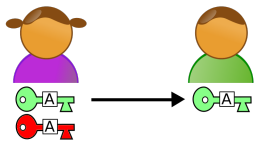
Informacje

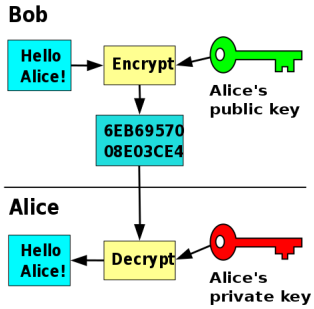
- **Kontakt:** katarzyna.mazur@umcs.pl
- **Konsultacje:** pokój 412 na 4 piętrze, przed konsultacjami proszę o wiadomość mailową
- **Zasady zaliczenia:**
- **Materiały, aktualności, zmiany terminów zajęć:** `https://kampus.umcs.pl/course/view.php?id=15080`

Teoria

Koncepcja ***kryptografii asymetrycznej*** lub ***kryptografii z kluczem publicznym*** pojawiła się w XX w. Za jej pomysłodawców uznaje się Whitfielda Diffiego i Martina Hellmana, którzy w 1976 roku zaprezentowali protokół uzgadniania klucza. Podstawową cechą odróżniającą kryptografię asymetryczną od symetrycznej jest wykorzystanie w komplementarnych operacjach (np. szyfrowaniu i deszyfrowaniu, podpisywaniu i weryfikowaniu) nie jednego, a dwóch kluczy – tradycyjnie nazywanych publicznym i prywatnym. Klucze te są od siebie zależne w sposób uwarunkowany przez specyfikę danego algorytmu. Niezbędne dla zachowania atrybutów bezpieczeństwa jest, by każdy użytkownik zachował swój klucz prywatny w tajemnicy. Algorytmy klucza publicznego projektuje się w taki sposób, by odtworzenie klucza prywatnego na podstawie znajomości klucza publicznego było zadaniem trudnym obliczeniowo.



Rysunek 1: Krok 1: Alice przesyła do Boba swój klucz publiczny.



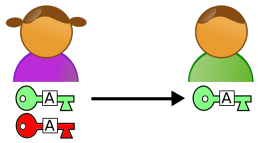
Rysunek 2: Kroki 2 i 3: Bob szyfruje wiadomość kluczem publicznym Alice, która to następnie otrzymuje zaszyfrowaną wiadomość i rozszyfrowuje ją kluczem prywatnym

W systemach kryptografii publicznej, każdy z użytkowników posiada dwa klucze - publiczny, udostępniany wszystkim, i prywatny, przechowywany pieczołowicie tylko przez właściciela. Na podstawie znajomości klucza publicznego, nie można odtworzyć klucz prywatnego, i na odwrót. Taki układ wyklucza niebezpieczeństwo przesyłania przez publiczne sieci komputerowe przesyłania jakichkolwiek danych, umożliwiających dostęp do listu osobom niepowołanym.

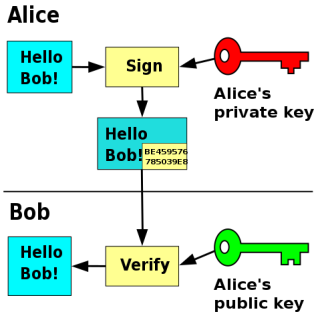
Klucz publiczny służy do szyfrowania wiadomości, która jednak może być rozszyfrowana tylko przy użyciu odpowiedniego klucza prywatnego. Klucz publiczny i prywatny danego użytkownika stanowią unikalną parę, tak że nawet osoba szyfrująca list czyimś kluczem publicznym nie może go przeczytać.

Podpis cyfrowy również związany jest kryptografią. Przez jakąś zaufaną instytucję, która potwierdza tożsamość konkretnego użytkownika generowana jest para kluczy (unikalnych ciągów bitów/cyfr): klucz prywatny - chroniony przez użytkownika (niedostępny dla nikogo innego, nawet dla instytucji generującej go) i klucz publiczny pasujący do klucza prywatnego. (Użytkownicy mogą również samiwygenerować parę kluczy, a następnie poprosić zaufaną instytucję o potwierdzenie tożsamości.) Na podstawie klucza publicznego "nie da się" odgadnąć klucza prywatnego - odgadniecie metodą prób i błędów powinno zajmować superkomputerowi setki lat.

Klucz prywatny może służyć do stworzenia podpisu dla danych cyfrowych w postaci "odcisku palca", tj. jawny algorytm korzysta z klucza prywatnego i danych tworząc dość krótki ciąg bitów, który potem daje się zweryfikować z pomocą klucza publicznego, że dane cyfrowe nie zostały zmienione od momentu "podpisania".



Rysunek 3: Krok 1: Alice przesyła do Boba swój klucz publiczny.



Rysunek 4: Kroki 2 i 3: Alice podpisuje wiadomość swoim kluczem prywatnym a następnie wysyła ją do Boba. Bob za pomocą klucza publicznego Alice weryfikuje, czy wiadomość rzeczywiście pochodzi od Alice.

Zadania

3.1 Wykorzystując bibliotekę openssl, wygeneruj 4096-bitowy klucz prywatny oraz przypisany mu klucz publiczny (w formacie `*.pem`) korzystając z algorytmu RSA. Oba klucze w czytelnej formie zapisz do pliku/ów.

