**Wrocław 01.10.2024**

**cyberbezpieczństwo**

**Laboratorium nr7**

**Zastosowania Kryptografii**

# Wstęp.

Zastosowanie metod kryptograficznych jest bardzo szerokie. Laboratorium koncentruje się na aspektach związanych z podpisami cyfrowymi i infrastrukturą klucza publicznego (PKI). Podpisy cyfrowe są powszechnie używane na co dzień, dlatego w kontekście bezpieczeństwa ważne jest, aby zapoznać się z ich działaniem i właściwościami. Infrastruktura klucza publicznego (PKI) składa się z protokołów, standardów i usług, które ustanawiają i obsługują aplikacje takiego systemu zaufania.

Podpis cyfrowy jest używany głównie do celów uwierzytelniania. Służy do przekonywania stron komunikujących się o swojej tożsamości i wymiany kluczy sesji. Jest to elektroniczny format podpisu, którego może użyć osoba do uwierzytelnienia tożsamości nadawcy wiadomości lub osoby podpisującej dokument. Zapewnia, że ​​oryginalna treść wysłanej wiadomości lub dokumentu pozostaje nienaruszona. Innymi słowy, jest to narzędzie kryptograficzne do podpisywania wiadomości, danych i weryfikacji autentyczności wiadomości, danych. Podpisy cyfrowe wykorzystują infrastrukturę klucza publicznego (PKI), aby zapewnić najwyższy poziom bezpieczeństwa i powszechną akceptację. Ponadto podpisy cyfrowe są dziś szeroko stosowane w biznesie, np. do autoryzacji płatności bankowych, do wymiany podpisanych dokumentów elektronicznych, do podpisywania transakcji w publicznych systemach blockchain, do podpisywania umów cyfrowych iw wielu innych sytuacjach.

Podpis cyfrowy zapewnia:

* uwierzytelnianie - pozwala sprawdzić, czy nadawca utworzył i podpisał wiadomość,
* integralność - pozwala sprawdzić, czy dane nie zostały zmienione po podpisaniu,
* niezaprzeczalność - osoba podpisująca wiadomość nie może odmówić jego podpisu,

Na proces podpisu cyfrowego składają się trzy algorytmy:

* generowanie kluczy - generuje zestaw kluczy, prywatnych i publicznych,
* podpisywanie - algorytm podpisuje wiadomość kluczem prywatnym, zwykle skrót wiadomości jest szyfrowany.
* weryfikacja - algorytm sprawdza autentyczność wiadomości weryfikując podpis kluczem publicznym,

Do podpisywania wiadomości/danych stosowane są asymetryczne metody kryptograficzne. Proces podpisu cyfrowego zapobiega kilku typowym atakom, takim jak:

* tylko klucz - atakujący ma dostęp do klucza publicznego
* znana wiadomość - atakujący ma dostęp do ważnych podpisów znanych wiadomości, ale nie do tych, które wybrał
* adaptacyjna wybrana wiadomość - atakujący uzyskuje dostęp do podpisów różnych wiadomości, które wybrał.

Najpopularniejsze kryptosystemy klucza publicznego to:

- RSA,

- DSA (Digital Signature Algorithm),

- ECDSA / EdDSA na podstawie krzywych eliptycznych (ECC),

A diagram of a network

Description automatically generated

Rysunek 1 Przepływ podpisów cyfrowych - źródło <https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq>

Infrastruktura klucza publicznego (PKI) umożliwia wydajną i bezpieczną identyfikację kluczy publicznych. Może być używany wewnątrz lub między organizacjami za pomocą Internetu. Różne typy PKI można wdrażać, zmieniając podstawowe szczegóły konfiguracji, reguły zaufania. PKI to zestaw tożsamości, ról, zasad i działań służących do tworzenia, używania, zarządzania, dystrybucji i odwoływania kluczy publicznych i prywatnych. Istnieje kilka standardów PKI w branży, najpopularniejszym jest X.509 w wersji 3. Ponadto PKI zapewnia:

* uwierzytelnianie,
* poufność,
* niezaprzeczalność,
* integralność,

Model zarządzania PKI polega na określeniu reguł formatów wiadomości i procedur używanych do komunikacji. Model wyjaśnia również, w jaki sposób różne podmioty komunikują się ze sobą. Główne podmioty zarządzające PKI to:

**Jednostka końcowa (EE):** może to być użytkownik lub aplikacja, do której jest obsługiwany certyfikat. Potrzebuje bezpiecznego dostępu przynajmniej do swojej nazwy i klucza prywatnego.

