1. Rozproszone łamanie szyfrów

Piotr Szpryngier

Politechnika Gdańska,

Wydział Elektroniki, Telekomunikacji i Informatyki,

Katedra Architektury Systemów Komputerowych

e-mail: piotrs@eti.pg.gda.pl

Streszczenie

W rozdziale zaprezentowano podstawowe techniki łamania szyfrów symetrycznych i asymetrycznych o stosunkowo niewielkiej długości kluczy. Przedstawiono ogólną charakterystykę metod łamania szyfrów. Ilustracją tych metod jest zaprezentowana aplikacja służąca do łamania haseł lub badania odporności haseł na odgadnięcie.

Słowa kluczowe: łamanie szyfrów, łamanie haseł, obliczenia rozproszone, RSA, faktoryzacja dużych liczb.

1.1. Wprowadzenie

We współczesnej kryptografii bezpieczeństwo algorytmu szyfrującego jest oparte na kluczu, a nie na utrzymywaniu algorytmu w tajemnicy (nb. wszystkie takie próby zakończyły się niepowodzeniem). W literaturze opisano sześć klas metod łamania szyfrów. Każda z nich zakłada, że kryptoanalityk posiada pełną wiedzę o stosowanym algorytmie szyfrowania. Kryptoanaliza, czyli działania i prace podejmowane przez kryptoanalityka, pozwala (oczywiście gdy jest zakończona sukcesem) na odtworzenia tekstu jawnego lub klucza na podstawie szyfrogramu. Zajmuje się również wyszukiwaniem słabych punktów systemów kryptograficznych, punktów, które mogłyby otworzyć drogę do poznania tekstu jawnego lub klucza. Działania prowadzące do utraty tajności klucza szyfrującego, wykorzystujące takie inne metody (nie kryptoanalizę, ale np. socjotechnikę), noszą ogólnie miano próby kompromitacji klucza. Łamanie szyfru polega na znalezieniu klucza szyfrującego (deszyfrującego w przypadku asymetrycznym) wiadomość zaszyfrowaną albo znalezieniu tekstu jawnego, który odpowiada danemu szyfrogramowi. Wyróżniamy następujące metody łamania szyfrów [2]:

1. Łamanie z szyfrogramami – znamy tylko szyfrogram, tryb pracy i długość klucza. Kryptoanalityk posiada jedynie kilka szyfrogramów utworzonych z wykorzystaniem tego samego algorytmu i klucza. Celem jest znalezienie

tekstów jawnych odpowiadających innym szyfrogramom utworzonym z wykorzystaniem tego właśnie klucza bądź klucza szyfrującego. Ta metoda nosi także nazwę łamania brutalnego, bowiem polega na badaniu wszystkich możliwych kombinacji bitów klucza aż do znalezienia tej właściwej, dzięki której można dokonać przekształcenia odwrotnego do szyfrowania.

2. Łamanie ze znanym tekstem jawnym – obok szyfrogramu, trybu pracy i długości klucza znamy także przynajmniej fragment tekstu jawnego, np. początek lub koniec tekstu jawnego. W tej metodzie kryptoanalityk ma w dyspozycji kilka szyfrogramów i odpowiadających im tekstów jawnych. Szyfrogramy są utworzone przy użyciu tego samego klucza tajnego. Celem jest znalezienie tekstów jawnych innych szyfrogramów utworzonych wykorzystaniem tego klucza (np. zapis sesji SSL) bądź tego klucza.

3. Łamanie z wybranym tekstem jawnym. Ta metoda jest rozwinięciem poprzedniej. Kryptoanalityk otrzymuje sposobność wybrania tekstu jawnego i odpowiadającego mu szyfrogramu. Pozwala to uzyskać więcej informacji o kluczu i dzięki temu można przyśpieszyć łamanie.

4. Łamanie z adaptacyjnym wyborem tekstu jawnego. Jest to specyficzne rozwinięcie łamania z wybranym tekstem jawnym. Kryptoanalityk wykonuje kolejne próby, wybierając tekst jawny do szyfrowania wg swojego uznania i pewnych wskazówek uzyskanych z wyników poprzednich szyfrowań. Może np. podzielić duży blok tekstu jawnego na mniejsze porcje i przy łamaniu z adaptacyjnie wybranym tekstem jawnym może wybrać kolejny blok tekstu jawnego. W kolejnym kroku wybiera inny blok tekstu jawnego, biorąc pod uwagę skutki pierwszego wyboru i kolejne uzyskane wyniki.

