samego z nakładki WinPT należy w oknie głównym zaznaczyć interesujący nas klucz, a następnie wybrać bądź to z menu *Edit -> Copy* bądź z menu podręcznego (prawy klawisz myszy) *Copy Key to Clipboard*. Wybrany klucz publiczny znajduje się w schowku.

Serwery kluczy potrafią przechowywać klucze publiczne i udostępniać je każdemu, kto o nie poprosi. Jest to rozwiązanie wygodne, zwłaszcza jeśli chcemy rozdystrybuować klucz większej liczbie osób. Istnieje wiele serwerów kluczy, jednak nie musimy wysyłać naszego klucza do każdego z nich. Zdecydowana większość, a przynajmniej popularne serwery, synchronizuje się ze sobą. W efekcie po pewnym czasie odpytanie któregokolwiek z nich o wyeksportowany klucz zwróci spodziewany wynik.

Procedura wyeksportowania klucza na serwer jest prosta. W konsoli systemowej wystarczy w tym celu użyć polecenia:

gpg --keyser SERWER_KLUCZY --send-keys ID_KLUCZA

gdzie ID_KLUCZA to jego identyfikator, a SERWER_KLUCZY jest nazwą domenową lub adresem wybranego serwera kluczy.

🔍 Key Manager								_ = ×
File Edit View Key Groups	Keyserver	?						
	1 💼 🖻							
User ID		Key ID	Туре	Size	Cipher	Validity	Trust	Creation
		▶ 0xC235493E	pub/sec	1024/1024	DSA/ELG	Ultimate	Ultimate	2006-10-15
	- @	Key Attributes	•	Copy Use	r ID to Clipb	oard	Ultimate	2006-10-15
	-	Copy Key to Clipbo Paste Key from Clip Send Key to Mail Re	ard board acipient	Copy E-M Copy Key Copy Fin Copy Key	Copy E-Mail to Clipboard Copy Key ID to Clipboard Copy Fingerprint to Clipboard Copy Key Info to Clipboard			
		Add Key Edit	•				F.	
		Sign Revoke Cert Delete List Signatures Enable Disable Set Implicit Trust						
Default Key: 0x538E5975		Set preferred Keyse Refresh from Keyse Send to Keyserver	erver URL erver	-	2 keys			1
		Set as Default Key						
		Properties						

Rys. 6 Eksport klucza publicznego do schowka w programie WinPT.

Listę wszystkich kluczy wraz z identyfikatorami uzyskamy wywołując:

gpg --list-keys

🖏 Konsola	
C:\WINDOWS>gpg ——list-key gpg: checking the trustdb gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u C:/Documents and Settings/sensei/Dane aplikacji/gnupg\pubring.gpg	
pub 1024D/E170D45B 2006-12-14 uid Jan Kowalski (Janek) (jan.kowalski@mail.pl) sub 2048g/3AA7ABB3 2006-12-14	
C:\WINDOWS>	
	-

Rys. 7 Listowanie wszystkich dostępnych kluczy. W ramce ID klucza.



Rys. 8 Eksport klucza na serwer kluczy.

Operację eksportu klucza na serwer można dokonać także za pomocą interfejsu WinPT. W tym celu otwórzmy menu podręczne na interesującym nas kluczu i wybierzmy pozycję *Send to keyserver* i dowolny serwer z listy. Jej zawartość można modyfikować w menu *Keyserver*.