**Urząd certyfikacji (CA):** może to być strona trzecia lub organizacja EE, która wydaje certyfikat EE.

**Urząd rejestracyjny (RA):** jest to podzbiór EE i jest składnikiem opcjonalnym. Jeśli RA nie ma, to CA wykonuje funkcje RA. RA realizuje takie funkcje, jak generowanie kluczy, zarządzanie parami kluczy, dystrybucja tokenów itp.

**Wystawca list odwołania certyfikatów (CRL):** jeśli niektóre certyfikaty muszą zostać unieważnione, zajmie się tym wystawca list CRL. Jest to również składnik opcjonalny.

**Repozytorium certyfikatów:** jednostka pamięci służąca do definiowania sposobu przechowywania certyfikatów i list CRL oraz dostępu do nich przez EE.

Żądanie podpisania certyfikatu (CSR) powinno zostać wygenerowane na komputerze klienta. CSR zawiera klucz publiczny i identyfikator do certyfikacji. CSR jest następnie wysyłany do RA / CA w celu podpisania certyfikatu. Następnie certyfikat nabiera pełnej wartości. Generowanie CSR na komputerze klienta jest ważne, ponieważ tylko wtedy klucz prywatny jest znany właścicielowi, a inne podmioty nie mają do niego dostępu.

Diagram of a building activity

Description automatically generated

Rysunek 2 <https://cybersecuritynews.co.uk/how-does-public-key-infrastructure-pki-work>

# Zastosowanie kryptografii

# OpenSSL

Wygeneruj przy użyciu OpenSSL klucz RSA (2048 bits):

openssl genrsa -out private\_key.pem 2048

Wygeneruj odpowiedni klucz publiczny:

openssl rsa -in private\_key.pem -outform PEM -pubout -out public\_key.pem

Podpisz plik:

openssl dgst -sha256 -sign private\_key.pem -out encrypted\_hash.sha256 data.txt

Zweryfikuj podpis:

openssl dgst -sha256 -verify public\_key.pem -signature encrypted\_hash.sha25 data.txt

Dokumentacja OpenSSL: <https://www.openssl.org/docs/>

# OpenPGP: (https://pl.wikibooks.org/wiki/GnuPG/)

Wygeneruj parę kluczy korzystając z narzędzia GPG:

gpg --expert --full-gen-key

Przed wygenerowaniem kluczy należy uzupełnić następujące pola: odpowiedni algorytm, właściwości algorytmu (np. długość klucza, krzywa eliptyczna itp.), datę ważności, nazwę, adres e-mail i hasło.

Aby wyświetlić dostępne klucze, użyj:

- publiczne:

gpg --list-keys,

* prywatne:

gpg -–list-secret-keys,

Eksportuj klucz publiczny:

* gpg --export --armour --output *<file name>* *<email\_address>*   
  - text form - ASCII,
* gpg --export --output *<file name>* *<email\_address>*   
  - binary form,

Eksportuj klucz prywatny:

* gpg --export-secret-keys -a *<file name> <email\_address*>*,*

Importuj klucz:

gpg –-import *<file name>*

Edytuj klucz:

gpg –-edit-key *<email-address>*

Podpisz plik:

* gpg --sign *<file name>,*

Podpisz plik wybranym kluczem:

* gpg –-sign -u *<key\_id> <file name>*

Zweryfikuj podpis:

* gpg -–verify *<file name>.*gpg

Podpisz zaimportowany klucz publiczny własnym kluczem:

* gpg –sign-key <email\_address>,

Zaszyfruj plik przy użyciu gpg:

* gpg --encrypt –-recipient <email\_address> --output <output.file> <input.file>

Odszyfruj plik przy użyciu gpg:

* gpg --decrypt --output <output.file> <input.file>

Inne użyteczne operacje gpg:

* gpg –-fingerprint
* gpg –edit-key

## **Zadania**:

### Wygeneruj zestaw kluczy za pomocą OpenSSL (RSA) i OpenPGP (ECC i RSA).

### Eksport klucza publicznego z GPG (format ACSII).

### Przenieś swój klucz publiczny (OpenSSL i OpenPGP) na inny komputer / konto użytkownika.

### W przypadku OpenPGP klucz publiczny należy zaimportować i podpisać lokalnym kluczem prywatnym. Sprawdź odcisk palca certyfikatu na obu kontach komputera /użytkownika.

### Utwórz plik tekstowy i podpisz go za pomocą obu metod.

### Przenieś plik i podpisy na inny komputer / konto użytkownika i zweryfikuj podpisy.

### Po pomyślnym zweryfikowaniu podpisu należy zmodyfikować treść oryginalnego dokumentu i ponownie zweryfikować podpis cyfrowy.

### Tylko dla OpenPGP. Zaszyfruj plik za pomocą klucza publicznego, a następnie odszyfruj zaszyfrowany plik. Następnie zmień jeden znak w zaszyfrowanym pliku i spróbuj ponownie go odszyfrować.

