Biuletyn WAT
Vol. LXII, Nr 4, 2013

Projektowanie ataku z kompromisem czasu i pamięci na przykładzie szyfru A5/1

KRZYSZTOF KANCIAK, MICHAŁ MISZTAL

Wojskowa Akademia Techniczna, Wydział Cybernetyki, Instytut Matematyki i Kryptologii,
01-980 Warszawa, ul. gen. S. Kaliskiego 2, kkanciak@wat.edu.pl, mmisztal@wat.edu.pl

Streszczenie. W artykule przeanalizowano różne warianty ataku brutalnego z fazą obliczeń wstępnych (lub inaczej ataku kompromisu czasu–pamięci) na przykładzie algorytmu strumieniowego A5/1. Omówiono również teoretyczną skuteczność różnych wariantów ataku, przedstawiono wyniki optymalizacji parametrów ataku, sformułowano wnioski dotyczące projektowania ataku kompromisu czasu–pamięci. Przedstawione zostały także otrzymane wyniki kryptoanalizy algorytmu A5/1 w zestawieniu z oczekiwanymi wartościami teoretycznymi.
Słowa kluczowe: kryptologia, kryptoanaliza, szyfr strumieniowy, atak z fazą obliczeń wstępnych, kompromis czasu–pamięci

1. Wstęp

Atak kompromisu czasu–pamięci został zaproponowany przez Martina Hellmana w 1980 roku [1]. Polega on na podzieleniu obliczeń koniecznych do odtworzenia szukanego tajnego klucza komunikacji na dwa etapy. Pierwszy etap to długa, często wielomiesięczna, ale wykonywana tylko raz *faza obliczeń wstępnych*. Drugim etapem jest *faza ataku* w *czasie rzeczywistym*, która wymaga zapisu przechwyconej, zaszyfrowanej komunikacji oraz rezultatu fazy obliczeń wstępnych. Skuteczność i praktyczna realizowalność ataku polega na takim zaplanowaniu obu faz ataku, aby nie przekroczyły one dostępnych zasobów, tj.:

— żeby złożoność czasowa i pamięciowa były akceptowalne na etapie projektu, tj. czas trwania fazy obliczeń wstępnych oraz jej rezultat spełniały ograniczenia czasowe i pamięciowe ataku,

— żeby faza ataku w czasie rzeczywistym przebiegała w bardzo ograniczonym czasie (minuty) z wykorzystaniem jak najmniejszej liczby przechwyconych szyfrogramów.

Im bardziej złożona obliczeniowo jest faza obliczeń wstępnych, tym większe są koszty jej przeprowadzenia i pamięć konieczna do przechowania jej rezultatu, ale tym mniejsza jest złożoność obliczeniowa fazy ataku w czasie rzeczywistym. Znalezienie optymalnego kompromisu konieczne jest na etapie projektowania całego ataku z fazą obliczeń wstępnych.

W pracy przedstawiono poszukiwanie takiego kompromisu na przykładzie ataku na *szyfr strumieniowy A5/1*, który jest standardem szyfrowania w systemach GSM na całym świecie. Jednak ten typ ataku może być zastosowany do każdego szyfru. Jego złożoność zależy tylko od długości klucza i/lub długości strumienia.

Praca w części wprowadzającej zawiera krótki opis algorytmu, na przykładzie którego prowadzone będą rozważania, oraz szczegółowy opis koncepcji ataku wraz z jego modyfikacjami. Następnie sformułowano metodę projektowania opisanego typu ataków oraz przykład wykorzystania tej metody. Na koniec podsumowane zostały otrzymane wyniki i wnioski z przeprowadzenia ataku.