	🔍 Key Ma	nager	-								
	File Edit	View Key	aroups Keyserv	/er <u>?</u>							
		' 🖪 🔎 l	j 🔒 🖻	i.							
	User ID			(Key ID	Туре	Size	Cipher	Validity	Trust	Crea
Key Attributes	∞əllan Ko ►	walski (Janek)	<jan.kowalski@n< th=""><th>nail.pl></th><th>0xE170D45B</th><th>pub/sec</th><th>1024/2048</th><th>DSA/ELG</th><th>Ultimate</th><th>Ultimate</th><th>200</th></jan.kowalski@n<>	nail.pl>	0xE170D45B	pub/sec	1024/2048	DSA/ELG	Ultimate	Ultimate	200
Copy Key to Clipboard Paste Key from Clipbo Send Key to Mail Recip	ard ient										
Add Key Edit	٠										
Sign <u>R</u> evoke Cert Delete List Signatures											
Enable											•
Disable Set Implicit Irust		E170D458		1 sec	ret keys			1 keys			
Set preferred Keyserv Refresh from Keyserve	er URL er										
Send to Keyserver	•	http://subk	eys.pgp.net								
Set as Default Key		http://subk	eys.pgp.net								
Properties		hkp://subke Idap://keys	ys.pgp.net erver.pgp.com								

Rys. 9 Eksport klucza na serwer w interfejsie WinPT.

Jeśli chcemy przesłać jakiejś osobie zaszyfrowaną wiadomość, to także musimy zdobyć jej klucz publiczny. Jeśli dysponujemy już takim, zapisanym w pliku tekstowym to by go zaimportować do naszego GPG wystarczy zrobić:

gpg --import zapisany_klucz_publiczny.txt

Naturalnie należy podać poprawną ścieżkę do pliku lub musimy znajdować się w katalogu gdzie jest klucz. Co ciekawe w pliku zawierającym klucz mogą znajdować się także inne dane – wczytany zostanie jedynie blok pomiędzy znacznikami.

----- BEGIN PGP PUBLICKEY BLOCK-----

----- END PGP PUBLICKEY BLOCK-----



Rys. 10 Import klucza zapisanego w pliku tekstowym.

Te same operacje wykonamy z nakładki WinPT. Odpowiednią opcję służącą do importu obcego klucza publicznego z pliku tekstowego znajdziemy w menu *Key -> Import*. Tutaj mamy jeszcze możliwość zaimportować klucz wprost czyjejś witryny www – wybierzmy wtedy *Key -> Import via HTTP*.

🔍 Key Manager								
File Edit View	Key Groups Keyserver ?							
	New	•						
User ID	Edit Sign	Key ID 0x69383D7E	Type	Size 1024/2048	Cipher DSA/ELG	Validity None	Trust None	Creation 2006-10-15
	Delete Revoke Cert List Signatures List Trust Path	0xC235493E 0x53BE5975 0x56A1520D	pub/sec pub/sec pub	1024/1024 1024/2048 1024	DSA/ELG DSA/ELG RSA	Ultimate Ultimate None	Ultimate Ultimate None	2006-10-15 2006-10-15 1995-04-24
	Import via HTTP							
	Import							
	Export Export Secret Key							
	Properties							
Refr Relo Rev	Refresh Keys (Keyserver) Reload Key Cache Reverify Signatures							

Rys. 11 Import klucza zapisanego w pliku tekstowym z poziomu WinPT.

Importowanie klucza z serwerów jest znacznie wygodniejsze, nie wymaga uprzedniego dostarczenia klucza w formacie tekstowym. Najpierw musimy zlokalizować interesujący nas klucz. Załóżmy, że chcielibyśmy nawiązać bezpieczną korespondencję z Janem Nowakiem. Odnajdźmy najpierw jego klucz:

gpg --keyserver SERWER_KLUCZY --search-key adres_email

Gdy odnajdziemy właściwy klucz i poznamy jego ID, możemy go zaimportować:

gpg --keyserver SERWER_KLUCZY --recv-key ID_KLUCZA



Rys. 12 Wyszukanie i import klucza z serwera.

Ta sama operacja w nakładce WinPT sprowadza się do wywołania menu *Keyserver*. Uruchomi się wtedy nowe okienko w którym możemy wpisać szukany przez nas klucz. Po odnalezieniu na serwerze zostanie utworzone nowe okienko, z którego możemy dokonać już importu.

