Biuletyn WAT
Vol. LVII, Nr 4, 2008

Wykorzystanie deszyfrowania progowego w bazach danych

BARTOSZ NAKIELSKI, JACEK POMYKAŁA*,
JANUSZ ANDRZEJ POMYKAŁA**

*Uniwersytet Warszawski, Wydział Matematyki Informatyki i Mechaniki, Instytut Matematyki,
02-097 Warszawa, ul. Banacha 2
**Wyższa Szkoła Menedżerska, Wydział Informatyki Stosowanej,
03-772 Warszawa, ul. Kawęczyńska 36

Streszczenie. Artykuł przedstawia nową metodę zabezpieczania baz danych — szyfrowanie z zastosowaniem technik kryptografii progowej. Dzięki temu możliwe jest zminimalizowanie zagrożenia, jakim są ataki napastników wewnętrznych (np. pracowników posiadających dostęp do bazy), gdyż wymagana jest współpraca kilku osób przy odszyfrowywaniu (czyli odczycie) danych. Jest to o tyle ważne, iż obecne systemy zabezpieczeń nie zapewniają takiej ochrony.

Opisane w pracy, oparte na RSA i CRT, deszyfrowanie ze zmiennym progiem oferuje pełną elastyczność progu (liczba osób, które muszą współpracować przy deszyfrowaniu) — może on być ustalany osobno dla każdych danych. Sprawia to, iż możliwe jest dokładne dopasowanie poziomu zabezpieczeń do indywidualnych potrzeb.

Pierwsza część artykułu wprowadza podstawowe pojęcia z zakresu matematyki (CRT) i kryptografii (RSA i kryptografia progowa), a także zawiera opis podstawowego modelu (jego elementy składowe oraz zastosowane algorytmy szyfrowania i deszyfrowania).

W drugiej części artykułu opisano modyfikacje modelu — wprowadzono pojęcia filtrów (ograniczających dostęp do danych podgrupom użytkowników) oraz grup kluczowych (składających się z osób, z których przynajmniej jedna musi brać udział w deszyfrowaniu, by proces ten zakończył się powodzeniem).

W drugiej części opisano też dwie sytuacje, w których można zastosować przedstawiony model. Zawarty tam został także przykład liczbowy, ilustrujący poprawność przedstawionych algorytmów.

Słowa kluczowe: kryptografia, deszyfrowanie progowe, RSA, bezpieczeństwo baz danych
Symbole UKD: 681.3.05

1. Wstęp

W wielu obecnych systemach wartość danych znajdujących się w bazie wielokrotnie przekracza wartość sprzętu, na którym ta baza się znajduje. Z tego względu stworzenie odpowiednich zabezpieczeń jest jednym z najważniejszych zadań dla projektantów baz danych.

Nowo powstające systemy baz danych stosują coraz więcej zabezpieczeń — hasła dostępowe, kontrola użytkownika, monitoring — mających na celu utrudnienie kradzieży informacji. Okazuje się jednak, że i takie zabezpieczenia mogą nie być wystarczające. W przypadku fizycznego zdobycia całej bazy (np. w przypadku przechwycenia kopii zapasowej) lub w przypadku, gdy napastnikiem jest osoba o uprawnieniach administratora, wymienione zabezpieczenia okazują się nieskuteczne. Z tego powodu coraz częściej stosuje się kolejne zabezpieczenie — szyfrowanie danych znajdujących się w bazie. Można je spotkać np. w Microsoft SQL Server 2005 (patrz [1, 11]). Jeśli dane w bazie są zaszyfrowane, to osoba nieznająca klucza deszyfrującego nie będzie w stanie ich odczytać.

Szyfrowanie może być przeprowadzone na trzech poziomach — na poziomie całych rekordów (wierszy), całych argumentów (kolumn) lub pojedynczych komórek. Opis kilku algorytmów szyfrowania baz danych można znaleźć w książce [6]. W naszej pracy przedstawiamy następny krok w dziedzinie zabezpieczania baz danych — szyfrowanie z użyciem technik kryptografii progowej.

Kevin Kenan w swojej książce [3] podaje, że znaczna część ataków i kradzieży jest dziełem napastników wewnętrznych — a więc ludzi posiadających hasło odszyfrowujące. Zastosowanie deszyfrowania progowego uniemożliwia atak pojedynczemu napastnikowi, gdyż wymaga współpracy przy odszyfrowywaniu kilku osób posiadających „fragmenty" klucza (zwane udziałami).

Opisany przez nas model jest bardzo elastyczny. Pozwala ustalać różne progi (czyli ilości udziałów potrzebnych do odszyfrowania) dla różnych danych (dane w bazie mają różną wartość i mogą wymagać różnego poziomu zabezpieczenia).

