**SPRAWOZDANIE**

**BEZPIECZEŃSTWO SYSTEMÓW INFORMATYCZNYCH**

|  |  |  |
| --- | --- | --- |
| **Osoba wykonująca** | **Grupa** | **Data** |
| Michał J. Sidor | 5.5/9 | 10.12.2018r. |
| **Uczelnia** | **Wydział** | **Kierunek** |
| Politechnika Lubelska | Elektrotechniki i Informatyki | Informatyka I. stopnia, stacjonarne |
| **Temat** | | |
| **LABORATORIUM NR 5.**  Kryptografia klucza publicznego oraz PKI | | |
|

**5.3 Zadanie 3: Zastosowanie PKI do stron WWW**

P1. Ponownie wpisz w przeglądarce adres: https://PKILabServer.com:4433

Opisz i wyjaśnij swoje obserwacje. Co się stało i dlaczego?

Zobaczyliśmy informacje o certyfikatach - zawartość adresu została tym razem wyświetlona, ponieważ dodaliśmy do przeglądarki certyfikat go autoryzujący (ca.crt), przez co przeglądarka mogła go zidentyfikować jako “zaufany”. Nie zadziałały więc zabezpieczenia przeglądarki zapobiegające wyświetlaniu zawartości stron, które nie mają odpowiednich certyfikatów bezpieczeństwa.

P2. Wygeneruj i certyfikuj certyfikat dla serwera testowy.cs.pollub.pl. W sprawozdaniu zamieść użyte polecenia oraz zrzut ekranu prezentujący odpowiedź uruchomionego serwera w przeglądarce.

[12/17/2018 18:59] student@ubuntu:~/Desktop/demoCA$ **openssl genrsa -aes128 -out testowy.key 1024**

[12/17/2018 19:07] student@ubuntu:~/Desktop/demoCA$ **openssl req -new -key testowy.key -out testowy.csr -config openssl.cnf**

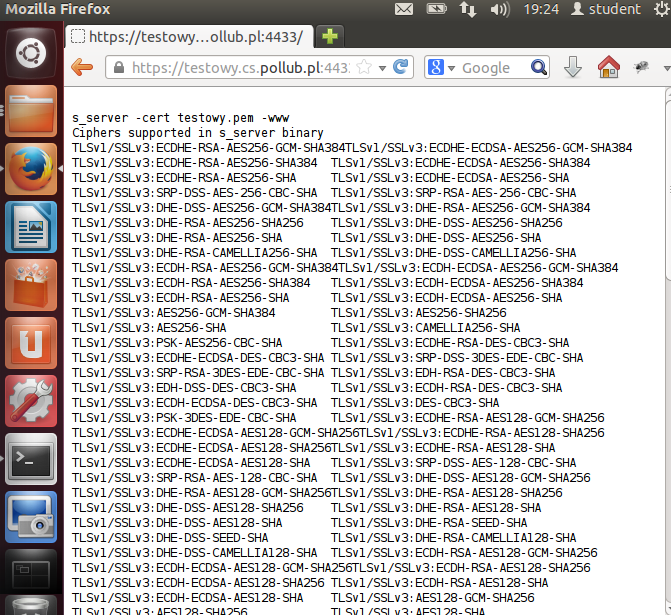
[12/17/2018 19:09] student@ubuntu:~/Desktop/demoCA$ **openssl ca -in testowy.csr -out testowy.crt -cert ca.crt -keyfile ca.key -config openssl.cnf**

[12/17/2018 19:14] student@ubuntu:~/Desktop/demoCA$ **sudo gedit /etc/hosts** - dodanie domeny testowy.cs.pollub.pl do pliku hosts

[12/17/2018 19:17] student@ubuntu:~/Desktop/demoCA$ **cp testowy.key testowy.pem**

[12/17/2018 19:17] student@ubuntu:~/Desktop/demoCA$ **cat testowy.crt >> testowy.pem**

[12/17/2018 19:20] student@ubuntu:~/Desktop/demoCA$ **openssl s\_server -cert testowy.pem -www**



P3. Zmodyfikuj pojedynczy bit w pliku server.pem. Uruchom ponownie serwer i przeładuj zawartość udostępnianej przez niego strony. Określ z jakich elementów składa się plik certyfikatu (\*.pem) i jak wpływa modyfikacja bitów w poszczególnych elementach tego˙ z pliku na działanie serwera. Wyniki zamieść w tabeli zawierającej trzy kolumny: element pliku pem, opis elementu, opis wpływu modyfikacji bitu w tym elemencie na działanie serwera.