Popularne serwery kluczy: •pgp.mit.edu •subkeys.pgp.net •wwwkeys.eu.pgp.net •keyserver.pgp.com •keyserver.veridis.com

Key Mana File Edit Vi	ger w Key Groups K	evserver ?		-		-			IX
🗢 😜 😭	3 🔎 😂 🔳 🗖	, b							
User ID			Key ID	Туре	Size	Cipher	Validity	Trust	Crea
😂 Jan Kowa	lski (Janek) <jan.kowa< td=""><td>lski@mail.pl></td><td>0xE170D45B</td><td>pub/sec</td><td>1024/2048</td><td>DSA/ELG</td><td>Ultimate</td><td>Ultimate</td><td>200</td></jan.kowa<>	lski@mail.pl>	0xE170D45B	pub/sec	1024/2048	DSA/ELG	Ultimate	Ultimate	200
		- VI							
Keyserver Access									
DNS Name	Port								
http://pgp.mit.edu	11371 11271								
hkp://subkevs.pap.net	11371								
http://subkeys.pgp.net	11371								
] Catidat	sult Change prov							1	۰
serger	auk <u>Ch</u> ange proxi		a succe barrow			Leve			
			ecrec keys	_	_	I Keys	_	_	
Key ID or email address you	want to search for	10							
Jan.nowak@mail.pl		<u>- 11</u>							
Receive Search	Close							104-3	1
	Keyserver Search	ing						×	1
	Connect to "http://s	ubkeys.pgp.net	' to search for ''ja	n.nowak@	mail.pl''				
	Size Algori	Key ID	Creation	UserID					
	1024 DSA	0xFA9175FB	2006-12-14	Jan Now	ak <jan.nowal< td=""><td>k@mail.pl></td><td></td><td></td><td></td></jan.nowal<>	k@mail.pl>			
	2						-		
						<u>R</u> eceive		ancel	

Rys. 13 Wyszukanie i import klucza z serwera za pomocą WinPT.

Jest jeszcze jedna czynność, którą powinniśmy wykonać po zaimportowaniu. Taki klucz domyślnie nie należy do grupy zaufanych. Jeśli będziemy z niego korzystać, otrzymamy ostrzeżenia, że jego autentyczność nie została potwierdzona. Jeśli jednak jesteśmy przekonani co do jego pochodzenia, to by pozbyć się ostrzeżeń wystarczy ustawić odpowiedni poziom zaufania względem niego. W konsoli wpiszmy:

gpg --edit-key NAZWA_KLUCZA

Wejdziemy w tryb interaktywny. Wydajmy w nim polecenie:

trust

Zostaniemy poproszeni o wybranie poziomu zaufania. Wybierzmy opcję przedostatnią, tj. 5 = I trust ultimately.

Aby wyjść z trybu interaktywnego i zapisać wprowadzone zmiany wpiszmy:

quit

Alternatywnie ta sama czynność w WinPT polega na wybraniu klucza z listy, a po uruchomieniu na nim menu podręcznego – wybraniu opcji *Set implicit trust*. Pełną funkcjonalność interaktywnego trust znajdziemy pod opcją *Key edit* w tym samym menu.



Rys. 13 Ustawianie pełnego zaufania dla zaimportowanego klucza.

7. Podpisywanie i weryfikacja

Dzięki OpenPGP otrzymujemy mechanizm weryfikacji poprawności i autorstwa istotnych dla nas danych. Wystarczy, że nadawca podpisze cyfrowo swoim kluczem prywatnym dane, których autentyczność sprawdzimy my, korzystając z jego klucza publicznego. Procedura wygląda następująco – obliczania jest wartość funkcji mieszającej dla danego zestawu danych, następnie ta wartość jest szyfrowana kluczem prywatnym nadawcy. Można ją zdeszyfrować tylko kluczem publicznym tego nadawcy. To upewnia nas o pochodzeniu. Po zdeszyfrowaniu możemy porównać wartość funkcji mieszającej nadawcy z wynikiem takiego haszowania u nas. Jeśli są zgodne, to znaczy że dane nie uległy zmianie. Mamy wtedy już całkowitą pewność, że to co otrzymaliśmy, po przejściu przez kanał komunikacyjny, zgadza się z oryginałem.

