Informacje

- **Kontakt:** katarzyna.mazur@umcs.pl
- **Konsultacje:** pokój 412 na 4 piętrze, przed konsultacjami proszę o wiadomość mailową
- **Zasady zaliczenia:** `https://kampus.umcs.pl/pluginfile.php/806844/mod_resource/content/1/zasady_zaliczenia_bsk_2022.pdf`
- **Materiały, aktualności, zmiany terminów zajęć:** `https://kampus.umcs.pl/course/view.php?id=20754`

Teoria

PGP (Pretty Good Privacy Całkiem Niezła Prywatność) jest kryptosystemem (tzn. systemem szyfrująco-deszyfrującym) autorstwa Phila Zimmermanna wykorzystującym ideę klucza publicznego. Podstawowym zastosowaniem PGP jest szyfrowanie poczty elektronicznej, transmitowanej przez kanały nie dające gwarancji poufności ani integralności poczty. Przez poufność rozumiemy tu niemożność podglądania zawartości listów przez osoby trzecie; przez integralność niemożność wprowadzania przez takie osoby modyfikacji do treści listu. PGP pozwala nie tylko szyfrować listy, aby uniemożliwić ich podglądanie, ale także sygnować (podpisywać) listy zaszyfrowane lub niezaszyfrowane w sposób umożliwiający adresatowi (adresatom) stwierdzenie, czy list pochodzi rzeczywiście od nadawcy, oraz czy jego treść nie była po podpisaniu modyfikowana przez osoby trzecie. Szczególnie istotny z punktu widzenia użytkownika poczty elektronicznej jest fakt, że techniki szyfrowania oparte o metodę klucza publicznego nie wymagają wcześniejszego przekazania klucza szyfrowania/deszyfrowania kanałem bezpiecznym (tzn. gwarantującym poufność). Dzięki temu, używając PGP, mogą ze sobą korespondować osoby, dla których poczta elektroniczna (kanał niepoufny) jest jedyną formą kontaktu. Weryfikacja kluczy opiera się o sieć zaufania (web of trust).

Za pomocą PGP można wygenerować parę kluczy (prywatny/publiczny), wyeksporotwać posiadane klucze do pliku, zaimportować klucz z pliku, uzyskać informacje o posiadaczu klucza (***key fingerprint***), podpisać klucz innej osoby. Odcisku tego użyć można do weryfikacji klucza innym kanałem, niekoniecznie bezpiecznym z punktu widzenia kryptografii, ale dającym nam pewność, że po drugiej stronie jest osoba, której się spodziewamy. Jeśli znamy właściciela klucza osobiście, może to być np. sprawdzenie ódcisku klucza"przez telefon. W innych przypadkach pozostaje albo osobiste spotkanie z właścicielem klucza, albo zaufanie do sygnatur już się w tym kluczu znajdujących. Kiedy już weszliśmy w posiadanie klucza publicznego innej osoby, aby móc z niego korzystać należy dołączyć go do swojego kółka z kluczami (***keyring***). Kółko z kluczami to nic innego, jak plik zawierający wiele kluczy w odpowiednim formacie zazwyczaj będzie to to samo kółko, na którym znajduje się klucz publiczny właściciela.

Sieć zaufania (ang. web of trust) zdecentralizowana metoda uwierzytelniania osób, w której nie ma hierarchicznej struktury organizacji uwierzytelniających, a zaufanie do poszczególnych certyfikatów jest sumą podpisów złożonych przez innych uczestników sieci. Każdy uczestnik sieci podpisuje klucze osób, które osobiście zweryfikował. Podpis stanowi poświadczenie, że osoba podpisująca jest przekonana o autentyczności klucza oraz tym, że faktycznie należy on do osoby, która deklaruje że jest jego właścicielem. Jeśli ktoś z uczestników sieci chce sprawdzić, czy pewien klucz rzeczywiście należy do danej osoby, próbuje ułożyć łańcuch zaufania w postaci: "Ja zweryfikowałem, że X to X i podpisałem jego

klucz KX, X zweryfikował, że Y to Y i podpisał jego klucz KY, itd., Z zweryfikował, że sprawdzany klucz rzeczywiście należy do tej osoby."