**Wnioski**/ **Pytania**:

### Czy można wygenerować tylko jeden z pary kluczy dla algorytmów asymetrycznych (np. klucz prywatny)? Czy to miałoby sens?

### Jaką formę ma klucz GPG? Czym różni się klucz PEM od klucza OpenPGP?

### Czy wymiana kluczy prywatnych jest uzasadniona?

### Jaki rezultat zostałosiągniety w zadanie 1.7 i dlaczego ?

### Co to jest odcisk palca?

### Czy udało ci się zmienić jeden znak w zaszyfrowanym pliku i odszyfrować zaszyfrowany tekst? Odpowiedź uzasadnij.

### Jaka jest różnica w sygnaturze (OpenPGP) dla algorytmu ECC i RSA?

### Czy wiadomość może być podpisana kluczem publicznym? Jeśli tak, jakie mogą być konsekwencje.

# PKI - X.509 standard.

OpenVPN wykorzystuje certyfikaty do szyfrowania ruchu między serwerem a klientami. Centrum certyfikacji (CA) jest zbudowane w celu wydawania zaufanych certyfikatów. Easy-RSA to narzędzie służące do tworzenia urzędów certyfikacji PKI i zarządzania nimi. Easy-RSA jest oparty na bibliotece OpenSSL.

## Utwórz CA.

W katalogu domowym (/home/server) znajduje się folder z narzędziem do budowania i zarządzania certyfikatami do OpenVPN, folder znajduje się na maszynie Ubuntu\_server.

**Generowanie zestawu certyfikatów:**

przejdź do katalogu EasyRSA:

cd ~/EasyRSA-3.0.10/

skopiuj przykład:

cp vars.example vars

edytuj zmienne:

nano vars

Usuń # i nadaj odpowiednie wartości zmiennym:

#set\_var EASYRSA\_REQ\_COUNTRY "PL"

#set\_var EASYRSA\_REQ\_PROVINCE "Wroclaw"

#set\_var EASYRSA\_REQ\_CITY "Wroclaw"

#set\_var EASYRSA\_REQ\_ORG "PWR"

#set\_var EASYRSA\_REQ\_EMAIL "<student e-mail>"

#set\_var EASYRSA\_REQ\_OU "PWR"

Uruchom skrypt z parametrem init-pki aby zainicjalizować CA:

./easyrsa init-pki

Aby zbudować CA, uruchom następujące polecenie, generuje ca.key (private.key) i ca.crt (certyfikat publiczny). Bardzo ważne jest, aby przechowywać klucz w bezpiecznym miejscu, ponieważ od tego zależy bezpieczeństwo sieci VPN.

./easyrsa build-ca nopass

Zwróć uwagę, że nazwa zwyczajowa powinna być poprawnie ustawiona zgodnie z jej przeznaczeniem.

Aby unieważnić certyfikat klienta, musisz unieważnić klienta w CA:

./easyrsa revoke *<client’s common name>*

Następnie wygeneruj plik CRL i prześlij go na serwer:

./easyrsa gen-crl

## Wygeneruj certyfikaty serwera i klienta.

Uruchom polecenie, aby wygenerować klucz prywatny serwera (server.key) i żądanie certyfikatu CSR (server.req):

./easyrsa gen-req server nopass

Podpisz żądanie serwera za pomocą CA, generuje server.crt:

./easyrsa sign-req server server

Uruchom polecenie, aby wygenerować klucz prywatny klienta (client1.key) i żądanie certyfikatu (client1.req):

./easyrsa gen-req client1 nopass

Podpisz żądanie klienta za pomocą CA, wygeneruje client1.crt:

./easyrsa sign-req client client1

Wygeneruj klucz Diffiego-Hellmana (dh.pem), który umożliwia wymianę kluczy:

./easyrsa gen-dh

Wygeneruj podpis HMAC (ta.key), aby wzmocnić możliwości weryfikacji integralności TLS serwera:

openvpn --genkey --secret ta.key

Serwer składa się z plików:

* server.key – klucz prywatny serwera,
* server.crt – certyfikat serwera,
* ta.key HMAC podpis,
* dh.pem - Diffie-Hellman klucz,
* ca.crt - CA certyfikat,

Klient:

* client.key – klucz prywatny,
* client.crt – certyfikat,
* ta.key - HMAC podpis,
* ca.crt – certyfikat CA,

Sprawdź właściwości jednego z certyfikatów i odpowiedz na następujące pytania:

### Jaki jest używany algorytm podpisu?

### Jaki jest używany algorytm klucza publicznego?

### Jaka wersja standardu X.509 została użyta do wygenerowania certyfikatu?

### Co jeszcze można odczytać z certyfikatu serwera / klienta?