Aby podpisać plik wystarczy w konsoli wpisać:

gpg --clearsign PLIK

Zostanie do tego użyty nasz domyślny klucz. Program poprosi nas o podanie naszego hasła do klucza, a wynikiem jego działania będzie plik PLIK.asc. Może on wyglądać następująco:



Rys. 14 Podpisywanie wiadomości wybranym kluczem. Listing wiadomości.

Jeżeli chcemy użyć do podpisu innego klucza niż nasz domyślny, wtedy dołączmy parametr --localuser. Jako argument przyjmuje nazwę wybranego klucza, tj. adres email wpisany przy generacji klucza, imię i nazwisko lub identyfikator.

gpg --local-user NAZWA --clearsign PLIK

Powyższe sposoby sprawdzają się kiedy podpisujemy pliki tekstowe. Jednak taka modyfikacja jak dodanie popisu do pliku binarnego uszkodzi go. Wtedy powinniśmy stosować podpis cyfrowy w osobnym pliku:

gpg --output PLIK_PODPISU.sig --detach-sign PLIK

Utworzony podpis będzie w formacie binarnym, nieczytelnym dla człowieka. Jeśli chcemy by podpis był w formacie pliku ASCII dodajmy przełącznik

--armor



Rys. 15 Podpisywanie pliku binarnego. Podpis w formacie tekstowym w osobnym pliku.

Weryfikacja popisanego pliku jest również prosta. Jeżeli posiadamy klucz publiczny osoby podpisującej, to po wpisaniu w konsoli:

gpg --verify **PODPIS** (*lub plik tekstowy z załączonym podpisem*)

otrzymamy potwierdzenie autentyczności, jeśli plik nie uległ zmianie. Jeśli nie posiadamy w naszym zbiorze klucza publicznego osoby podpisującej, musimy go wcześniej zaimportować.

Wszystkie operacje podpisywana i weryfikacji można uprościć stosując rozszerzenie powłoki systemowej Windows, *GPGee*. Dzięki niemu wystarczy wybrać odpowiednią opcję z menu kontekstowego pliku. Prześledźmy proces podpisywania (rys. 17, 19) i weryfikacji pliku (rys. 18, 20).



Rys. 16 Podpisywanie pliku binarnego. Weryfikacja podpisu.



Rys. 17 Podpisywanie pliku.

Rys. 18 Weryfikacja podpisu.

Sign/Encrypt Files	You need a passphrase to unlock the following secret key:					
lame						
Beltarar shultararddesy	ID: E1700436 Type: D3A Size: 1024 Date: 2006/12/14					
Jan Kovalski -jan.koval Jan Novak -jan.novak@ma	Enter passphrase: 🔽 Hide Typing					
an fatzi uzez	Ok Cancel					
y choups.						
gning Keys: Jan Kowalski (Janek) <ja< td=""><td>n.kowalski@mail.pl> (DSA/0xE170D45B)</td></ja<>	n.kowalski@mail.pl> (DSA/0xE170D45B)					
Signature Options	ncryption Options Misc. Options					
None Gearsian C	None Image: Text output (ASCII Armor) Public-key Image: Text input (textmode - see help) Make source files read only					

Rys. 19 Podpisywanie pliku. Zaznaczona opcja podpisu w osobnym pliku. tekstowym.

autentykttxt autentykttx	
Filename	Туре
✓C:\\sensei\Pulpit\aut	entyk.txt.asc Sig
Good SHA1-hashed signature ma (DSA/0xDFC83996) on 2006-12-1 WARNING: This signature was cre	de by Farmer (marked) < (marked) (marked) 7 12:02:58. ated with a key of "undefined" validity.