5. Łamanie z wybranym szyfrogramem. Ta metoda jest stosowane głównie w kryptosystemach z kluczem publicznym. Kryptoanalityk dysponuje zbiorem szyfrogramów i odpowiadających im tekstów jawnych. Może nawet sam je wytworzyć, używając do tego celu dowolnie lub specyficznie (metoda adaptacyjna) wybranych tekstów jawnych i klucza publicznego (który z definicji jest znany). Z tego zbioru wybiera pewne szyfrogramy i podejmuje próby znalezienia klucza szyfrującego (odszyfrowującego), mając do dyspozycji także tekst jawny.

6. Łamanie z wybranym kluczem. W tej metodzie kryptoanalityk nie zna klucza (nie może go wybrać), ale ma pewną wiedzę o powiązaniach pomiędzy różnymi kluczami stosowanymi w tym samym systemie (np. informacje o generatorach liczb pseudolosowych, technikach używanych do uzyskania tzw. ziarna losowego, itp.)

Dodatkowo wypada wspomnieć o innej metodzie, wykorzystującej techniki manipulacji socjotechnicznej.

- Łamanie pod przymusem („z gumową pałką”). Atakujący używa siły fizycznej, przymusu, przekupstwa, gróźb, szantażu, itp., by uzyskać klucz szyfrujący. Jest to metoda skuteczna dużo bardziej niż typowe zabiegi

kryptoanalityczne, ale raczej niemożliwa do zastosowania w rozproszonym systemie obliczeniowym.

Istnieją metody łamania szyfrów symetrycznych (np. DES) zbudowanych wg schematu zwanego siecią Feistela (FN) o złożoności obliczeniowej mniejszej niż łamanie brutalne, np. kryptoanaliza różnicowa [2]. Wymagają one jednak dużych ilości pamięci i z tego powodu są trudne do zastosowania w środowisku rozproszonym ze względu na konieczność w miarę swobodnego dostępu do różnych pośrednich wyników obliczeń.

Łamania kluczy systemach z kluczem publicznym, oprócz opisanych powyżej metod w p. 5 i 6, mogą wykorzystywać słabości związane z niewłaściwym stosowaniem protokołów z kluczem publicznym, a także technik tworzenia kluczy (zarządzania kluczami). Podawane w literaturze przedmiotu (np. [1,2,4,5]) skuteczne ataki na RSA były możliwe, bowiem użytkownicy nie przestrzegali ograniczeń związanych ze stosowaniem RSA, zestawionych następująco:

- Znajomość jednej pary wykładników szyfrowania/deszyfrowania dla danego modułu umożliwia atakującemu faktoryzację modułu oraz ewentualnie obliczenie innych par wykładników szyfrowania/deszyfrowania, bez konieczności faktoryzacji *n*.
- Wnioskiem z powyższego jest stwierdzenie, że wspólny moduł nie powinien być używany w protokołach wykorzystujących RSA w sieci telekomunikacyjnej.
- Aby zapobiec atakom skierowanym na małą wartość wykładnika szyfrującego, komunikaty powinny być dopełniane losowymi wartościami w taki sposób, by uzyskać ich długość zbliżoną do modułu *n*.
- Wykładniki (e,d) wybierane w protokole powinny być duże.

Kilka lat temu doniesiono o złamaniu kryptosystemu RSA o znacznej długości klucza (1024 bity), ale okazało się, że projektanci zastosowali w systemie kiepskiej jakości generatory ciągów pseudolosowych stosowanych do budowy elementów kluczy i tym sposobem te klucze złamano. Natomiast rzeczywiście złamano system z modułem 1039 bity (faktoryzacja liczby $n=2^{1039}-1$) [3,4].

Możliwe są także ataki na protokoły stosujące RSA do szyfrowania i podpisywania, jeśli ich użytkownicy nie przestrzegają określonych zasad:

- Nie wolno podpisywać ani szyfrować ciągów bitowych nieznanego pochodzenia bądź podstępnie podsuniętych do przetworzenia przez atakującego.
- Nie wolno używać tej samej pary kluczy (d,e) do różnych celów i w różnych protokołach, np. do szyfrowania i podpisywania równocześnie.

Obecnie stosowane metody łamania RSA wykorzystują metody tzw. *sit liczbowych* i rozproszone środowisko obliczeniowe [3,4,5].

