Praktyczne aspekty wykorzystania nowoczesnej kryptografii

Wojciech A. Koszek

IX Liceum Ogólnokształcące im. C. K. Norwida w Częstochowie
dunstan@freebsd.czest.pl

Streszczenie

Kryptologia jest nauką rozwijającą się od najdawniejszych czasów. Od początków cywilizacji ludzie czuli potrzebę przekazywania danych w poufny sposób. Najpopularniejszym szyfrem starożytności jest najprawdopodobniej szyfr Cezara i choć podobnie jak w każdej dziedzinie historii nie można powiedzieć, czy pewien fakt nie miał miejsca wcześniej, to właśnie szyfr Cezara wspominany jest przez większość publikacji dotyczących kryptografii jako jeden z pierwszych. Działanie tego szyfru jest niezwykle proste i polega na zamianie litery z tekstu jawnego (*ang. cleartext*), na literę położoną o 3 miejsca dalej w alfabecie, tworząc w wyniku kryptogram (*ang. ciphertext*). Oczywiście obecnie nie można tego nazwać zabezpieczeniem. Najsłynniejszym przykładem stosowania kryptografii może być maszyna szyfrująca Enigma [1].

1 Wstęp

Kryptografia od zawsze odgrywała w komunikacji kluczowe znaczenie. Dziś z koniecznością utajnienia danych spotykają się wszyscy: przeciętni, często nieświadomi użytkownicy domowych komputerów, programiści, czy też administratorzy systemów. W ogólnie pojętym bezpieczeństwie, efekty pracy przeciwnika mobilizują bądź też zmuszają do wymyślania coraz to nowszych technik obrony lub ataku. Tak jest również z kryptologią, która łączy w sobie kryptografię (zajmującą się tworzeniem i budową wszelkich metod szyfrowania) i kryptoanalizę (zajmującą się badaniem metod szyfrowania w celu „złamania szyfru"). Rozwój kryptografii pociągnął za sobą rozwój kryptoanalizy (od pierwszych analiz statystycznych zaszyfrowanych tekstów po wykorzystywanie ogromnych mocy obliczeniowych [2]. Kluczem obecnej kryptografii są złożone obliczenia matematyczne. Operacje na pojedynczych znakach alfabetu przeprowadzane przez wykwalifikowane osoby, jak to miało miejsce podczas Pierwszej i Drugiej Wojny Światowej, ustąpiły miejsca operacjom na bitach przeprowadzanym przez nowoczesne, często dedykowane procesory. Większość z obecnie stosowanych algorytmów w dużej mierze bazuje na teorii liczb, liczbach losowych, zagadnieniach liczb pierwszych, arytmetyki modulo, faktoryzacji. Co ciekawsze, bezpieczeństwo tych metod nie polega na utrzymywaniu w tajemnicy metody szyfrowania. Algorytmy kryptograficzne są dostępne publicznie [3, 4], tak by każdy mógł poddać je dokładniej analizie.

W poniższej pracy zaprezentowano praktyczne aspekty nowoczesnych technik kryptograficznych, ze szczególnym skupieniem uwagi na algorytmy i protokoły kryptograficzne zapewniające:

- uwierzytelnianie (*ang. authentication*),
- autentyczność (*ang. authenticity*),
- integralność (*ang. integrity*).

2 Przegląd technik kryptograficznych

2.1 Bezpieczeństwo i integralność danych

Jednym z podstawowych zadań kryptografii jest zapewnienie bezpieczeństwa danych. Poprzez bezpieczeństwo należy rozumieć tu sytuację, gdzie nawet w przypadku przejęcia określonych porcji informacji przez osoby niepowołane, przechwycone zasoby są bezużyteczne.

W przypadku wszelkich metod gromadzenia danych z wykorzystaniem nośników informacji, spotykamy się z pojęciem integralności. Poprzez integralność systemu plików należy rozumieć kontrolę nad wszelkimi modyfikacjami zasobów (plików bądź katalogów), szczególnie tych o kluczowym znaczeniu dla systemu. W rozwiązaniach informatycznych z pojęciem integralności można spotkać się na co dzień. Przykładem mogą być sumy kontrolne protokołu TCP. W systemie NetBSD za sprawą mechanizmu *vexec* jądro jest w stanie kontrolować integralność plików, na których przeprowadzane są operacje przy pomocy wywołań systemowych. Podobną funkcjonalność w systemie OpenBSD pełni przeniesiony mechanizm *vexec*, dostarczany przez projekt Stephanie (łata na źródła systemowe).