Rys. 20 Weryfikacja podpisu. Należy pamiętać o uprzednim zaimportowaniu klucza publicznego.

8. Szyfrowanie i deszyfrowanie

Standard OpenPGP zapewnia wysoką ochronę prywatności umożliwiając szyfrowanie wiadomości i danych kryptografią asymetryczną.

Plik zaszyfrujemy z konsoli w następujący sposób:

gpg --recipient ODBIORCA --output PLIK.gpg --encrypt PLIK

Nowo powstały plik binarny *PLIK.gpg* zawiera zaszyfrowane dane kluczem publicznym odbiorcy. Tylko on będzie mógł do zdeszyfrować swoim kluczem prywatnym.

Zaszyfrowany plik wynikowy może być również tekstowy, jeśli jego rozmiar nie jest zbyt wielki:

gpg --armor --recipient ODBIORCA --output PLIK.asc --encrypt PLIK

🖾 Konsola	- 🗆 ×
C:\>gpgrecipient "Jan Nowak"output autentyk.txt.gpgencrypt autentyk.	.txt
C:\>gpgarmorrecipient "Jan Nowak"output autentyk.txt.ascencrypt a ntyk.txt	aute
C:\>type autentyk.txt.asc BEGIN PGP MESSAGE Version: GnuPG v1.4.5 (MingW32)	
hQIOA5Deoqej4xXoEAf8CotwySnrs6mUbaxtf73RUb6TqUTagWFJ8kHSeWcgszUd uwurA6Ti7vvhlizFdJu8e5mfyEYKUdH1M4ScUcE7C9uZSSfWBh3Q3sWNm+yzRiRK CUmv8DdfvnnGgEcY3tDXUrfQeeIØyeG8EcT3E9jinYHGLfAØL67v7qb5EdCOOLJx HgaipC9SEkYyBkwed6kAMfP7msCWHcH2pHYAAnDs1cRuJmzFJg4L5UM7QYoKEXR3 RkRUXZTDJGJUHjoLXepy3AA7fYox7fMhD+qIn3Y72Va6hA2900eKKIØFYrp/7X6v H1oqfJb+biiZwMH55LfkRJmfOU?f+8sP8NED5BuNzQf/d/q57eW81DxWEVIPWL2J Kb/SH12vsX4maZhUzxPvPjOGj21nxrkGrI8BKAtMiv4Ax32gmf0MjUepgNAUU1Zw OØGKXZku5DUdF4G8BxzeQSMiXrAcnGIwWBhQ1/bXHN/RhpJØJaxlanwwtQn0BGu7 aH13MJuXbqHdcu806CsQSDksusNEleOWHZZS6mDWeKiXJb2S1IMSSeG8TZEWH4y/ /iQwB8jzM6KzUuYUjD9keBItia3Hc8sqPH/cU5pxUGJTUZI870d01h4p1bB/c0AD GX10RBSMgGZdZk927eUfHjnhwNiIK24D51SH1BJMpgHTYCizq9BcLPKKjvxqNNPN CtJvAb7iwjkgSpnVIkBØJydDBCEZK4qJXneZUqi8tcCsjROQUBAvMobnjLsSqUtD z5RgdpQKs8U2cjxVvtIL3sj+30S5MCcVqP1Xt8JL5d4MV2ca2UAxoczd86FwsM8I w1H5nYcTPAYzaGE9BkOCgbxV =KHsc END PGP MESSAGE	
C:\>_	

Rys. 21 Szyfrowanie pliku. Postać tekstowa po zaszyfrowaniu.