1.2. Rozproszone łamanie szyfrów symetrycznych o stosunkowo niewielkiej długości kluczy

Łamanie szyfrów symetrycznych w środowisku rozproszonym o stosunkowo niewielkiej długości kluczy, np. DES polega na tym, że usiłujemy znaleźć klucz szyfrujący, by odszyfrować szyfrogram, albo bezpośrednio odtworzyć tekst jawny z szyfrogramu. Operację można scharakteryzować następująco:

Parametry algorytmu

Długość bloku – 64 bitów, efektywna długość klucza – 56 bitów (w 64-bitowej sekwencji występuje 8 bitów parzystości).

Założenia

Dysponujemy szyfrogramem i odpowiadającym mu fragmentem tekstu jawnego. Ta metoda łamania to wg podanej poprzednio klasyfikacji łamanie brutalne ze znanym tekstem jawnym.

Zrównoleglenie obliczeń poprzez podział danych

Polegać to może na tym, że aplikacja internauty otrzymuje blok tekstu jawnego i blok szyfrogramu oraz początkowe bity klucza (np. o liczbie bitów 16-24). Każdy użytkownik otrzymuje inną kombinację początkowych bitów hipotetycznego klucza.

Zadaniem komputera użytkownika jest zbadanie pozostałych bitów klucza (32-40b) w celu określenia, czy są tą właściwą kombinacją pozwalającą łącznie z początkowymi 16-24 bitami uzyskać z szyfrogramu zadany tekst jawny;

Zaletą tej metody przetwarzania jest minimalna komunikacja (krótkie ciągi danych, gdy kod algorytmu jest obecny na komputerze internauty), a obliczenia odbywają się niemalże w czasie rzeczywistym (niekiedy kilka sekund). Natomiast czas obliczeń można regulować, ustając długość badanej sekwencji bitów.

1.3. Rozproszone łamanie haseł typu *brute-force* (badanie jakości haseł)

Badanie jakości haseł, jak również ich łamanie bądź odzyskiwanie na drodze znalezienia równoważnego ciągu znaków to jedno z najbardziej popularnych zastosowań kryptografii. Łamanie brutalne haseł o określonej długości, polegające na badaniu wszystkich kombinacji znaków „drukowalnych” danego alfabetu, może być znacząco przyśpieszone poprzez zastosowanie tzw. łamania słownikowego. Nie polega ono na testowaniu wszystkich możliwych kluczy w porządku numerycznym, ale na sprawdzeniu najpierw najłatwiejszych kluczy (ciągów znaków tworzących hipotetyczne hasło). Użytkownicy często wybierają hasła w miarę łatwe do zapamiętania, często zawierające ciągi znaków w jakiś sposób związane z konkretną osobą bądź spośród słów potocznie używanych. Już w latach 60-tych ub. wieku udawało się złamać 30–40% haseł w przeciętnych systemach komputerowych dzięki tej metodzie

[2]. Listę tych łatwiejszych haseł tworzy się, kierując następującymi przesłankami:

- Nazwisko użytkownika, jego inicjały, nazwa konta w systemie i wszelkie kombinacje słów w oparciu o te bazowe, zamieniając duże litery na małe, np. literę o na znak cyfry 0, itp.;
- Słowa z różnych baz danych zawierających imiona żeńskie i męskie, nazwy biologiczne, geograficzne, popularne medyczne, nazwy bohaterów filmów, książek, komiksów, programów telewizyjnych, słuchowisk radiowych, postaci internetowych, powszechnie używanych wyrażeń i słów wulgarnych, itp., dokonując także podobnych jak powyżej modyfikacji, a także zdrobnień nazw;
- Różne permutacje słów z podpunktu a i b, z wykorzystaniem zamiany małych liter na duże, cyfr na litery i odwrotnie, itp., uzupełnionych o wszystkie 4-cyfrowe kombinacje liczbowe stosowane w niektórych systemach operacyjnych jako tzw. wydłużenie hasła (tzw. *salting*) przeciwdziałające ułatwieniom w jego złamaniu;
- Listę popularnych słów obcojęzycznych, używanych w różnych środowiskach kulturowych i zawodowych, określenia żargonowe, oraz ich odpowiedniki sylabowe w danym języku – poddane takim samym permutacjom jak w podpunkcie c;
- Kombinacje słów – pary, trójki słów z podpunktów a-d.