W artykule poza modelem podstawowym przedstawiamy kilka modyfikacji, pozwalających na dodatkowe zwiększenie bezpieczeństwa.

Pierwszą modyfikacją jest możliwość wskazania podgrupy (bądź podgrup) użytkowników i utworzenia swego rodzaju filtrów ograniczających dostęp do konkretnych danych użytkownikom nienależącym do wybranej podgrupy. Dodatkowo możliwe jest wprowadzenie deszyfrowania progowego w obrębie tej wskazanej podgrupy (tzn. takie zaszyfrowanie danych, by do ich odszyfrowania konieczna była współpraca określonej liczby osób należących do wskazanej podgrupy).

Kolejną modyfikacją jest możliwość tworzenia podgrupy osób „kluczowych" i takie zaszyfrowanie danych, by do ich odszyfrowania (w ramach ustalonego progu) **konieczny** był udział przynajmniej jednej osoby „kluczowej". Rozszerzeniem tej modyfikacji jest możliwość wskazania kilku podgrup i wymuszenie współpracy

przynajmniej jednej osoby z każdej podgrupy w celu odszyfrowania danych. Modyfikacje te mogą funkcjonować równocześnie na jednej bazie tzn. możemy do części danych zastosować model podstawowy, a do części model zmodyfikowany.

Ogromną zaletą opisanego modelu jest fakt, iż niezależnie od ilości progów i modyfikacji w ramach jednej bazy użytkownicy posługują się tylko jedną parą kluczy.

W rozdziale 6 przedstawiamy dwie sytuacje, w których zastosowanie opisanego przez nas modelu jest, naszym zdaniem, najlepszym rozwiązaniem. Model matematyczny zastosowany przez nas został opisany w pracy [2] i jest oparty na systemie RSA [9].

2. Podstawy matematyczne

2.1. Chińskie twierdzenie o resztach

Jeśli liczby całkowite dodatnie $n_1,n_2,\ldots,n_k$ są parami względnie pierwsze, zaś $a_1,a_2,\ldots,a_k$ są dowolnym liczbami całkowitymi, to istnieje liczba całkowita a spełniająca warunek: $a \equiv a_i \ mod \ n_i$ (dla $i = 1,2,\ldots k$).

Można ją wyliczyć ze wzoru $a = (\sum_{i=1}^{k} a_i y_i z_i) \bmod n$, gdzie:

$$n = \prod_{i=1}^{k} n_i,$$

$$z_j = \frac{n}{n_j} = (\prod_{i=1}^{j-1} n_i) \cdot (\prod_{i=j+1}^{k} n_i) \quad \text{oraz}$$

$$y_j = z_j^{-1} \bmod n_j.$$

2.2. Kryptosystem RSA

System kryptograficzny RSA jest najpowszechniej używanym kryptosystemem z kluczem publicznym. Umożliwia on zarówno szyfrowanie, jak i tworzenie podpisów cyfrowych, a jego bezpieczeństwo opiera się na problemie faktoryzacji liczb naturalnych.

Parametry kryptosystemu RSA:

Klucz publiczny to (e, N), a klucz prywatny to d gdzie:

$N = p \cdot q$ (p, q to liczby pierwsze),

$\varphi(N) = (p - 1) \cdot (q - 1)$,

e jest dowolną liczbą względnie pierwszą z $\varphi(N)$, a

d jest liczbą spełniającą warunek : $e \cdot d \equiv 1 \bmod \varphi(N)$.

Szyfrowanie wiadomości m polega na wyliczeniu $c = m^e \bmod N$, a deszyfrowanie szyfrogramu c to policzenie $m = c^d \bmod N$.

Uwaga: *Dowód poprawności systemu RSA opiera się na tzw. małym twierdzeniu Fermata:*

Jeśli p jest liczbą pierwszą oraz NWD$(m, p) = 1$, to $m^{p-1} \equiv 1 \bmod p$

Definicja: *Przez* λ *będziemy oznaczać funkcję lambda Carmichaela, zdefiniowaną następująco:*

$$\lambda(2) = 1, \quad \lambda(4) = 2, \quad \lambda(2^\alpha) = 2^{\alpha-2} \quad \text{dla} \quad \alpha \geq 3,$$

$\lambda(p^\alpha) = p^{\alpha-1}(p-1)$, gdzie p jest nieparzystą liczbą pierwszą oraz

$$\lambda(\prod_p (p^{\alpha_p})) = NWW(\lambda(p^{\alpha_p})).$$

Więcej informacji nt. kryptosystemu RSA można znaleźć w pracy [9] oraz w publikacjach: [4, 6, 10].