Czasem chcemy przekazać szyfrowaną wiadomość więcej niż jednej osobie. Stwarza to pewien problem, gdyż dzielenie klucza prywatnego nie wchodzi w rachubę, a szyfrowanie dla każdego z osobna nieco kłopotliwe. GnuPG pozwala nam jednak zakodować wiadomość dla więcej niż jednego odbiorcy jednocześnie. Wtedy jeden z ich kluczy prywatnych wystarcza do rozszyfrowania, więc każdy może to zrobić indywidualnie. Składnia tej operacji różni się jedynie dodaniem większej ilości odbiorców --recipient :

gpg --armor --recipient ODBIORCA_1 --recipient ODBIORCA_2 --output PLIK.asc -encrypt PLIK

Teraz spróbujmy taki zaszyfrowany plik odkodować. Jeśli był on adresowany do nas, musimy posiadać odpowiedni klucz prywatny.

Poniższe polecenie :

gpg --decrypt-files PLIK.gpg

spowoduje zdekodowanie zaszyfrowanego pliku. Po podaniu hasła do klucza plik rozkodowany plik zostanie umieszczony w bieżącym katalogu pod nazwą *PLIK*.

Jeśli szyfrowany plik był wiadomością tekstową, można użyć alternatywnie polecenia:

gpg --decrypt PLIK.asc

Nie wypakuje ono pliku z zaszyfrowanej postaci, lecz wyświetli zawartość na ekranie.

Monsola
C:\>gpg --decrypt autentyk.txt.asc
You need a passphrase to unlock the secret key for
user: "Jan Nowak (jan.nowak@mail.pl>"
2048-bit ELG-E key, ID A3E315E8, created 2006-12-14 (main key ID FA9175FB)
gpg: encrypted with 2048-bit ELG-E key, ID A3E315E8, created 2006-12-14
"Jan Nowak (jan.nowak@mail.pl>"
Quidquid latine dictum sit, altum videtur.
C:\>gpg --decrypt-files autentyk.txt.gpg
You need a passphrase to unlock the secret key for
user: "Jan Nowak (jan.nowak@mail.pl>"
2048-bit ELG-E key, ID A3E315E8, created 2006-12-14
"Jan Nowak (jan.nowak@mail.pl>"
2048-bit ELG-E key, ID A3E315E8, created 2006-12-14 (main key ID FA9175FB)
gpg: encrypted with 2048-bit ELG-E key, ID A3E315E8, created 2006-12-14
"Jan Nowak (jan.nowak@mail.pl>"
C:\>type autentyk.txt
Quidquid latine dictum sit, altum videtur.
C:\>gp autentyk.txt

Rys. 22 Deszyfrowanie pliku.

Podobnie jak przy podpisywaniu tak i operacje szyfrowania/ deszyfrowania można przeprowadzić za pomocą rozszerzeń powłoki *GPGee*. Proces jest bardzo podobny, wystarczy tylko zmienić używane opcje.

9. Sieć zaufania (web of trust)

Posiadanie kluczy publicznych daje możliwość weryfikacji nadawcy przesyłki. Otrzymując podpisanego maila od Jana Kowalskiego mamy pewność, że to właśnie Jan Kowalski jest nadawcą przesyłki. Jest jednak pewien szczegół, który do tej pory pomijaliśmy. Załóżmy następujący scenariusz. Dowolna osoba generuje parę kluczy. Tworzy e-maila i w polu nadawca wpisuje "Jan Kowalski", po czym podpisuje przesyłkę wygenerowanym kluczem prywatnym i wysyła go do nas. Udostępnia wygenerowany niedawno klucz publiczny na serwerze kluczy. My odbieramy tak spreparowanego maila. Nieświadomi niebezpieczeństwa pobieramy klucz publiczny z serwera, aby sprawdzić autentyczność nadawcy. Oczywiście weryfikacja przebiega pomyślnie, choć nie wiemy, że zostaliśmy oszukani.

Standard OpenPGP definiuje, a GnuPG implementuje, rozwiązanie tego problemu. Cała idea opiera się na podpisywaniu kluczy publicznych i przechodniości zaufania.