Tak przygotowana lista potencjalnych haseł może być wcześniej wygenerowana w trybie *offline*, przekształcona z wykorzystaniem wskazanych jednokierunkowych funkcji skrótu (np. DES-CBC, MD5, SHA-1, RIPEMD) i zapisana do pliku. Przed rozpoczęciem jakiegokolwiek łamania brutalnego najpierw należy sprawdzić tak wygenerowaną listę skrótów w celu zbadania, czy nie występuje zgodność z danym skrótem hasła.

Ułatwieniem w łamaniu haseł mogą być słabości stosowanych funkcji skrótu. Już w połowie lat 90-tych ub. wieku opublikowano doniesienia o stosunkowo łatwych możliwościach wystąpienia kolizji dla jednokierunkowych funkcji skrótu MD-4 i MD-5 [7]. Kolizją oznaczamy sytuację, w której znajdujemy ciąg znaków skracający się do tej samej wartości skrótu. W przypadku funkcji MD-4 i MD-5 jest to znacznie łatwiejsze niż łamanie brutalne. Od tego czasu nie zaleca się stosowania tych funkcji skrótu w poważnych zastosowaniach kryptografii, ale nadal wielu producentów, np. systemów operacyjnych używa funkcji MD-5 do tworzenia skrótów haseł użytkowników przechowywanych w plikach systemowych. Od części tych słabości nie jest wolna także stosowana, m.in. w Polsce do tworzenia tzw. kwalifikowanych podpisów elektronicznych, funkcja skrótu SHA-1 [6]. Z tego też względu opracowano i zaczęto stosować dłuższe (256 do 512 bitów) funkcje skrótu tworzące standard SHA-2. Obecnie trwają prace nad standardem SHA-3.

W dalszej części punktu przedstawiono charakterystykę obliczeń dla rozproszonego łamania brutalnego haseł, wykorzystujących zgłoszone do współpracy komputery (część ich zdolności obliczeniowych):

Rozproszony system obliczeń

System obliczeń składa się z grupy węzłów wewnętrznych (przyjmujących zlecenia wykonania obliczeń, dzielących dane zadania na paczki, organizujące obliczenia we współpracy z zewnętrznymi węzłami), węzłów zewnętrznych (przekazujących na żądanie kody aplikacji i kolejne paczki danych do współpracujących komputerów internautów) i komputerów wykonujących obliczenia (zgłoszonych dobrowolnie użytkowników biorących udział w całym przedsięwzięciu);

Parametry obliczeń

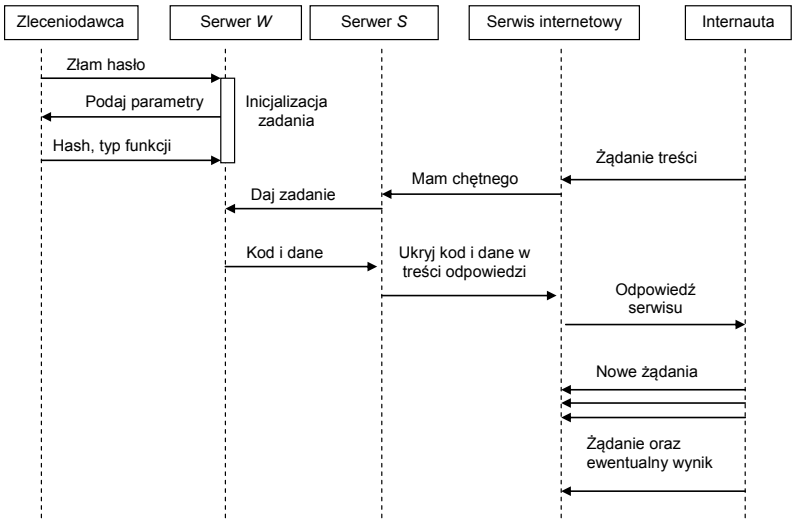
Długość wartości skrótu – 128b (MD5), 160b (SHA-1, RIPEMD), 256b (SHA-256), długość ciągu znaków hasła – od 1 do pewnej górnej granicy.

Założenia

Dysponujemy wartością skrótu hasła i typem stosowanej funkcji skrótu, pomocna może być informacja o użytkowniku (rozbudowa o elementy tzw. łamania słownikowego). Ten sposób znajdowania haseł jest podobny do podanej poprzednio klasyfikacji metody łamania z szyfrogramem.