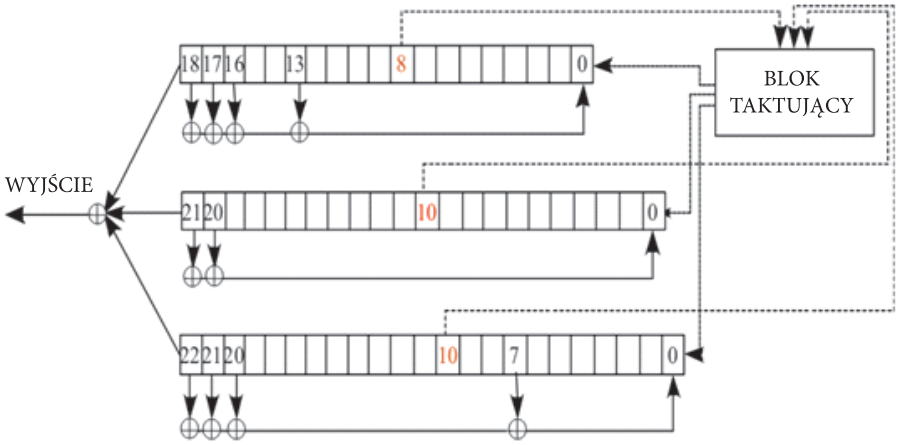
2. Algorytm A5/1

Alorytm A5/1 jest szyfrem strumieniowym. Szyfrowanie wykorzystuje operację XOR, tzn. *i*-ty bit szyfrogramu *C* powstaje przez dodanie modulo 2 *i*-tego bitu tekstu jawnego *M* z *i*-tym bitem strumienia klucza *S*: $C_i = M_i \oplus S_i$, z kolei deszyfrowanie odbywa się w identyczny sposób, tzn. strumień klucza *S* dodany modulo 2 do szyfrogramu *C* daje wiadomość *M*: $S_i \oplus C_i = S_i \oplus S_i \oplus M_i = M_i$. Algorytm A5/1 zbudowany jest na trzech rejestrach *LFSR* (ang. *Linear Feedback Shift Register*): *R*1, *R*2 i *R*3, których długości wynoszą odpowiednio: 19, 22 i 23 bity. Budowę algorytmu opisuje rysunek 1. Wielomiany połączeń rejestrów to odpowiednio: $R1: x^{19} \oplus x^5 \oplus x^2 \oplus x \oplus 1$; $R2: x^{22} \oplus x \oplus 1$; *R*3: $x^{23} \oplus x^{15} \oplus x^2 \oplus x \oplus 1$. Są to wielomiany nieredukowalne i dzięki temu długości okresów rejestrów są maksymalne i wynoszą odpowiednio $2^{19}-1; 2^{22}-1; 2^{23}-1$.

Każdy rejestr mniej więcej w środku ma swój tzw. *bit taktujący*, który jest odpowiedzialny za nieregularne taktowanie rejestrów. W rejestrze *R*1 jest to *R*1 [8], w *R*2 to *R*2 [10], a w *R*3 [10]. W każdym kolejnym cyklu działania A5/1 liczona jest funkcja większości tych trzech bitów, a potem taktowane są te rejestry, których bity taktujące są zgodne z obliczoną funkcją większości (w każdym cyklu taktowane są dwa lub trzy rejestry).

Jeżeli $i \in \{1, 2, 3\}$ to numer rejestru, a klucz $K_C \in \{0, 1\}^{64}$ oraz *numer ramki* to $F_{Num} \in \{0, 1\}^{22}$, to inicjacja kluczem polega na wyzerowaniu wszystkich rejestrów, następnie dla 64 taktów (bez reguły większości, tzn. wszystkie rejestry są taktowane)

następuje przypisanie $R_i[0] = R_i[0] \oplus K_C[n],\ n = 0 \ldots 63$. Identycznie wprowadzany jest numer ramki $R_i[0] = R_i[0] \oplus F_N[m],\ m = 0 \ldots 21$. Generowanie strumienia klucza następuje po wprowadzeniu klucza i numeru ramki oraz 100 taktach zgodnych z regułą taktowania, których wynik jest ignorowany. Generowanie strumienia trwa 228 taktów, a bit wyjścia uzyskiwany jest jako XOR najbardziej znaczących bitów rejestrów (rys. 1).



Rys. 1. Schemat blokowy szyfru A5/1

Dla opisywanej metody ataku budowa algorytmu ma ograniczone znaczenie, jednak należy zwrócić uwagę, że stan wewnętrzny rejestrów po inicjacji jest liniową funkcją klucza i znanego publicznie numeru ramki. Choć algorytm A5/1 jest szyfrem strumieniowym, na potrzeby kompromisu czasu–pamięci będziemy go widzieć jak szyfr, który z 64-bitowego bloku tekstu jawnego P na wejściu produkuje 64-bitowy blok szyfrogramu C z wykorzystaniem 64-bitowego klucza K: $C = S_K(P)$.

Scenariusz ataku zakłada posiadanie szyfrogramu i odpowiadającego mu tekstu jawnego (atak ze znanym tekstem jawnym). W przypadku systemu GSM pozyskanie znanych tekstów jawnych jest stosunkowo łatwe, stąd atak ma charakter zupełnie praktyczny.