2.2 Szyfrowanie symetryczne

Najpopularniejszym sposobem szyfrowania jest szyfrowanie symetryczne. Nazwa pochodzi od sposobu dokonywania operacji szyfrowania i deszyfrowania - obie dokonywane są przy pomocy tego samego klucza, od którego zależy bezpieczeństwo. W przeciwieństwie do popularnych technik szyfrowania przy pomocy operacji XOR, obecne metody szyfrowania oparte na haśle wybieranym przez użytkownika powinny spełniać odpowiednie założenia [5].

2.3 Kryptografia z kluczem publicznym

Do czasu wynalezienia kryptografii z kluczem publicznym sądzono, że wymiana właściwego klucza szyfrującego poprzez publiczny kanał danych nie jest możliwa. Teorię tą obalili Whitfield Diffie i Martin Hellman, choć tak na prawdę za twórcę tego sposobu szyfrowania był Ralph Merkle. Najpopularniejszym przykładem kryptosystemu klucza publicznego jest RSA będący asymetrycznym szyfrem, w którym proces szyfrowania odbywa się przy pomocy dwóch par kluczy: klucza prywatnego i publicznego odbiorcy wiadomości, oraz publicznego i prywatnego nadawcy. Nadawca szyfrując wiadomość,

pobiera klucz publiczny adresata, po czym używa go do szyfrowania wiadomości. Po przesłaniu wiadomości, odbiorca przeprowadza proces deszyfrowania danych, używając swojego klucza prywatnego. Ten system daje gwarancję, że treść zaszyfrowanej wiadomości będzie możliwa do odczytania jedynie przez osobę, do której jest adresowana. Przy wykorzystaniu odmiennego sposobu użycia kluczy (nadawca podpisuje wiadomość swoim kluczem prywatnym, a następnie szyfruje kluczem publicznym docelowego odbiorcy), możliwe jest potwierdzenie tożsamości nadawcy.

3 Podejście praktyczne do kryptografii

Mimo pozornej złożoności posługiwania się technikami kryptograficznymi, sposób szyfrowania danych wybranymi przez nas algorytmami jest bardzo prosty. Ich wybór jest z kolei niezwykle bogaty. Od najprostszych monoalfebetycznych szyfrów przestawieniowych i podstawieniowych (które praktycznie nie są metodami szyfrowania), przez ogólnie znane szyfry symetryczne takie jak DES, aż po zaawansowane techniki szyfrów opartych o szyfrowanie asymetryczne. Przykładem może być OpenSSL.

OpenSSL jako otwarta i wolna biblioteka kryptograficzna daje programiście możliwość rozszerzenia swoich aplikacji o funkcje kryptograficzne. Ilość udostępnianych algorytmów jest spora: DES, AES, algorytm DSA, kryptograficzne funkcje skrótu (md*, sha*, rmd160), certyfikaty x509. Dobrym źródłem informacji do poznania możliwości biblioteki są strony pomocy dostarczane wraz z pakietem biblioteki.

Mimo, iż znajomość podstaw teoretycznych kryptografii jest niezbędna do zrozumienia podstaw działania określonych algorytmów i protokołów, to głównym powodem stosowania kryptografii są praktyczne zastosowania. Pomija się tutaj programy wyłącznie komercyjne bądź te, które dostępne są na komercyjnych platformach, skupiając się na najczęstszym zastosowaniu szyfrowania w systemach rodziny UNIX.