Zrównoleglenie obliczeń poprzez podział danych

Aplikacja uruchomiona na komputerze użytkownika (współpracującego z rozproszonym systemem obliczeń) otrzymuje wartość skrótu, kod funkcji do obliczeń skrótu i początkowy ciąg znaków hasła. Każdy komputer użytkownika otrzymuje inną kombinację początkowych znaków hipotetycznego hasła (patrz rys. 1.1).



Rys. 1.1. Ilustracja sekwencji działań podczas rozproszonego łamania haseł

Zadaniem aplikacji na komputerze użytkownika jest zbadanie pozostałych hipotetycznych ciągów znaków hasła (uzupełnienie zadanego ciągu początkowego) o długości od 1 znaku do 3 znaków większej od zadanej w celu ustalenia, czy są one łącznie tą właściwą kombinacją pozwalającą wyliczyć zadaną wartość funkcji skrótu.

Zalety

Minimalna komunikacja, a obliczenia niemalże w czasie rzeczywistym (niekiedy kilka sekund); czas obliczeń można regulować, ustając długość badanej sekwencji znaków.

1.4. Przykład aplikacji

W dalszej części tego podpunktu przedstawiono przykład aplikacji rozproszonego łamania haseł zrealizowanej poza środowiskiem Comcute, ale przygotowanej z myślą o przyszłej implementacji w systemie laboratoryjnym Comcute Politechniki Gdańskiej. Przewidziano, że będą łamane hasła przechowywane w postaci skrótów utworzonych z wykorzystaniem jednokierunkowych funkcji MD5 oraz SHA-1. Aplikacja została stworzona z wykorzystaniem języków Python, C++ i JavaScript.

Architektura aplikacji obejmuje następujące elementy:

- Serwer sterujący – główny element, zwany również kontrolerem. Jest to wielowątkowy serwer TCP/IP, który zajmuje się dzieleniem przestrzeni problemu na mniejsze zakresy, przydziałem zadań oraz przyjmowaniem rozwiązań, a także ich weryfikacją. Ta część de facto symuluje zachowanie warstw W i Z środowiska COMCUTE.
- Baza danych MySQL, w której przechowywane są informacje o użytkownikach systemu, wygenerowanych zadaniach, a także problemach, które należy jeszcze zaadresować. Jest to symulacja warstwy danych serwerów *W* środowiska Comcute.
- Moduł serwera WWW. Jest to symulacja serwerów *S* środowiska Comcute.
- Klienci:
  - pierwszego typu – łączą się poprzez aplikację desktopową bezpośrednio z kontrolerem, będąc świadomymi uczestnikami obliczeń (*voluntary computing*);
  - drugiego rodzaju, nie są świadom faktu brania udziału w projekcie, stąd nie mogą oni w sposób bezpośredni komunikować się z kontrolerem.
- Serwer sterujący - Jest to aplikacja napisana w języku Python, zadaniem której, jest podział przestrzeni problemu na zakresy obliczeń stanowiące części zadań serwowanych klientom. Oprócz tego odpowiedzialny jest on też za przyjmowanie i weryfikację wyników zgłoszonych przez klientów.

Komunikacja z kontrolerem przebiega przy użyciu gniazd TCP/IP, z wykorzystaniem protokołu tekstowego przypominającego np. protokół FTP. W komunikacji udział biorą obiekty zadań, opakowane w formie dokumentów

formatu XML. Przykładowe zadanie w formacie XML ma więc postać (dla klientów pierwszego typu):

```
<?xml version="1.0" encoding="UTF-8" ?>
<task id="17">
    <problem>
    <algorithm>md5</algorithm>
    <dictionary>ABCDEFGHIJKLMNOPQRSTUVWXYZ</dictionary>
        <range_start>AAAAA</range_start>
        <range_end>EEEAA</range_end>
        <expected_length>5</expected_length>
        <hash>9da4fc50e09e5eeb8ae8149ef4f23792</hash>
    </problem>
    <challenge>a challenging string</challenge>
</task>
```

Odpowiedź zgłaszana jest natomiast w następujący sposób:

```
<?xml version="1.0"?>
<task id="17">
    <result found="T">CCCCC</result>
    <challenge>a challenging string</challenge>
</task>
```

Co oznacza że zostało znalezione rozwiązane. W tym momencie można dokonać sprawdzenia, czy rzeczywiście znaleziono kolizję (hasło). W tym celu na serwerze (kontrolerze) należy wykonać obliczenia sprawdzające, czy znaleziony łańcuch znaków skraca się do zadanej wartości przy wykorzystaniu wskazanej jednokierunkowej funkcji skrótu.