|  |  |  |
| --- | --- | --- |
| **element pliku pem** | **opis elementu** | **opis wpływu modyfikacji bitu w tym elemencie na działanie serwera** |
| **pierwsza część pliku -**  -----BEGIN RSA PRIVATE KEY-----  …  -----END RSA PRIVATE KEY----- | element zawiera klucz prywatny | nie można odczytać klucza prywatnego, nie można uruchomić serwera |
| **środkowa część pliku-**  Certificate: | element zawiera główny certyfikat wraz z danymi | zmieniają się dane certyfikatu, serwer działa |
| **ostatnia część pliku -**  -----BEGIN CERTIFICATE-----  …  -----END CERTIFICATE----- | element zawiera certyfikat pomocniczy (ca.crt) | nie można odczytać certyfikatu,  nie można uruchomić serwera |

P4. Przywróć poprawną postać certyfikatu serwera, uruchom ponownie serwer a następnie wyświetl stronę WWW korzystając z adresu https://localhost:4433. Jaki jest efekt? Wyjaśnij co się stało.

Wyświetla się taka sama zawartość jak przy “przejściu” na adres pkilabserver.com. Dzieje się tak dlatego, że adres, na który dodaliśmy przekierowanie w pliku hosts z domeny pkilabserver.com - 127.0.0.1 - to po prostu adres komputera lokalnego, czyli tzw. localhost (wpisanie adresu “localhost” jest automatycznie kojarzone z adresem IP 127.0.0.1).

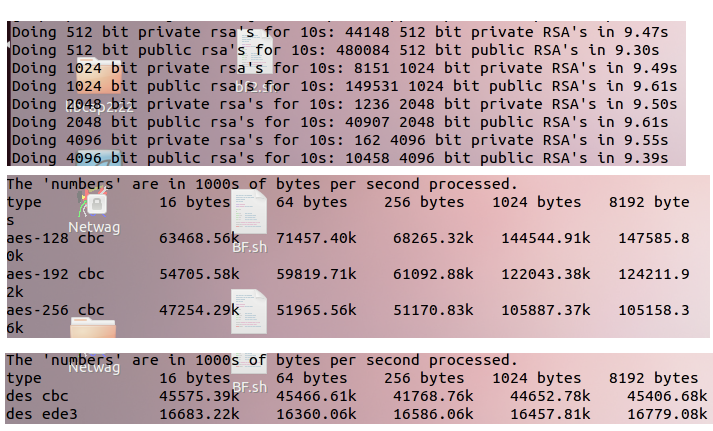
Serwer jest hostowany przez nasz komputer, który jest równocześnie klientem, stąd - wymiana danych odbywa się w zakresie jednego urządzenia, a więc - w zakresie lokalnym. To dlatego używamy ręcznych ustawień usługi DNS (kojarzenie nazw domen z adresami IP) w taki sposób, aby “odwiedzenie” domeny pkilabserver.com przekierowało nas na hostowany przez nas serwer poprzez użycie adresu 127.0.0.1 - adresu localhost.