2.3. Deszyfrowanie ze zmiennym progiem

Ideą deszyfrowania progowego jest podział klucza deszyfrującego na części i wyposażenie użytkowników w „udziały", przy użyciu których będą mogli deszyfrować dane. Jednak w odróżnieniu od klasycznego deszyfrowania użytkownicy muszą współpracować, by odczytać dane. W trakcie tworzenia udziałów dla użytkowników ustala się próg — czyli minimalną ilość tych udziałów, która musi być użyta, by to deszyfrowanie się powiodło.

Największą słabością takiego podejścia jest konieczność stosowania tego samego progu dla wszystkich danych (a rzadko się zdarza, by wszystkie dane miały identyczny poziom ważności i wymagały takiego samego poziomu zabezpieczenia). Jakakolwiek zmiana progu wymaga zmiany wielomianu (którego stopień określa wartość progu), dzielącego odpowiedni sekret grupy, a co za tym idzie wymaga wygenerowania nowych udziałów dla użytkowników. Ostatnio powstało kilka prac oferujących trochę większą elastyczność progu (np. [5, 7, 8]), jednak przedstawione tam rozwiązania nie usuwają całkowicie tej wady.

Deszyfrowanie ze zmiennym progiem (opisane po raz pierwszy w pracy [2]) nie posiada tej słabości. Umożliwia ono ustalanie progu dla każdych danych osobno. Możliwe jest to dzięki odwróceniu procesu tworzenia udziałów — nie są one wynikiem „podzielenia" klucza deszyfrującego, to raczej klucz deszyfrujący jest tworzony (dla każdych danych oddzielnie) z wybranych udziałów. Dzięki takiemu podejściu użytkownicy korzystają tylko z jednej pary kluczy (będącej udziałami) niezależnie od ilości progów funkcjonujących w systemie.

3. Opis modelu

3.1. Oznaczenia

Niech $G = \{P_1,\ldots,P_n\}$ oznacza grupę użytkowników posiadających możliwość odszyfrowywania danych (posiadających udziały). Zakładamy, że członkowie tej grupy posiadają klucze RSA — $(d_i, (e_i, N_i))$, spełniające warunek:

$$N_1 < N_2 < N_3 < \ldots < N_n. \tag{1}$$

Warunek ten wprowadzamy tylko ze względu na uproszczenie oznaczeń, nie ma on wpływu na działanie algorytmów.

W modelu będziemy przeprowadzać szyfrowanie na poziomie pojedynczych komórek. Podyktowane jest to ograniczeniem (opisanym poniżej) nałożonym na wielkość danych, które mogą być szyfrowane.

Zakładamy, że $M(i)$ oznacza komórkę o numerze i, a M_i oznacza dane znajdujące się w tej komórce. Dla uproszczenia przyjmiemy, że do ich odszyfrowania potrzeba i udziałów, należących do użytkowników. W związku z wykorzystaniem operacji modulo, dane w komórce M_j muszą spełniać warunek: $M_j < \prod_{i=1}^{j} N_i$ (by zapewnić jednoznaczność wyniku przy deszyfrowaniu).

(Nierówność (1) zapewnia, że jest to najmniejszy spośród takich iloczynów).

3.2. Combiner

W celu zapewnienia prawidłowego funkcjonowania systemu, wprowadzamy dodatkowego uczestnika — combinera, pełniącego kilka funkcji. Po pierwsze, będzie on pośrednikiem w przesyłaniu danych. Użytkownicy będą przesyłać zapytania do combinera, a on będzie pobierał odpowiednie dane z bazy.

Jednak głównym zadaniem combinera będą operacje kryptograficzne — będzie on szyfrował zapisywane przez użytkowników dane, a także będzie je odszyfrowywał przy odczycie (współpracując z użytkownikami posiadającymi udziały). Dokładny zakres zadań combinera będzie opisany przy okazji opisu poszczególnych algorytmów.

3.3. Komunikacja

W modelu użytkownicy nie posiadają bezpośredniego połączenia z bazą danych — wszelkie zapytania kierowane są do combinera, który pobiera (lub zapisuje) odpowiednie dane z bazy.

Celem naszej pracy jest pokazanie zabezpieczenia danych znajdujących się w bazie, a nie zabezpieczenia ich przesyłania i dlatego zakładamy, że wszystkie

połączenia są nawiązywane za pomocą bezpiecznych kanałów (aczkolwiek dane przepływające pomiędzy combinerem a bazą danych są zaszyfrowane, więc to łącze nie musi być dodatkowo zabezpieczane).

4. Schemat podstawowy

4.1. Szyfrowanie danych w komórce $M(1)$

Dane znajdujące się w tej komórce może odszyfrować pojedyncza osoba z grupy G. Z tego powodu musi być spełniony warunek $M_1 < N_1$ (po raz kolejny wykorzystujemy nierówność (1)).