Wprowadzony został moduł serwera WWW (*WebServer*), będącego swoistym rodzajem *proxy*, jako pośrednik pomiędzy kontrolerem a klientami (drugiego typu). W tym przypadku mamy do czynienia z wieloma użytkownikami, którzy nie są zbyt świadomi faktu użyczania swej mocy obliczeniowej. Z uwagi na pewną trudność związaną z obsługą protokołu zgłaszania odpowiedzi, zaproponowany został element pośredniczący pomiędzy klientem niejawnym, a kontrolerem.

Kod aplikacji serwera został on napisany w języku Python i zapewnia podstawową oczekiwaną funkcjonalność, tj. jest w stanie udostępniać pewne zasoby internetowe (strony www, grafikę czy też aplety), jak również obsługiwać żądania rodziny GET/POST i inne, jak np. możliwość dynamicznego wypełniania szablonów dokumentów, informacjami charakterystycznymi dla zadań naszego systemu. Serwer jest pośrednikiem i z punktu widzenia kontrolera jest użytkownikiem reprezentującym wszystkich klientów drugiego typu. W momencie nawiązywania połączenia, przedstawia się więc on jako inna maszyna, następnie pobiera w jej imieniu zadanie. Zadanie to przesyła, używając jednego z dostępnych silników, dla klienta – zwykle przeglądarki internetowej - po czym odbiera od klienta wynik, który znowu raportuje do kontrolera, również w imieniu klienta. Uzupełnienie działania serwera stanowią proste moduły, zwane też silnikami, które realizują właściwą funkcjonalność obliczeniową. Przygotowane zostały 3 takie implementacje:

- Silnik JavaScript pozwala wstawić skrypt na dowolną stronę, który raz uruchomiony, zajmuje się wykonaniem obliczeń, a następnie odesłaniem rezultatu do serwera, z którego został pobrany. Realizuje to przy wykorzystaniu technologii AJAX. Skrypt ma możliwość wprowadzenia opóźnień czasowych, dzięki którym pozostaje niezauważalny, gdyż zabiera relatywnie mało zasobów systemowych.
- Silnik Flash został zrealizowany tylko w częściowej funkcjonalności. Nie występuje tu osadzenie obiektu (WDF) na stronie, są dostępne jedynie moduły w postaci plików typu .FLA i .AS, z którymi to współpracuje zmodyfikowana wersja WebServera.
- Silnik apletów Java uruchamia aplety Javy poprzez mechanizm JNLP. Poprzez skrypt, który je ładuje, są wstawiane parametry do obliczeń. Aby można było odesłać wyniki obliczeń nie podpisując apletu, odsyła się je na WebServer za pomocą java.net.URLConnection.

Klient drugiego typu na żądanie (GET/POST) ze strony serwera otrzymuje skrypt (JavaScript, aplet Java, Flash), zawierający funkcję skrótu (MD5, SHA1) i zakres danych do obliczeń. Przewidziano możliwość wykonania obliczeń na kartach graficznych z rodziny AMD (dawne ATI).

Po otrzymaniu wyniku obliczeń od klienta serwer przekazuje go do kontrolera.

Zastosowane technologie: MPI, Python, C++, JavaScript.

1.5. Rozproszone łamanie szyfrów asymetrycznych o stosunkowo niewielkiej długości kluczy, np. RSA z modułem 512b

Aktualne doniesienia [3,4] na temat łamania kluczy RSA wskazują, że dokonano rozproszonego łamania RSA z kluczem (a faktycznie modułem n=786 bitów – ok. 193 cyfr dziesiętnych), zakończone 12 grudnia 2009 roku. Operacja przebiegała w czasie 2,5 roku z wykorzystaniem pracy setek komputerów. Oceny dokonywane przez różnych kryptoanalityków mówią, że złamanie klucza o długości 1024 bitów stanie się możliwe na przestrzeni najbliższych kilkunastu lat.

Charakterystykę rozproszonego łamania RSA opisano następująco:

Parametry

Tekst jawny o długości bloku – np. 128 b, klucz publiczny – (n, e).