**5.4 Zadanie 4: Porównanie wydajności: RSA versus AES**

P5. Opracuj zestawienie uzyskanych wyników. W celu zminimalizowania błędów podaj wyniki średnie dla każdej operacji. Przedstaw je w postaci umożliwiającej porównanie wydajności np. obliczając ilość danych przetwarzanych w ciągu jednej sekundy. Jeśli któraś z operacji trwa zbyt krótko możesz zmierzyć czas jej wielokrotnego wykonania.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Średnie czasy (user+sys)** | **DES-3** | | | **RSA** | | | **AES-128** | | |
| **1024b** | **2048b** | **4096b** | **1024b** | **2048b** | **4096b** | **1024b** | **2048b** | **4096b** |
| **Szyfrowanie** | | | | | | | | | |
| **16B** | 0,0028 | 0,0032 | 0,0034 | 0,0028 | 0,0036 | 0,0040 | 0,0020 | 0,0024 | 0,0036 |
| **100kB** | 0,0034 | 0,0036 | 0,0040 | 0,0044 | 0,0046 | 0,0048 | 0,0028 | 0,0040 | 0,0044 |
| **5MB** | 0,0140 | 0,0260 | 0,0272 | 0,0176 | 0,0182 | 0,0204 | 0,0168 | 0,0176 | 0,0188 |
| **Średni czas/1MB** | **61,2** | **70,1** | **74,3** | **61,8** | **78,9** | **87,4** | **44,2** | **52,4** | **78,7** |
|  | **67,2** | | | **76** | | | **58,4** | | |
| **Deszyfrowanie** | | | | | | | | | |
| **16B** | 0,0024 | 0,0040 | 0,0042 | 0,0028 | 0,0040 | 0,0044 | 0,0020 | 0,0028 | 0,0028 |
| **100kB** | 0,0028 | 0,0048 | 0,0052 | 0,0032 | 0,0032 | 0,0048 | 0,0024 | 0,0028 | 0,0044 |
| **5MB** | 0,0188 | 0,0192 | 0,0196 | 0,0220 | 0,0196 | 0,0200 | 0,0176 | 0,0172 | 0,0180 |
| **Średni czas/1MB** | **52,4** | **87,4** | **91,8** | **61,2** | **87,4** | **96,1** | **43,7** | **61,2** | **71,2** |
|  | **77,2** | | | **81,6** | | | **58,7** | | |

P6. Użyj polecenia speed z OpenSSL’a do wykonania testu prędkości algorytmów

****

Porównajmy ze sobą ilość danych przetworzonych w ciągu jednej sekundy w przypadku 1024 public RSA, AES-128 CBC i DES CBC. W przypadku RSA musimy uwzględnić, że wartości podane są w bitach (nie bajtach), stąd - należy podzielić je przez 8.

**RSA: 19 449 935 bajtów/s.**

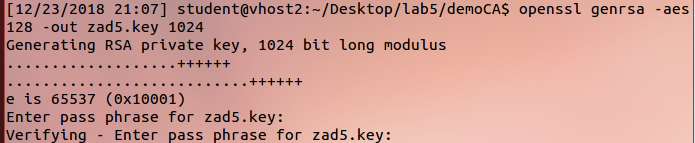
**AES-128: 144 544 910 bajtów/s.**

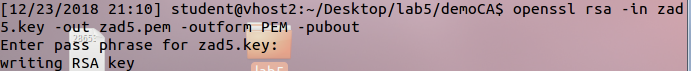
**DES: 44 652 780 bajtów/s.**

**Zarówno wyniki zmierzone poleceniem time, jak i te zmierzone poleceniem openssl speed pokazują, że najszybszym i najwydajniejszym algorytmem spośród powyższych jest algorytm AES-128, drugim - DES, a najmniej wydajnym z nich - RSA. “Wolniejsze” działanie RSA możemy wyjaśnić tym, że - w przeciwieństwie do (symetrycznych) algorytmów AES i DES - jest to algorytm asymetryczny.**

**5.5 Zadanie 5: Utwórz podpis cyfrowy**

Tworzenie pary kluczy:

****

****

1. Podpisz skrót SHA256 pliku example.txt; zapisz podpis w pliku example.sha256.

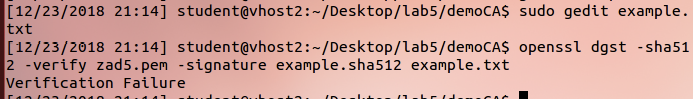


2. Zweryfikuj podpis umieszczony w pliku example.sha256.



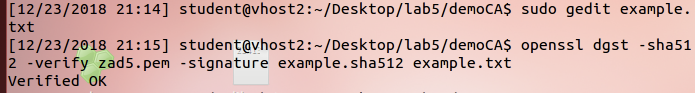
Zweryfikowano pomyślnie.

3. Zmodyfikuj zawartość pliku example.txt, a następnie ponownie zweryfikuj podpis cyfrowy.



Po zmianie zawartości pliku - weryfikacja nie powiodła się.

4. Przywróć oryginalną zawartość pliku example.txt, a następnie ponownie zweryfikuj podpis cyfrowy.



Po przywróceniu oryginalnej zawartości - weryfikacja zakończyła się sukcesem.

Z otrzymanych danych możemy wnioskować, że weryfikacja podpisów cyfrowych pozwala na sprawdzenie, czy otrzymana wiadomość (plik) jest identyczna do pierwotnej, dla której podpis został wygenerowany, co czyni podpisy cyfrowe użytecznym narzędziem w zapewnianiu bezpieczeństwa w danym systemie.