Proces szyfrowania danych w tej komórce przebiega następująco [2]:

— Użytkownik przesyła dane M_1 do combinera celem ich zaszyfrowania i zapisania do bazy.
— Combiner wylicza $C_1 = M_1^{\,E} \bmod N$, gdzie $E = \prod_{i=1}^{n} e_i$ a $N = \prod_{i=1}^{n} N_i$.
— Combiner zapisuje (E, C_1) do komórki $M(1)$.

Uwaga: *Parametr E będzie wspólny dla wielu komórek. By zmniejszyć ilość danych przechowywanych w komórkach można przechowywać ten parametr poza nimi (np. utworzyć osobną tabelę na parametry, a w komórkach podawać tylko wskaźnik do miejsca przechowywania odpowiedniej wartości).*

4.2. Szyfrowanie danych w komórce $M(t)$ ($t \geq 2$)

Dane, które mają być zapisane w tej komórce muszą spełniać warunek: $M_t < \prod_{i=1}^{t} N_i$ (nierówność (1) pozwala uprościć zapis).

By zaszyfrować dane M_t combiner:

— wylicza $m_i = M_t \bmod N_i$ dla $i = 1,2,\ldots,n$,
— wybiera losową nieparzystą liczbę k oraz liczbę pierwszą p: $k < p < N_1$,
— tworzy wielomian: $f(x) = k + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1} \in GF(p)$,
— wylicza $c_i = f(x_i)$ dla $i = 1,2,\ldots,n$ (x_i są znane) i usuwa k,
— wylicza $c_i^{\,e_i} \bmod N_i$ oraz $m_i^{\,k} \bmod N_i$ dla $i = 1,2,\ldots,n$,
— korzystając z CRT (chińskiego twierdzenia o resztach) wylicza C_1 i C_2 takie, że:
— $c_i^{\,e_i} \equiv C_1 \bmod N_i$ dla $i = 1,2,\ldots,n$ oraz
— $m_i^{\,k} \equiv C_2 \bmod N_i$ dla $i = 1,2,\ldots,n$
— zapisuje (p, t, C_1, C_2) do komórki $M(t)$.

Uwaga: *Parametr C_1 będzie wspólny dla komórek szyfrowanych przy pomocy tego samego wielomianu. Można go przechowywać w osobnej tabeli, a w komórkach umieszczać tylko informacje o miejscu jego przechowywania.*

4.3. Deszyfrowanie komórki $M(1)$

Dowolny użytkownik z grupy G może w pojedynkę odszyfrować dane z tej komórki. Proces ten przebiega następująco [2]:

— użytkownik P_i przesyła do combinera zapytanie o dane z komórki $M(1)$,
— combiner pobiera $(E, C1)$ z komórki i przesyła je użytkownikowi,
— użytkownik P_i wylicza $C_1^{D_i} = M_1^{(E \cdot D_i)} = M_1 \bmod N_i$, gdzie $D_i = E^{-1} \bmod \lambda(N_i)$.

4.4. Deszyfrowanie komórki $M(t)$ ($t \geq 2$)

Niech $B = \{P_1, P_2, \ldots P_t\} \subset G$ będzie podgrupą zawierającą użytkowników, którzy chcą odszyfrować dane M_t. Po przesłaniu do combinera zapytania o te dane:

— combiner pobiera (p, t, C_1, C_2) z komórki i przesyła je do użytkowników,
— użytkownik $P_i \in B$ $(i = 1,2,\ldots,t)$ wylicza (a następnie zwraca do combinera) $c_i = (c_i^{e_i})^{d_i} \bmod N_i = C_1^{d_i} \bmod N_i$,
— combiner po otrzymaniu t wartości c_i wylicza k (rozwiązując układ t równań z t z niewiadomymi) i przesyła je do użytkowników,
— użytkownicy wyliczają $l = k^{-1} \bmod \lambda(N_i)$, a następnie $m_i = (m_i^k)^l \bmod N_i = C_2^l \bmod N_i$ (które przesyłają do combinera),
— combiner, korzystając z CRT, wylicza M_t z m_i i przesyła odszyfrowane dane do użytkowników.

5. Modyfikacje schematu

W modelu podstawowym wszyscy użytkownicy są traktowani jednakowo — mają dostęp do wszystkich danych (mogą je z powodzeniem deszyfrować w pojedynkę lub grupowo). Opisane niżej modyfikacje pozwalają ograniczyć dostęp do pewnych danych wybranym użytkownikom.

Należy zaznaczyć, że uwagi z punktu 4, dotyczące zmniejszenia ilości danych zapisywanych w komórkach, mają zastosowanie także w opisywanych niżej modyfikacjach.

