Biblioteka NaCl

mgr inż. Michał Trojnara

Instytut Telekomunikacji
Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Praca realizowana pod kierunkiem prof. Zbigniewa Kotulskiego

25 kwietnia 2012

Plan prezentacji

Plan prezentacji

Plan prezentacji

Plan prezentacji

Autorzy

- Daniel J. Bernstein (University of Illinois at Chicago)
- Tanja Lange (Technische Universiteit Eindhoven)
- Peter Schwabe (Academia Sinica)

`http://nacl.cr.yp.to/`

Główne cele

- Bezpieczeństwo
  - Kryptograficzne
  - Użytkowe
- Wysoka wydajność
- Licencja typu Public Domain

Główne cele

- Bezpieczeństwo
  - Kryptograficzne
  - Użytkowe
- Wysoka wydajność
- Licencja typu Public Domain

Główne cele

- Bezpieczeństwo
  - Kryptograficzne
  - Użytkowe
- Wysoka wydajność
- Licencja typu Public Domain

Odporność na znane ataki

- Brak skoków zależnych od tajnych danych
- Brak indeksów zależnych od tajnych danych
- Odporność na ataki z wyrocznią dopełnień[1]
  - Najpierw szyfowanie, potem podpis
  - Czas weryfikacji niezależny od dopełnienia
  - Odrzucanie wiadomości z tym samym nonce
- Używanie losowości dostarczanej przez jądro

[1] Padding Oracle

Odporność na znane ataki

- Brak skoków zależnych od tajnych danych
- Brak indeksów zależnych od tajnych danych
- Odporność na ataki z wyrocznią dopełnień[1]
  - Najpierw szyfowanie, potem podpis
  - Czas weryfikacji niezależny od dopełnienia
  - Odrzucanie wiadomości z tym samym nonce
- Używanie losowości dostarczanej przez jądro

[1] Padding Oracle

Odporność na znane ataki

- Brak skoków zależnych od tajnych danych
- Brak indeksów zależnych od tajnych danych
- Odporność na ataki z wyrocznią dopełnień[1]
  - Najpierw szyfowanie, potem podpis
  - Czas weryfikacji niezależny od dopełnienia
  - Odrzucanie wiadomości z tym samym nonce
- Używanie losowości dostarczanej przez jądro

[1] Padding Oracle

Odporność na znane ataki

- Brak skoków zależnych od tajnych danych
- Brak indeksów zależnych od tajnych danych
- Odporność na ataki z wyrocznią dopełnień[1]
  - Najpierw szyfowanie, potem podpis
  - Czas weryfikacji niezależny od dopełnienia
  - Odrzucanie wiadomości z tym samym nonce
- Używanie losowości dostarczanej przez jądro

[1] Padding Oracle

Skoki zależne od danych (1)

```
// return true when inputs are identical
bool compare1(const char *a,
              const char *b,
              const size_t len) {
    for(size_t i=0; i<len; ++i)
        if(a[i]!=b[i])
            return false;
    return true;
}
```

Skoki zależne od danych (2)

```
// return true when inputs are identical
bool compare2(const char *a,
              const char *b,
              const size_t len) {
    char x='\0';
    for(size_t i=0; i<len; ++i)
        x|=a[i]^b[i];
    return x=='\0';
}
```

Odporność na błędy użycia

- Ograniczona parametryzacja przez końcowego użytkownika
- Współbieżność bez potrzeby użycia synchronizacji
- Brak dynamicznego zarządzania pamięcią
- Dostępność w różnych[2] językach programowania

[2] C, C++, Python

Odporność na błędy użycia

- Ograniczona parametryzacja przez końcowego użytkownika
- Współbieżność bez potrzeby użycia synchronizacji
- Brak dynamicznego zarządzania pamięcią
- Dostępność w różnych[2] językach programowania

[2] C, C++, Python

Odporność na błędy użycia

- Ograniczona parametryzacja przez końcowego użytkownika
- Współbieżność bez potrzeby użycia synchronizacji
- Brak dynamicznego zarządzania pamięcią
- Dostępność w różnych[2] językach programowania

[2] C, C++, Python

Odporność na błędy użycia

- Ograniczona parametryzacja przez końcowego użytkownika
- Współbieżność bez potrzeby użycia synchronizacji
- Brak dynamicznego zarządzania pamięcią
- Dostępność w różnych[2] językach programowania

[2] C, C++, Python

Wydajność – użyte mechanizmy

- Curve25519 zamiast RSA
- Salsa20 zamiast AES
- Poly1305 zamiast HMAC
- EdDSA zamiast ECDSA

Wydajność – użyte mechanizmy

- Curve25519 zamiast RSA
- Salsa20 zamiast AES
- Poly1305 zamiast HMAC
- EdDSA zamiast ECDSA

Wydajność – użyte mechanizmy

- Curve25519 zamiast RSA
- Salsa20 zamiast AES
- Poly1305 zamiast HMAC
- EdDSA zamiast ECDSA

Wydajność – użyte mechanizmy

- Curve25519 zamiast RSA
- Salsa20 zamiast AES
- Poly1305 zamiast HMAC
- EdDSA zamiast ECDSA

Wydajność – testy

Platforma testowa: AMD Phenom II X6 1100T CPU

- ponad 80000 operacji crypto_box na sekundę
- ponad 80000 operacji crypto_box_open na sekundę
- ponad 70000 operacji crypto_sign_open na sekundę
- ponad 180000 operacji crypto_sign na sekundę

Generowanie pary kluczy

```
#include "crypto_box.h"

std::string pk;
std::string sk;

pk = crypto_box_keypair(&sk);
```

Uwierzytelnione szyfrowanie

```
#include "crypto_box.h"

std::string pk;
std::string sk;
std::string n;
std::string m;
std::string c;

c = crypto_box(m,n,pk,sk);
```

Uwierzytelnione deszyfrowanie

```
#include "crypto_box.h"

std::string pk;
std::string sk;
std::string n;
std::string c;
std::string m;

m = crypto_box_open(c,n,pk,sk);
```

Wnioski

- Użytkownikcy bibliotek kryptograficznych (zwykle) nie są kryptologami
- Nowe algorytmy pozwalają na zapewnienie wysokiej wydajności bez obniżenia poziomu bezpieczeństwa

Wnioski

- Użytkownikcy bibliotek kryptograficznych (zwykle) nie są kryptologami
- Nowe algorytmy pozwalają na zapewnienie wysokiej wydajności bez obniżenia poziomu bezpieczeństwa

Pytania

Dziękuję za uwagę