Założenia

Dysponujemy szyfrogramem (np. podpisem) S i tekstem jawnym (skrót podpisywanej wiadomości) H oraz pewną początkową liczbą bitów klucza prywatnego albo zakresem badanych liczb pierwszych;

Zrównoleglenie obliczeń poprzez podział danych

Aplikacja internauty otrzymuje aplikację do badania liczb pierwszych, tekst jawny, szyfrogram oraz klucz publiczny. Każdy użytkownik otrzymuje inną

kombinację początkowych bitów hipotetycznego klucza prywatnego albo inny hipotetyczny przedział dla liczby poszukiwanej pierwszej, np. p. Zadaniem internauty jest znalezienie takiej pary liczb pierwszych p,q (problem faktoryzacji), że $n=pq$ oraz liczby d będącej tożsamościową odwrotnością e, tzn. $S^d \bmod n = H$, $H^{ed} \bmod n \equiv H$; Do badania, czy dana liczba x jest liczbą pierwszą, można użyć:

- Testu Fermata – dla liczby x i znanej liczby pierwszej a obliczamy $f \equiv a^{x-1} \pmod{x}$; jeśli $f \neq 1 \pmod{x}$, to x nie jest liczbą pierwszą, a w przeciwnym wypadku prawdopodobnie; test powtarzamy dla upewnienia się dla kilku liczb pierwszych a, np. 2, 5, 7, 11,13. Niestety, istnieją liczby złożone (liczby Carmichaela), dla których zachodzi $a^{x-1} \pmod{x} \equiv 1 \pmod{x}$, dla dowolnego a względnie pierwszego z x. Zaletą tego jest testu jest prostota obliczeń.
- Testu Millera-Rabina – który jest bardziej skomplikowany. Wykonujemy go następująco:
  Dla danego nieparzystego n niech $a^{n-1} = t2^s$, gdzie t jest nieparzyste.
  1. Jeśli liczba n jest liczbą pierwszą, to $a^{n-1} \equiv 1 \pmod{n}$ (zachodzi małe twierdzenie Fermata).
  2. Jeśli liczba n jest liczbą pierwszą i $a^{2^r}t \equiv 1 \pmod{n}$ dla $0 < r \leq s$, to
  3. $a^{2^{r-1}}t \equiv 1 \pmod{n}$ albo $a^{2^{r-1}}t \equiv -1 \pmod{n}$.
  4. Losujemy $0 < a < n$ i obliczamy: $a^t \bmod n$, $a^{2t} \bmod n$, $a^{4t} \bmod n$, . . . , $a^{n-1} \bmod n$.
  5. Jeśli ostatnia liczba z powyższej listy jest różna od 1 lub, przeglądając je od końca, pierwsza napotkana liczba różna od 1 jest tez różna od $n-1$, to a jest świadkiem, że n jest złożona.
  6. Dla każdej liczby złożonej n prawdopodobieństwo tego, że losowo wybrane a jest świadkiem, wynosi co najmniej 3/4.
  7. Jeśli wypróbujemy m losowo wybranych wartości a i żadna z nich nie okaże się świadkiem złożoności dla n, to z prawdopodobieństwem co najmniej $1 - (1/4)^m$ liczba n jest pierwsza. Im więcej badań przeprowadzimy, tym większe szanse na to, że liczba n jest liczbą pierwszą.
- Testu pierwszości Solovaya–Strassena, który korzysta z obliczeń symbolu Legendre'a i symbolu Jacobiego, przez co obliczenia są jeszcze bardziej skomplikowane. Pewność znalezienia liczby pierwszej opiera na podobnych statystycznych badaniach jak w przypadku testu Millera-Rabina.

Do faktoryzacji liczby n można użyć wielu algorytmów. Aktualnie najlepszym algorytmem do rozwiązania tego problemu jest ogólne sito ciał liczbowych (GNFS) [1,3], o podwykładniczej złożoności $\exp((c+O(1))n^{1/3} \log^{2/3} n)$ dla $c<2$.

Popularną metodą jest także użycie kwadratowego sita liczbowego. Korzystamy tutaj z prostej własności: jeśli $n = a^2 - b^2$, to $n = (a + b)(a - b)$. Poszukiwanie czynników liczby n polega na testowaniu kolejnych wartości a począwszy od najbliższej mniejszej od sqrt(n) i sprawdzenia, czy liczba $a^2 - n$ jest kwadratem. Jeśli tak, to znaleźliśmy takie $b^2 = a^2 - n$, a zatem rozkład liczby n.