5.1. Filtry

Ta modyfikacja polega na wprowadzeniu swoistych filtrów ograniczających dostęp do danych konkretnym osobom (a dokładniej umożliwiające wskazanie osób, które mogą deszyfrować te dane). Zaletą tych filtrów jest fakt, iż mogą dotyczyć nie tylko kolumn czy wierszy, ale także pojedynczych komórek.

Niech $A=\{P_{i_1},P_{i_2},...,P_{i_l}\}\subset G$ będzie podgrupą osób uprawnionych do deszyfrowania wybranych danych.

Szyfrując dane M_1 (korzystając z algorytmu opisanego w 4.1.) combiner korzysta z następujących wartości E i N:

$$E=e_{i_1}\cdot e_{i_2}\cdot...\cdot e_{i_l} \quad \text{oraz} \quad N=N_{i_1}\cdot N_{i_2}\cdot...\cdot N_{i_l}$$

Po takim zabiegu odszyfrowanie danych M_1 będzie mogła przeprowadzić tylko osoba należąca do grupy A.

Wprowadzenie filtrów przy szyfrowaniu danych M_t (dla $t \geq 2$) jest równie nieskomplikowane. Jeśli w pierwszym kroku algorytmu opisanego w punkcie 4.2. ograniczymy wartości i do zbioru $\{i_1, i_2, \ldots i_l\}$ (zamiast $i = 1, 2, \ldots, n$), to odszyfrowanie M_t będzie wymagać udziału t osób należących do podgrupy A (przy założeniu, że $t \leq l$).

5.2. Szyfrowanie ze wskazaniem koniecznych odbiorców

Opisane wyżej filtry pozwalają wskazać listę osób, które nie mogą brać udziału w deszyfrowaniu danych, ta modyfikacja pozwala wskazać osoby, które muszą brać w nim udział. A dokładniej pozwalają wskazać grupę osób i tak zaszyfrować dane by do ich odszyfrowania konieczny był udział przynajmniej jednego reprezentanta tej wskazanej grupy (zwanej „kluczową").

Niech $A=\{P_{i_1},P_{i_2},P_{i_3}\}$ będzie podgrupą kluczową. Proces szyfrowania przebiega następująco:

- combiner dzieli dane M_t na dwie losowe części M_t' i M_t'' w taki sposób, że $M_t=M_t'\oplus M_t''$ (gdzie M_t, M_t', M_t'' są ciągami bitów, a $\oplus$ to sumowanie modulo 2),
- następnie combiner szyfruje M_t' algorytmem opisanym w punkcie 4.1. zmodyfikowanym dla potrzeb grupy A (tzn. $E=e_{i_1}\cdot e_{i_2}\cdot e_{i_3}$ oraz $N=N_{i_1}\cdot N_{i_2}\cdot N_{i_3}$). Otrzymany szyfrogram to (E,C_1'),
- część M_t'' jest szyfrowana przy pomocy algorytmu opisanego w punkcie 4.2., a następnie całość tzn. (E,C_1',p,t,C_1'',C_2'') jest zapisywana do bazy.

Do odczytania M_t konieczne jest odzyskanie zarówno M_t' jak i M_t'' (odzyskanie tylko jednej z tych wartości nie pozwala na uzyskanie informacji na temat M_t). Deszyfrowania M_t' może dokonać jedynie osoba należąca do grupy A (wystarczy jedna, postępuje ona zgodnie z algorytmem opisanym w punkcie 4.3. i przesyła otrzymany wynik do combinera).

Odzyskiwanie M_t'' przebiega dokładnie według metody opisanej w punkcie 4.4. — do jej przeprowadzenia potrzeba t osób (członek grupy A może brać udział w deszyfrowaniu obu fragmentów). Po odszyfrowaniu obu fragmentów combiner może wyliczyć M_t i przesłać je do użytkowników.

6. Zastosowania

6.1. Zabezpieczanie danych o wysokim stopniu tajności

Dane oznaczone klauzulą „tajne" lub „ściśle tajne" można spotkać np. w bazach danych policji lub wojska (mogą się tam znajdować np. dane agentów wywiadu, działających poza granicami kraju). Kradzież takich danych (ich ujawnienie) może mieć katastrofalne skutki — ujawnienie nazwisk agentów i miejsc ich działania nie tylko zagraża ich życiu, ale może także spowodować polityczne trzęsienie ziemi.

Oczywiste jest, że takie dane muszą być zabezpieczone w najlepszy dostępny sposób. Stosując nasz model możemy sprawić, by do odszyfrowania krytycznych danych potrzebna była współpraca kilku lub kilkunastu osób (a więc także ich zgoda na odczytanie tych informacji). Metoda ta uniemożliwia atak pojedynczemu napastnikowi.