3. Opis idei ataku kompromisu czasu–pamięci

3.1. Pierwotna koncepcja Hellmana w kontekście algorytmu A5/1

Faza obliczeń wstępnych. Niech P_0 oznacza wybrany *blok wejściowy* oraz $f(K) = R\left[S_K(P_0)\right]$, gdzie R jest *funkcją redukcji*, tj. dowolnym przekształceniem szyfrogramu $S_K(P_0)$ zachowującym zgodność ze strukturą tekstu jawnego (długość

64 bitów). W naszym przypadku funkcja redukcji będzie miała postać sumy modulo 2 z 64-bitową stałą. Faza obliczeń wstępnych polega na jednorazowym wybraniu *punktów startowych* $SP_1, SP_2, .., SP_m$ dla $1 \le i \le m$, które będą pierwszymi elementami wierszy macierzy, tzn. $X_{i,1} = SP_i$, oraz obliczeniu $X_{ij} = f\left(X_{i,j-1}\right), 1 \le j \le t$. Wybór parametrów m, t należy do atakującego i zostanie omówiony dalej. Otrzymujemy w ten sposób macierz:

$$SP_1 = X_{1,1} \xrightarrow{f} X_{1,2} \xrightarrow{f} X_{1,3} \xrightarrow{f} \ldots \xrightarrow{f} X_{1,t} = EP_1,$$

$$SP_2 = X_{2,1} \xrightarrow{f} X_{2,2} \xrightarrow{f} X_{2,3} \xrightarrow{f} \ldots \xrightarrow{f} X_{2,t} = EP_2,$$

$$\ldots$$

$$SP_m = X_{m,1} \xrightarrow{f} X_{m,2} \xrightarrow{f} X_{m,3} \xrightarrow{f} \ldots \xrightarrow{f} X_{m,t} = EP_m.$$

Wiersz takiej macierzy nazywa się łańcuchem, a ostatni element i-tego łańcucha *punktem końcowym*, oznacza się go przez $EP_i = f^t(SP_i)$. Rezultatem fazy obliczeń wstępnych jest *tablica* $\{SP_i, EP_i\}_{i=1}^{m}$ posortowana rosnąco po punktach końcowych (wszystkie punkty pośrednie są pominięte).

Faza ataku w czasie rzeczywistym. Niech $C_0 = S_K(P_0)$ będzie przechwyconym blokiem szyfrogramu dla tekstu jawnego P_0 oraz nieznanego, szukanego klucza K. Atakujący oblicza $Y_1 = R(C_0) = f(K)$ i sprawdza, czy Y_1 jest punktem końcowym (w jednym „kroku"). Jeżeli tak, tzn. $Y_1 = EP_i$, to spodziewamy się, że $K = X_{i,\,t-1}$. Atakujący „odtwarza" wszystkie stany łańcucha, mając SP_i, i sprawdza, czy $X_{i,\,t-1}$ deszyfruje C_0 do P_0. Jeżeli tak, to szukany klucz został znaleziony. Jeżeli nie, czyli $K \neq X_{i,t-1}$, to znaczy, że nastąpił tzw. *fałszywy alarm*, czyli sytuacja, w której EP_i jest następnikiem innego poprzednika. Jeżeli natomiast Y_1 nie jest punktem końcowym lub zdarzył się fałszywy alarm, atakujący oblicza $Y_2 = f(Y_1)$ i sprawdza, czy Y_2 jest punktem końcowym, tzn. powtarza opisaną wyżej procedurę, obliczając kolejne wartości $Y_t = f(Y_{t-1})$. Jeżeli wszystkie elementy mt macierzy w kolumnach od 0 do $t-1$ są różne, to prawdopodobieństwo sukcesu rozumianego przez odnalezienie szukanego klucza w wygenerowanej macierzy wyraża się wzorem:

$$P(S) = \frac{mt}{N}, \tag{3.1.1}$$

gdzie N to przestrzeń klucza, co w przypadku A5/1 oznacza przestrzeń stanów wewnętrznych szyfru.

W praktyce w tak zbudowanej tablicy elementy będą się powtarzać (tj. będą występować złączenia łańcuchów), a to zmniejsza liczbę odrębnych kluczy, czyli teoretyczne zmniejsza „pokrycie" przestrzeni klucza. Im większa jest tablica, tym większe prawdopodobieństwo, że nowy łańcuch złączy się z którymś z poprzednich.

Prawdopodobieństwo, że kolejny wygenerowany punkt $X_{i,j}$ jest nowy, tzn. nie powoduje złączenia, można ograniczyć przez:

$$P_{KNew} \geq \left(1 - \frac{it}{N}\right)^{j}, \tag{3.1.2}$$

gdzie i oznacza liczbę dotychczas obliczonych łańcuchów (razem z obliczanym), a $j + 1$ aktualną długość obliczanego łańcucha. Dolne ograniczenie prawdopodobieństwa sukcesu w całej (jednej!) tabeli o m łańcuchach i długości łańcucha t wynosi:

$$P_{\text{single}}(S) \geq \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{t} \left(1 - \frac{it}{N}\right)^{j}. \tag{3.1.3}$$