Do faktoryzacji liczby n można także użyć np. metody Pollarda. Każdy internauta otrzymuje zakres liczb x i y do zbadania, czy różnica $(x - y)$ jest równa 0 mod n. Jeśli tak, to NWD$(x - y, n)$, czyli znaleziono dzielnik liczby n. Do generowania liczb można użyć generatora Brenta – który wykorzystuje wzór $x^2 - c$, $c \neq 2$ – i dla kolejnych iterowań dokonujemy sprawdzenia, czy zachodzi poszukiwana zależność.

Niezbędna jest jednak dodatkowa uwaga. Zanim przystąpimy do wyczerpującego ataku (np. faktoryzacja GNFS), należy wykonać różnego rodzaju badania i testy wykorzystujące słabości implementacji RSA. Należą do nich:

- Ataki elementarne, wykorzystujące fakt stosowania wspólnego modułu dla grupy par kluczy oraz tzw. „oślepiania", czyli skłonienia właściciela klucza prywatnego do podpisania specjalnie spreparowanej wiadomości, wyglądającej jak losowy ciąg bitów [1,2];
- Łamanie wg metody Wienera, opartej o mały wykładnik prywatny d. Użycie tej metody może złamać klucz prywatny w czasie liniowym na drodze kolejnych przybliżeń;
- Niekiedy w celu redukcji czasu weryfikacji podpisu elektronicznego jest wybierany niski wykładnik e (klucz publiczny), co też może być przedmiotem ataku [1,2]. Najpopularniejszy atak na niski wykładnik publiczny RSA jest oparty na twierdzeniu Coppersmitha. Wśród popularnych implementacji [5] opartych na tym twierdzeniu znajdujemy algorytm LLL (Lovasz, Lenstra, Lenstra jr.), atak Hastada, atak Franklina-Reitera, atak częściową ekspozycją klucza.
- Ataki na implementację RSA – skierowane na błędy techniczne popełnione przez informatyków wdrażających schemat. Wśród tych ataków wyróżniamy: ataki czasowe – opierają się na pomiarze czasu realizacji operacji kryptograficznych w kartach chipowych, ataki eksploatujące metody przyspieszania realizacji podpisu RSA, ataki na losowe uzupełnienie szyfrowanej wiadomości, ataki wykorzystujące wahania poboru energii podczas realizacji działań kryptograficznych (karty chipowe), a także ataki skierowane na wykrycie słabości zastosowanych generatorów liczb pseudolosowych.

Zalety: czas obliczeń można regulować, ustając długość zadanej sekwencji bitów lub wielkość przedziału badanych liczb pierwszych.

1.6. Podsumowanie

W rozdziale przedstawiono metody rozproszonego łamania szyfrów symetrycznych, łamania haseł bądź testowania ich odporności na odgadnięcie oraz wybrane metody łamania schematów asymetrycznych na przykładzie algorytmu RSA. Zaprezentowano także przykład aplikacji służącej do rozproszonego łamania haseł z wykorzystaniem mocy obliczeniowej komputerów wolontariuszy uczestniczących w obliczeniach. Z przedstawionych rozważań wynika, że wybrane obliczenia kryptograficzne realizowane w środowisku rozproszonym mają duże znaczenie praktyczne i pozwalają

znacznie przyśpieszyć różnego rodzaju działania mające znaczenie dla bezpieczeństwa różnych firm, organizacji i instytucji.

1.7. Wykaz literatury

1. A. J. Menezes, P. C. van Oorschot, S. A. Vanstone: *Handbook of Applied Cryptography* (Kryptografia stosowana), WNT 2005.
2. B. Schneier: *Kryptografia dla praktyków*, wyd.2, WNT 2000.
3. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev, P. Zimmermann: *Factorization of a 768-bit RSA modulus*, version 1.4, February 18, 2010, http://eprint.iacr.org/2010/006.pdf
4. http://pl.wikipedia.org/wiki/RSA_Factoring_Challenge
5. http://e-wiki.pl/articles/30RSA.php
6. *Cryptanalysis of SHA-1*, Schneier on Security, http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html
7. Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger: *MD5 considered harmful today*, http://www.win.tue.nl/hashclash/rogue-ca/