3.2 Wygeneruj parę kluczy opartych na krzywej eliptycznej NIST P-256 prime256v1.

3.3 Zaszyfruj ciąg znaków `96c91ac9c4b92882c105db9846caa31b` otrzymanym kluczem publicznym RSA znajdującym się w pliku `pub.pem`. Zaszyfrowane dane zakoduj algorytmem Base64 i zapisz do pliku `myenc3.3.txt`. Prawidłowa odpowiedź znajduje się w pliku `enc3.3.txt`. Porównaj swoją odpowiedź z prawidłową odpowiedzią. Jakiego narzędzia możesz użyć do porównania?

3.4 Posiadając parę kluczy `priv3.4.pem` odszyfruj plik `enc3.4.txt`. Wynik zapisz do pliku `dec3.4.txt`.

3.5 Posiadając parę kluczy `priv3.5.pem`, zweryfikuj podpis znajdujący się w pliku `sig3.5.txt`.

3.6 Posiadając klucz publiczny `pub3.6.pem`, zweryfikuj podpisany skrót SHA-1 pliku `ex3.6.txt` zapisany jako `sig3.6.txt`.

3.7 Korzystając z narzędzi dostępnych w Kali Linux, utwórz plik o rozmiarze 1MB oraz wypełnij go losową zawartością (zakodowaną Base64). Wykorzystując bibliotekę `OpenSSL`, utwórz skrót (SHA-256) tak utworzonego pliku. Następnie podpisz cyfrowo ten skrót (SHA-256), po czym zweryfikuj ten podpis.

Linki

- `http://www.it-professional.pl/bezpieczenstwo/artykul,8748,zastosowania-polecen-openssl-tworzenie-centrum-autoryzacji.html`
- `https://www.openssl.org/docs/man1.0.2/man1/ecparam.html`
- `https://www.openssl.org/docs/man1.0.2/man1/rsautl.html`

Odpowiedzi

3.1 Generowanie pary kluczy:
```
openssl genrsa -out priv.pem 4096
```

Zapis klucza prywatnego do pliku:
```
openssl rsa -in priv.pem --text > privkey.txt
```

Generowanie klucza publicznego z klucza prywatnego:
```
openssl rsa -in priv.pem -pubout -out pub.pem
```

Zapis klucza publicznego do pliku:
```
openssl rsa -in pub.pem --pubin --text > pubkey.txt
```

3.2
```
openssl rsautl -encrypt -inkey pub.pem -pubin -in ex3.3.txt | base64 > enc3.3.txt
```

3.3 Sprawdzenie dostępnych krzywych eliptycznych:
```
openssl ecparam -list_curves
```

Generowanie pary kluczy:
```
openssl ecparam -name prime256v1 -genkey -out privec.pem
openssl ecparam -in privec.pem -text
```

Generowanie klucza publicznego z klucza prywatnego:
```
openssl ec -in privec.pem -out pubec.pem -pubout
```

3.4 Plik zaszyfrowano za pomocą polecenia (zaszyfrowano kluczem publicznym):
```
openssl rsa -in priv3.4.pem -pubout -out pub3.4.pem
openssl rsautl -encrypt -inkey pub3.4.pem -pubin -in ex3.4.txt -out enc3.4.txt
```

Odszyfrowanie (kluczem prywatnym):
```
openssl rsautl -decrypt -in enc3.4.txt -out dec3.4.txt -inkey priv3.4.pem
```

3.5
```
openssl genrsa -out priv3.5.pem 4096
```

Podpisanie pliku (z użyciem klucza prywatnego):
```
openssl rsautl -sign -in ex3.5.txt -inkey priv3.5.pem -out sig3.5.txt
```

Weryfikacja (z użyciem klucza publicznego):
```
openssl rsa -in priv3.5.pem -pubout -out pub3.5.pem
openssl rsautl -verify -in sig3.5.txt -inkey priv3.5.pem
```

3.6 Generowanie pary kluczy:
```
openssl genrsa -out priv3.6.pem 4096
```

Generowanie klucza publicznego z klucza prywatnego:
```
openssl rsa -in priv3.6.pem -pubout -out pub3.6.pem
```

```
openssl list -digest-commands
```

Podpisanie skrótu SHA-1 pliku kluczem prywatnym:
```
openssl dgst -sha1 -sign priv3.6.pem -out sig3.6.txt ex3.6.txt
```

Weryfikacja skrótu SHA-1 pliku kluczem publicznym:
```
openssl dgst -sha1 -verify pub3.6.pem -signature sig3.6.txt ex3.6.txt
```

3.7 Generowanie pliku 1MB, losowa zawaratość, kodowanie Base64:
```
head -c 1048576 < /dev/urandom | base64 > ex3.7.txt
```

Generowanie pary kluczy:
```
openssl genrsa -out priv3.7.pem 4096
```

Generowanie klucza publicznego z klucza prywatnego:
```
openssl rsa -in priv3.7.pem -pubout -out pub3.7.pem
```

Podpisanie skrótu SHA-256 pliku kluczem prywatnym:
```
openssl dgst -sha256 -sign priv3.7.pem -out sig3.7.txt ex3.7.txt
```

Weryfikacja skrótu SHA-256 pliku kluczem publicznym:
```
openssl dgst -sha256 -verify pub3.7.pem -signature sig3.7.txt ex3.7.txt
```