Opisana w punkcie 5.2 modyfikacja algorytmu wprowadza dodatkowe zabezpieczenie. Dzięki niemu można wskazać grupę osób (np. wyższych dowódców) i tak zaszyfrować dane, by do ich odszyfrowania konieczna była współpraca (a więc i zgoda) t osób, w tym przynajmniej jednego wyższego dowódcy.

6.2. Bezpieczne udostępnianie bazy danych

W tym przykładzie mamy do czynienia z sytuacją, gdy jedna instytucja chce mieć dostęp do bazy danych, należącej do innej instytucji (oczywiście właściciel bazy chce w pełni kontrolować i ewentualnie blokować operacje wykonywane przez „gościa" na danych).

Stosując opisany przez nas model można przekazać „gościowi" n udziałów przy jednoczesnym ustaleniu progu na poziomie $n + k (k > 0)$. To sprawia, że za każdym razem gdy „gość" będzie chciał odczytać dane, będzie potrzebował udziału przynajmniej k osób reprezentujących właściciela bazy. W takiej sytuacji „właściciel" ma lepszą kontrolę nad tym kto, kiedy i co czyta z bazy. Może także w prosty sposób zabronić dostępu do konkretnych danych (nie udostępniając potrzebnej ilości udziałów).

7. Podsumowanie

W naszej pracy przedstawiliśmy nowy sposób zabezpieczania baz danych — szyfrowanie z użyciem technik kryptografii progowej.

Model korzysta z algorytmu, którego twórcami są Ghodosi, Pieprzyk i Safavi-Navi (opisanego w artykule [2]), opartego na chińskim twierdzeniu o resztach i kryptosystemie RSA. Zaletą modelu jest możliwość ustalania różnych progów dla różnych danych bez konieczności przydzielania użytkownikom dodatkowych udziałów.

W artykule skupiliśmy się bardziej na przedstawieniu modelu oraz dowodzie jego poprawności i przydatności w realnych sytuacjach.

Opisaliśmy zarówno model podstawowy, jak i kilka modyfikacji zwiększających bezpieczeństwo danych oraz sprawiających, że system jest jeszcze bardziej elastyczny. Należy zaznaczyć, że opisane modyfikacje także nie wymagają tworzenia nowych (ani zmiany już posiadanych) udziałów dla użytkowników. Takie podejście sprawia, że nasz model oferuje proste rozwiązania w przypadkach, gdy zwykłe (stosowane obecnie) zabezpieczenia nie są do końca skuteczne.

W rozdziale 6 opisaliśmy dwa przykłady, w których zaproponowany przez nas model jest, naszym zdaniem, najlepszym rozwiązaniem.

8. Przykład

- Zakładamy, że grupa składa się z 3 użytkowników $G = \{P_1, P_2, P_3\}$
- Parametry związane z kluczami RSA należącymi do użytkowników:

$N_1 = 23 \cdot 167 = 3841$ $\phi(N_1) = 3652$ $\lambda(N_1) = 1826$ $e_1 = 17$ $d_1 = 1289$

$N_2 = 47 \cdot 107 = 5029$ $\phi(N_2) = 4876$ $\lambda\,(N_2) = 2438$ $e_2 = 13$ $d_2 = 4501$

$N_3 = 59 \cdot 83 = 4897$ $\phi(N_3) = 4756$ $\lambda\,(N_3) = 2378$ $e_3 = 11$ $d_3 = 3459$

- Niezaszyfrowana baza danych (dla uproszczenia — jedna tabela)

25	71	43
16	65	58
45	17	43

Przyjmujemy, że pierwszy wiersz zawiera tylko komórki $M(1)$, drugi tylko $M(2)$, a trzeci wiersz tylko komórki $M(3)$.

- Combiner szyfruje dane z pierwszego wiersza wyliczając:

$$E = e_1 \cdot e_2 \cdot e_3 = 17 \cdot 13 \cdot 11 = 2431$$

$$N = N_1 \cdot N_2 \cdot N_3 = 3841 \cdot 5029 \cdot 4897 = 94592356933$$

A następnie:

$$C_1(25) = 25^{2431} \bmod 94592356933 = 24397229255$$

$$C_1(71) = 71^{2431} \bmod 94592356933 = 48102756992$$

$$C_1(43) = 43^{2431} \bmod 94592356933 = 67954064948$$

- Parametry (wspólne dla całego wiersza) wykorzystywane do szyfrowania drugiego wiersza:

$k = 35$ $p = 37$ $f(x) = 35 + 11x$

$c_1 = f(1) = 46$ $c_2 = f(2) = 57$ $c_3 = f(3) = 68$

$$f_1 = c_1^{e_1} \bmod N_1 = 46^{17} \bmod 3841 = 2806$$
$$f_2 = c_2^{e_2} \bmod N_2 = 57^{13} \bmod 5029 = 1871$$
$$f_3 = c_3^{e_3} \bmod N_3 = 68^{11} \bmod 4897 = 3024$$

- Parametry różne dla poszczególnych komórek (16 65 58):

$m_i = 16$ $m_1^k \bmod N_1 = 3039$ $m_2^k \bmod N_2 = 4610$ $m_3^k \bmod N_3 = 566$

$m_i = 65$ $m_1^k \bmod N_1 = 2928$ $m_2^k \bmod N_2 = 4627$ $m_3^k \bmod N_3 = 1488$

$m_i = 58$ $m_1^k \bmod N_1 = 3640$ $m_2^k \bmod N_2 = 2297$ $m_3^k \bmod N_3 = 648$

- Korzystając z CRT, wyliczamy dla każdej komórki C_1 (wspólne) oraz C_2:

$$C_1 = \sum_{i=1}^{3} f_i \cdot N^i \cdot ((N^i)^{-1} \bmod N_i), \quad \text{gdzie} \quad N^i = \frac{N}{N_i}$$
$$C_2(m_i) = \sum_{i=1}^{3} m_i^k \cdot N^i \cdot ((N^i)^{-1} \bmod N_i)$$

$$C_1 = 2806 \cdot 24627013 \cdot 174 + 1871 \cdot 18809377 \cdot 2775 + \\ + 3024 \cdot 19316389 \cdot 1973 = 224931123096525$$

$$C_2(16) = 275217036604270$$

$$C_2(65) = 310767281963397$$

$$C_2(58) = 160188106194711$$

- Szyfrowanie danych w wierszu trzecim (45 17 43).

Parametry wspólne dla całego wiersza:

$k = 39$ $p = 43$ $f(x) = 39 + 13x + 11x^2$

$c_1 = f(1) = 63$ $c_2 = f(2) = 109$ $c_3 = f(3) = 177$

$$f_1 = c_1^{e_1} \bmod N_1 = 63^{17} \bmod 3841 = 3461$$
$$f_2 = c_2^{e_2} \bmod N_2 = 109^{13} \bmod 5029 = 702$$
$$f_3 = c_3^{e_3} \bmod N_3 = 177^{11} \bmod 4897 = 3599$$

Parametry różne dla poszczególnych komórek:

$m_i = 45$ $\quad m_1^{\,k} \bmod N_1 = 3311$ $\quad m_2^{\,k} \bmod N_2 = 3930$ $\quad m_3^{\,k} \bmod N_3 = 1167$

$m_i = 17$ $\quad m_1^{\,k} \bmod N_1 = 1322$ $\quad m_2^{\,k} \bmod N_2 = 862$ $\quad m_3^{\,k} \bmod N_3 = 4260$

$m_i = 43$ $\quad m_1^{\,k} \bmod N_1 = 2284$ $\quad m_2^{\,k} \bmod N_2 = 4376$ $\quad m_3^{\,k} \bmod N_3 = 101$

- Korzystając z CRT, wyliczamy C_1 oraz C_2 (analogicznie jak w przypadku szyfrowania wiersza drugiego):

$C_1 = 188634675425335$

$C_2(45) = 263794142010231$

$C_2(17) = 213011736016434$

$C_2(43) = 242046192445405$

- Po zaszyfrowaniu baza danych wygląda następująco:

$(T_1, 24397229255)$ $\quad (T_1, 48102756992)$ $\quad (T_1, 67954064948)$

$(T_2, 275217036604270)$ $\quad (T_2, 310767281963397)$ $\quad (T_2, 160188106194711)$

$(T_3, 263794142010231)$ $\quad (T_3, 213011736016434)$ $\quad (T_3, 242046192445405)$

Tabela T (zawiera parametry wspólne dla kilku komórek):

$E = 2431$

$p = 37, t = 2, C_1 = 224931123096525$

$p = 43, t = 3, C_1 = 188634675425335$

- Deszyfrowanie komórki pierwszej komórki z pierwszego wiersza przez P_3:

$D_3 = 2431^{-1} \bmod 2378 = 1705$

$C_1^{\,D_3} \bmod N_3 = 24397229255^{1705} \bmod 4897 = 25$

- Deszyfrowanie trzeciej komórki drugiego wiersza przez P_1 i P_2:

$(P_1) 224931123096525^{1289} \bmod 3841 = 46$ wynik wysyła do combinera

$(P_2) 224931123096525^{4501} \bmod 5029 = 57$ wynik wysyła do combinera

Combiner rozwiązuje układ równań: $\begin{cases} f(1) = k + a_1 = 46 \\ f(2) = k + 2a_1 = 57 \end{cases}$

i otrzymując $k = 35$, które przesyła do P_1 i P_2.