Zadania

Przykładowe maile, których można użyć w zadaniach: `student1.bsk@wp.pl`, `student2.bsk@wp.pl` *hasło:* `testowehaslo2021`.

4.1 Wykorzystując oprogramowanie `gpg`, wygeneruj zestaw kluczy (publiczny-prywatny) dla dowolnego użytkownika przy pomocy algorytmów RSA i DSA. (Pamiętaj, że algorytmu RSA możemy używać do szyfrowania/podpisywania, natomiast DSA do podpisywania.)

4.2 Wykorzystując oprogramowanie `gpg`, wygeneruj zestaw kluczy (publiczny-prywatny) dla **użytkownika BSK** (użyj przykładowych maili) przy pomocy algorytmu DSA. (Niech będzie to klucz bez określonego terminu ważności.)

4.3 Zmień datę wygaśnięcia klucza utworzonego w zad **4.2** na za 3 miesiące od dziś.

4.4 Wyświetl listę posiadanych kluczy, zarówno publicznych jak i prywatnych (czyli swój keyring, inaczej nazywany repozytorium kluczy programu `gpg`).

4.5 Usuń klucz użytkownika BSK (utworzony w zadaniu **4.2**) ze swojego repozytorium kluczy.

4.6 Wyeksportuj swój klucz publiczny do pliku w postaci tekstowej. Podejrzyj informacje o kluczu.

4.7 Wyeksportuj swój klucz publiczny do pliku w postaci binarnej. Podejrzyj informacje o kluczu.

4.8 Z serwera kluczy `hkp://keys.openpgp.org` zaimportuj klucz o ID `FD3FAC1D1237BEB6EB235994E448A9BB9F0A7A99`. Sprawdź ponownie swój keyring. Podpisz zaimportowany klucz. (*Poprzez podpisanie klucza innej osoby naszym kluczem zatwierdzamy jego autentyczność.*)

4.9 Wygeneruj losowy plik o wielkości 1MB korzystając z pliku `/dev/urandom`. Za pomocą `gpg` zaszyfruj plik przy pomocy szyfrowania symetrycznego i algorytmu AES-256. Wynik zapisz do pliku w postaci tekstowej. (*Oprogramowanie* `gpg` *umożliwia wykonanie szyfrowania symetrycznego, jak również asymetrycznego. W przypadku szyfrowania symetrycznego klucz szyfrujący/deszyfrujący jest generowany z podanego hasła.*)

4.10 Za pomocą `gpg` zaszyfrowano pewien plik przy pomocy algorytmu CAMELLIA128. Wynik szyfrowania zapisano do pliku `enc4.10.txt`. Hasło wykorzystane podczas szyfrowania to `test`. Odszyfruj plik `enc4.10.txt`, wynik deszyfrowania zapisz do pliku `dec4.10.txt`.

Oprogramowanie `gpg` umożliwia wykonanie szyfrowania symetrycznego, jak również asymetrycznego. W przypadku szyfrowania symetrycznego klucz szyfrujący/deszyfrujący jest generowany z podanego hasła.)

4.11 Wygeneruj 2 pary kluczy: dla użytkownika Alice (`alice.bsk@wp.pl`) i Bob (`bob.bsk@wp.pl`). Zakładając, że Bob chce wysłać zaszyfrowaną wiadomość do Alice, przygotuj plik `hello.txt` z przykładową wiadomością, zaszyfruj go (jako Bob), a następnie odszyfruj (jako Alice). Zaszyfrowany tekst powinien być zakodowany Base64.

Oprogramowanie `gpg` umożliwia wykonanie szyfrowania symetrycznego, jak również asymetrycznego. W przypadku szyfrowania asymetrycznego wykorzystywana jest para kluczy. Aby dokonać szyfrowania wiadomości, należy zaimportować lub posiadać klucz publiczny osoby, której chcemy przesłać zaszyfrowany plik. Wówczas jedynie ta osoba, przy pomocy swojego klucza prywatnego będzie w stanie odczytać wiadomość.