4 Podstawy tworzenia bezpiecznej transmisji danych

4.1 OpenSSL generacja certyfikatów

SSL jest standardem stworzonym przez firmę Netscape w celu przeprowadzania bezpiecznych połączeń między końcówkami obsługującymi protokół. SSL jest doskonałym rozwiązaniem w przypadkach, w których stronom łączącym się wzajemnie zależy na poufności i integralności danych. Możliwe jest również udowodnienie, iż strona, z którą następuje połączenie jest tą, za którą się podaje. Dzieje się tak z powodu drzewiastej struktury wzajemnie ufających sobie certyfikatów, które służą jako swojego rodzaju dowód tożsamości. Wydanie certyfikatu nadrzędnego urzędu certyfikacyjnego (*ang. CA - Certificate Authority*) możliwe jest w przypadku podpisania nowo tworzonego certyfikatu przez nadrzędny podmiot (przykładem mogę być certyfikaty podpisywane przez NASK) lub samodzielnego podpisania certyfikatu (ang. self signed certificate). To drugie rozwiązanie spotykane jest najczęściej. Szczególnie, gdy głównym powodem stosowania protokołu SSL jest zapewnienie szyfrowania. Nieinteraktywna metoda generacji certyfikatu składa się z kilku kroków. Pierwszym jest generacja prywatnego klucza RSA o wielkości 2048 bitów. W celu zapewnienia właściwego startu usłudze wykorzystującej protokół SSL nie zabezpiecza się klucza żadnym algorytmem symetrycznym:

```
$ openssl genrsa -out ca.key 2048
Generating RSA private key, 2048 bit long modulus
.............................................
..................+++
.........................+++
e is 65537 (0x10001)
```

Następnie wykorzystując jako podstawę identyfikacji późniejszego certyfikatu wcześniej stworzony plik:

```
$ cat ca.data
/C=PL/ST=Silesia/L=Czestochowa/O=Sample Company/OU=Sample Company/
CN=Sample Company CA/emailAddress=cert@foo.bar.pl
```

generujemy wniosek o certyfikat (*ang. CSR - Cerificate Signing Request*):

```
$ openssl req -new -key ca.key -out ca.csr -subj "`cat ca.data`"
```

Wygenerowane w ten sposób żądanie podpisujemy wcześniej stworzonym kluczem w celu uzyskanie podpisanego certyfikatu:

```
$ openssl x509 -req -in ca.csr -signkey ca.key -out ca.cert
Signature ok
subject=/C=PL/ST=Silesia/L=Czestochowa/O=Sample Company/
OU=Sample Company/CN=Sample Company CA/emailAddress=cert@foo.bar.pl
Getting Private key
```

Po tym następuje generacja właściwego certyfikatu, poprzez podpisanie go certyfikatem CA:

```
$ openssl genrsa -out server.key 2048
Generating RSA private key, 2048 bit long modulus
..............+++
.............................+++
e is 65537 (0x10001)

$ cat server.data
/C=PL/ST=Silesia/L=Czestochowa/O=Sample Company/
OU=Sample Company/CN=foo.bar.pl/emailAddress=cert@foo.bar.pl

$ openssl req -new -key server.key -out server.csr
-subj "`cat server.data`"

$ openssl x509 -CA ca.cert -CAkey ca.key -req -CAcreateserial
-in server.csr -out server.cert
Signature ok
subject=/C=PL/ST=Silesia/L=Czestochowa/O=Sample Company/
OU=Sample Company/CN=foo.bar.pl/emailAddress=cert@foo.bar.pl
Getting CA Private Key
```

Tak wygenerowane certyfikaty i klucze mogą posłużyć do konfiguracji bezpiecznych usług. W przypadku serwerów wspierających obsługę połączeń SSL należy pamiętać o zapewnieniu nieinteraktywnego startu. Dlatego też wyżej wspomniana metoda generacji kluczy pomija ich szyfrowanie, gdyż to wymagałoby podawania hasła przy każdym uruchomieniu sieciowego demona. Ważnym zagadnieniem jest również wartość pola *CN* certyfikatu z pliku *server.data*: jest to nazwa serwera, z którym klienci będą się łączyć. W przypadku niezgodności rzeczywistej nazwy serwera z nazwą umieszczoną w polu, użytkownik będzie każdorazowo informowany o zainstniałej sytuacji.

5 Podsumowanie

Znaczenie kryptografii dostrzegli również twórcy i projektanci systemów operacyjnych. Przykładem mogą być wolnodostępne systemy operacyjne, w których wsparcie dla operacji kryptograficznych dostarczane jest w warstwie jądra systemu, w postaci sterowników bądź modułów stworzonych specjalnie do tego celu. Możliwe jest dzięki temu pełne wykorzystanie kart sieciowych bądź innych urządzeń, stworzonych specjalnie z myślą o intensywnym wykorzystaniu technik kryptograficznych. Systemy te umożliwiają również akcelerację operacji w przestrzeni użytkownika (silnik biblioteki OpenSSL dostępny w systemie OpenBSD).