P_1 i P_2 liczą $l_i = k^{-1} \bmod \lambda(N_i)$ *oraz* $m_i = C_2^{\,l_i} \bmod N_i$
(m_i wysyłają do combinera):

$$l_1 = 35^{-1} \bmod 1826 = 1513 \quad m_1 = 160188106194711^{1513} \bmod 3841 = 58$$

$$l_2 = 35^{-1} \bmod 2438 = 209 \quad m_2 = 160188106194711^{209} \bmod 5029 = 58$$

Combiner (stosując CRT) wylicza:

$$M_2 = [(58 \cdot 5029 \cdot (5029^{-1} \bmod 3841) + 58 \cdot 3841 \cdot (3841^{-1} \bmod 5029)]$$

$$\bmod(3841 \cdot 5029) = 1120350620 \bmod 19316389 = 58$$

i przesyła odszyfrowaną zawartość komórki do P_1 i P_2.

Dalsze problemy:

1. Zbadać złożoność obliczeniową algorytmów w modelu podstawowym.
2. Zbadać ograniczenie wielkości bazy danych, do której można zastosować model.
3. Opisać ochronę serwera, na którym znajduje się oprogramowanie realizujące funkcje „Combinera".
4. Zbadać formalnie poprawność i bezpieczeństwo modelu.

Artykuł wpłynął do redakcji 21.04.2008 r. Zweryfikowaną wersję po recenzji otrzymano we wrześniu 2008 r.

LITERATURA

[1] E. L. Brown, *SQL Server 2005 distilled*, Microsoft Windows Server Series, 2006.

[2] H. Ghodosi, J. Pieprzyk, R. Safavi-Navi, *Dynamic threshold cryptosystems: A new scheme in group oriented cryptography*, Proceedings of PRAGOCRYPT'96, CTU, 1996, 370-379.

[3] K. Kenan, *Cryptography in the database. The last line of defense*, Symantec Press, 2005.

[4] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of applied cryptography*, CRC, Boca Rato, FL, 1997.

[5] B. Nakielski, J. Pomykała, A. Pomykała, M*ulti-threshold signature*, JTiT, 1, 2008, 51-55.

[6] J. Pieprzyk, T. Hardjono, J. Seberry, *Fundamentals of computer security*, Springer-Verlag, Berlin Heidelberg, 2003.

[7] J. Pomykała, T. Warchoł, *Threshold signatures in dynamic groups*, Proceedings of the Future Generation Communication and Networking, Jeju Island, IEEE Computer Society, December 6-8, 2007, 32-37.

[8] J. Pomykała, T. Warchoł, *Dynamic multithreshold signature without the Trusted Dealer*, IJMUE, vol. 3, no. 3, July 2008, 31-41.

[9] R. L. Rivest, A. Shamir, L. M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, vol. 21, no. 2, 1978, 120-126.

[10] B. Schneier, *Applied cryptography*, Wiley, New York, 1996.

[11] http://wss.pl/articles/6042.aspx

B. NAKIELSKI, J. POMYKAŁA, J. A. POMYKAŁA

Securing database with dynamic threshold decryption

Abstract. The works presents a new way of securing the database — dynamic threshold decryption of the encrypted data in the database. It helps to minimize the risk created by the "inside attackers" (e.g. corrupted employees having an access to the database) by requiring of users cooperation in the process of decryption (reading) of data from the database. The currently used database systems are not equipped with this kind of protection. Any single corrupted user with a password can decrypt and steal critical data from the system.

In the presented model, the users are equipped merely with the suitable shares. Decryption of data requires the specified number of these shares, so the group of the corrupted users must be large enough to steal the crucial data.

This minimal number of shares required for decryption is called "the threshold level".

In the "traditional" threshold cryptosystems the threshold level is fixed in advance and cannot be changed (unless the process of the generation of the new shares is executed). Therefore all the messages require the same number of shares needed for the decryption process.

The presented model offers the full flexibility of the threshold level. It may vary together with the data to be encrypted. It is important since some data may require higher level of secrecy (protection) than the others.

First part of the article gives the mathematical background — it contains the basic information about the RSA cryptosystem, the Chinese reminder theorem and dynamic threshold decryption. It presents the general system model (participants, initial conditions and the algorithms used).

Second part describes the possible modifications which make the presented model more secure and practical. It contains also the mathematical example simulating the work (and correctness of the system).

Keywords: cryptography, dynamic threshold decryption, RSA, database security

Universal Decimal Classification: 681.3.05