Budowanie sieci zaufania musi zacząć się od minimum dwóch osób, które znają identyfikatory swoich kluczy. Niech takimi osobami będą osoba **A** i osoba **B**. Załóżmy, że osoba **C** również pragnie, aby osoby do których wysyła maile mogły sprawdzić czy faktycznie pochodzą od niej. Oczywiście w tym celu komendą gpg –gen-key generuje klucz prywatny i publiczny. Jednak ma świadomość, że to nie wystarczy, aby odbiorcy jego poczty mogli być pewni jej autentyczności. W takim wypadku prosi osobę **A**, aby podpisała jego klucz publiczny swoim. Następnie wysyła maila do osoby **B**. Osoba **B**, pobiera z serwera klucz osoby **C** i widzi, że klucz jest podpisany przez osobę **A**. A ufa **B**, więc może także zaufać sygnowanym przez niego przesyłkom. Tym sposobem jeśli **C** wyśle maila do dowolnej osoby, która ufa kluczowi osoby **A** może liczyć na zaufanie także przesyłkom podpisanych pośrednio przez niego.

Teraz **B** może również złożyć podpis cyfrowy na kluczu **C**, dzięki czemu osoby, które ufają kluczowi **B** będą także mogły zaufać kluczowi **C**.

Jak widać sieć zaufania jest zdecentralizowana i opiera się na relacjach między kluczami. Relacja może być jednokierunkowa (osoba A podpisuje klucz osoby B) lub dwukierunkowa (osoby A i B wzajemnie popisują swoje klucze). Każdy klucz zawiera tzw. sygnaturę. Sygnatura jest skrótem (np. MD5 lub SHA) klucza. Jeśli sygnatura pasuje do klucza weryfikacja wiadomości przebiega pozytywnie. Sygnatury tworzą ścieżki zaufania, dzięki którym możliwe staje się budowanie sieci zaufania. W poprzednim przykładzie B zaufał kluczowi C, ponieważ po ścieżce zaufania zbudowanej przez A, był pewny autentyczności klucza publicznego, który pobrał z serwera.

Idealnie jest, kiedy każde podpisanie kluczy przebiega w sposób dwukierunkowy, dzięki czemu sieć zaufania może znacznie się rozszerzać.

Aby w programie GPG podpisać klucz należy przejść w tryb edycji.

C:\F gpg This This unde	Program Files\GNU (GnuPG) 1.4.5; C program comes w is free softwar r certain condit	\GnuPG>gpgedit-key opyright (C) 2006 Fre ith ABSOLUTELY NO WAR e, and you are welcom ions. See the file CO	e Software Foundatio RANTY. le to redistribute it PYING for details.	n, Inc.	
pub	1024D/53BE5975	created: 2006-10-15	expires: never	usage:	SC
sub [un	2048g/8AC6C876 known] (1). 🖃	created: 2006-10-15	expires: never	usage :	Ε
Comm	and> _				

Przed podpisaniem klucza zaleca się sprawdzenie jego odcisku (fingerprint) i skontaktowanie się z właścicielem w celu jego weryfikacji.



Jeśli fingerprint jest prawidłowy można przystąpić do podpisania klucza publicznego danej osoby swoim kluczem prywatnym. W trakcie podpisywania zostaniemy poproszeni o wpisanie hasła klucza, którego używamy do podpisania.

Teraz można odesłać właścicielowi jego klucz wraz z naszym podpisem i poprosić o podpisanie naszego klucza.

10. Bibliografia

- •OpenPGP Message Format. <u>http://www.ietf.org/rfc/rfc2440.txt</u>
- •Dokumentacja GnuPG. http://www.gnupg.org/
- •http://pl.wikipedia.org/wiki/Kryptografia
- •SPKI Certificate Theory. http://www.ietf.org/rfc/rfc2693.txt
- •Lista dysuksyjna Enigmail. http://mozdev.org/mailman/listinfo/enigmail/
- •MIME Security with OpenPGP. <u>http://www.ietf.org/rfc/rfc3156.txt</u>
- •<u>http://www.openpgp.org/</u>
- •<u>http://www.philzimmermann.com</u>