5.1 Rozwój kryptografii

Obecny stan wiedzy dotyczącej metod, technik i sposobów szyfrowania, ogólnych zasad podczas projektowania odpowiednich protokołów oraz algorytmów ograniczany jest głównie przez możliwości obliczeniowe obecnych komputerów. Już dawno minęły czasy, w których łamanie szyfrów polegało jedynie na odnalezieniu słabych punktów w sposobie szyfrowania. Obecne metody szyfrowania opierają się na wspomnianych wcześniej publicznie dostępnych algorytmach, dlatego też całą potęgę szyfru stanowią używane klucze. Procesem hamującym rozwój kryptografii są również ustawy patentujące kod, rządowe restrykcje dotyczące publikowania oraz rozpowszechniania wszelkich źródeł wiedzy dotyczącej kryptografii [6]. W środowisku Open Source znane są również sprawy sądowe osób związanych z rozwojem kryptografii [7, 8, 9]. W szyfrze Lucifer pod wpływem nacisków ze strony NSA zmieniono efektywną wielkości bloku z 64 bitów na 56. Ograniczenie siły algorytmu było wynikiem obawy przed niemożnością złamania bloku 64-bitowego. Jednym z powodów nałożenia ograniczeń prawnych na powszechny dostęp do technik kryptograficznych jest ponadto obawa rządów państw o wykorzystywanie szyfrów przez organizacje i grupy terrorystyczne.

5.2 Przyszłość kryptografii

Kryptografia zmierza do uzyskania jeszcze bardziej wydajnych algorytmów szyfrowania, protokołów kryptograficznych, miniaturyzacji, implementacji w dedykowanych procesorach czy też układach scalonych. Ogromne znaczenie dla kryptografii ma matematyka i fizyka. W dużej mierze od rozwoju matematycznych metod faktoryzacji zależy bezpieczeństwo obecnych szyfrów bazujących na liczbach pierwszych. Fizyka odgrywa ogromną rolę w kryptografii kwantowej, w której nośnikiem informacji kryptograficznych jest

foton. Mimo, iż kryptografia kwantowa jest daleko od popularnych zastosowań, stanowi niezwykle ciekawą ścieżkę dalszego rozwoju. Nie jest jednak utopią - jak pokazują ostatnie doniesienia, kryptografia kwantowa osiągalna jest już dla zakładów badawczych [10]. Prowadzone są również badania nad rozwojem technik z wykorzystaniem teorii kwantowej [11]. W realnych granicach mieści się za to kryptografia bazująca na krzywych eliptycznych, oparta na problemie obliczania logarytmu dyskretnego krzywych eliptycznych. Obecnie trwają prace nad ustandaryzowaniem tego rodzaju szyfrowania [12]. Wraz ze wzrostem złożoności technik wykorzystanych do kryptografii, rosną również komplikacje związane z użytym algorytmem. Podobnie jak w kryptografii symetrycznej problemem staje się sposób wymiany klucza, w kryptografii klucza publicznego prędkość dokonywania operacji, o tyle na przykład w kryptografii kwantowej problemem staje się szum oraz zakłócenia spowodowane wiązką fotonów.

Literatura

[1] Jak matematycy polscy rozszyfrowali Enigmę
http://www.impan.gov.pl/PTM/Rejewski/enigma.pdf.

[2] Factorization of RSA-576
http://www.rsasecurity.com/rsalabs/challenges/factoring/rsa576.html.

[3] National Institute of Standards and Technology *http://www.nist.gov*.

[4] RSA Security Inc. *http://www.rsasecurity.com/*.

[5] Password-Based Cryptography Standard *ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2-0.pdf*.

[6] Crypto law survey *http://rechten.uvt.nl/koops/cryptolaw/*.

[7] Bernstein v. United States *http://cr.yp.to/export.html*.

[8] Legal Cases - Crypto - Bernstein v. US Dept. of Justice
http://www.eff.org/Privacy/ITAR_export/Bernstein_case/.

[9] Niels Provos *http://www.citi.umich.edu/u/provos/*.

[10] NIST System Sets Speed Record For Generation of Quantum Keys for 'Unbreakable' Encryption *http://www.nist.gov/public_affairs/releases/quantumkeys.htm*.

[11] Physics Laboratory, Quantum Information *http://qubit.nist.gov/*.

[12] Elliptic Curve Cryptography Standard
http://www.rsasecurity.com/rsalabs/pkcs/pkcs-13/index.html.