4.12 Ze strony kursu pobierz 2 pliki: `mallory.pub` (klucz publiczny użytkownika Mallory) oraz pliki `msg.sig` (podpisany przez niego plik) i `msg.txt`. Zweryfikuj podpis.

4.13 Użytkownicy systemów linuksowych mają możliwość pobrania oprogramowania z repozytoriów przygotowanych dla danej dystrybucji systemu. Niekiedy jednak dodatkowe oprogramowanie można pobrać jedynie ze strony WWW. W takim przypadku, jaka jest pewność, że plik znajdujący się na stronie, został na niej w rzeczywistości umieszczony przez dewelopera aplikacji, a nie hakera? Niektórzy programiści podpisują swoje oprogramowanie przy pomocy rozwiązań PGP (takich jak np. GPG). Dzięki temu, jako użytkownicy, mamy możliwość weryfikacji integralności oprogramowania. Proces weryfikacji jest prosty, należy:

(a) Pobrać klucz publiczny autora oprogramowania
(b) Sprawdzić fingerprint klucza
(c) Zaimportować klucz do własnego keyringa
(d) Pobrać sygnaturę dla ściąganego oprogramowania
(e) Użyć klucza publicznego do zweryfikowania podpisu

Pobierz oprogramowanie VeraCrypt i zweryfikuj jego integralność.

4.14 Pobierz najnowszą wersję serwera Apache i zweryfikuj integralność pobranego pliku. Potrzebny serwer kluczy to `pgpkeys.mit.edu`.

4.15 Podpisz klucz publiczny Apache dwoma kluczami z własnego keyringu.

Bardzo istotna jest pewność, że klucz publiczny, który właśnie w ten czy inny sposób pozyskaliśmy, należy naprawdę do osoby, do której wydaje się należeć. Zapewnieniu tego służą certyfikaty kluczy. Certyfikowanie (podpisywanie) czyjegoś klucza to nic innego, jak sygnowanie go swoim własnym - ma to na celu potwierdzenie jego autentyczności.

Jeśli użytkownik A otrzyma klucz publiczny użytkownika B bezpośrednio od niego, to zapewne jest to naprawdę klucz użytkownika B. Ale jeśli otrzymuje go od użytkownika C, któremu niekoniecznie ufa, i podejrzewa, że w rzeczywistości klucz ten może być sfałszowany przez C? Cóż, jeśli klucz ten jest certyfikowany przez pewnego innego użytkownika D (któremu A ufa, i którego klucz publiczny już ma), i sygnatura się zgadza, to A zyskuje pewność, że klucz przekazany przez C jest zgodny z oryginałem. Oczywiście A zakłada w tym momencie, że D w chwili certyfikowania tego klucza miał 100% pewności, że klucz ten należy do B - albo dlatego, że otrzymał go od B bezpośrednio, albo dlatego, że klucz był certyfikowany przez kolejną osobę, do której D miał zaufanie. Należy z tego wysnuć jeden wniosek: NIGDY nie certyfikuj cudzego klucza, jeśli nie masz pewności, że nie jest on fałszywy. W przeciwnym razie osoby, które następnie ten klucz od Ciebie otrzymają, będą sądzić, że taką pewność miałeś i używać (być może fałszywego) klucza z powodu zaufania do Ciebie.

Linki

- `https://home.agh.edu.pl/~szymon/artykuly/pgp_opis.html`
- `https://crypto.stackexchange.com/questions/2585/whycantdsabeusedforencryptions`
- `https://gnupg.org/gph/en/manual.pdf`
- `https://www.cs.stonybrook.edu/sites/default/files/PGP70IntroToCrypto.pdf`
- `https://nordlocker.com/blog/rsavsdsa/`
- `https://superuser.com/questions/655246/are-gnupg-1-and-gnupg-2-compatible-with-each-other`