W przypadku, gdy zamiast jednej tablicy zbudujemy l krótszych tablic z różnymi funkcjami redukcji (modyfikacja pierwotnej koncepcji Hellmana), uzyskamy sytuację, w której prawdopodobieństwa złączeń będziemy rozpatrywać dla każdej tablicy z osobna. Inaczej mówiąc, zmiana funkcji redukcji oznacza, że nowy łańcuch na pewno nie złączy się z żadnym z poprzednich, a każdy kolejny po nim nie powoduje złączenia z prawdopodobieństwem od nowa spadającym od jedności. Otrzymamy w ten sposób następujące ograniczenie prawdopodobieństwa sukcesu:

$$P(S) \geq 1 - \left(1 - \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{t} \left(1 - \frac{it}{N}\right)^{j}\right)^{l}. \tag{3.1.4}$$

Takie podzielenie fazy obliczeń wstępnych powoduje, że każda tablica może być generowana oddzielnie, czyli daje możliwość zrównoleglenia obliczeń (także uodparnia przeprowadzenie fazy na nieoczekiwane sytuacje, tzn. bardzo ważna okazuje się możliwość podzielenia na powtarzalne i co najwyżej kilkudniowe etapy). Jednak wraz ze wzrostem liczby tablic rośnie liniowo złożoność obliczeniowa fazy ataku w czasie rzeczywistym.

Pierwotna metoda Hellmana zakładała odtworzenie szukanego klucza dla szyfru z N kluczami w $N^{\frac{2}{3}}$ operacjach, wykorzystując $N^{\frac{2}{3}}$ słów pamięci. Atak ten był wielokrotnie rozszerzany i ulepszany. Najważniejsze z tych modyfikacji, które zostały wykorzystane w ataku, opisujemy poniżej.

3.2. Kompromis Oechslina

Zaproponowana przez Oechslina [2] metoda generowania łańcucha zakłada zmianę wobec pierwotnej idei polegającą na tym, żeby w każdym kolejnym kroku

(kolumnie macierzy) generowania łańcuchów stosować inną funkcję redukcji. Taki krok generowania łańcuchów nazywać będziemy *rundą*, a element macierzy przyjmie postać $X_{ij} = f_{j-1}\left(X_{i,j-1}\right)$. W ten sposób kolizja łańcuchów spowoduje złączenie tylko wtedy, gdy nastąpi w tej samej rundzie. Jeżeli kolizja nastąpi w różnych rundach łańcuchy zastosują inne funkcje redukcji i złączenie nie wystąpi. Dla łańcucha wykorzystującego liczbę rund *r*, jeżeli wystąpi kolizja, szansa wystąpienia złączenia wynosi tylko $\frac{1}{r}$. W tak zmodyfikowanej metodzie prawdopodobieństwo sukcesu pojedynczej tablicy rozmiaru $m \times r$ przyjmuje postać:

$$P(S) = 1 - \prod_{i=1}^{r}\left(1 - \frac{m_i}{N}\right), \tag{3.2.1}$$

gdzie przez m_i rozumiemy liczbę unikalnych stanów w *i*-tej rundzie całej tablicy. To znaczy w pierwszej rundzie unikalnych stanów będziemy mieli tyle, ile wynosi wysokość tablicy, tj. $m_1 = m$. W drugiej rundzie spodziewamy się wystąpienia stanów z pierwszej rundy rozłożonych równomiernie w przestrzeni stanów, stąd:

$m_2 = N(1-(1-1/N)^{m_1}) \approx N(1-e^{(-m_1/N)})$, a ogólnie:

$$m_{n+1} = N\left(1 - \left(1 - \frac{1}{N}\right)^{m_n}\right) \approx N\left(1 - e^{-\frac{m_n}{N}}\right). \tag{3.2.2}$$

Tak określone prawdopodobieństwo nie jest dane wzorem jawnym, dlatego w dalszych rozważaniach posługiwać się będziemy szerszymi ograniczeniami prawdopodobieństwa dla klasycznych tablic Hellmana.

3.3. Punkt rozróżnialny

Ron Rivest w [3] zaproponował, żeby punkty końcowe spełniały specjalne, łatwe do sprawdzenia kryterium, np. pierwsze 10 bitów to zera. Niech $K \in \{0, 1\}^k$ oraz $d \in \{0,\ldots,k-1\}$. *K* jest *punktem rozróżnialnym* wtedy, gdy istnieje łatwo sprawdzalna własność, która zachodzi dla *K* i nie zachodzi dla $2^k - 2^{k-d}$ innych elementów z $\{0, 1\}^k$. Zatem faza ataku w czasie rzeczywistym zostaje zmodyfikowana tak, że w momencie gdy atakujący zdobył szyfrogram, nie sprawdza, czy jest on punktem końcowym w tabeli, a generuje łańcuch aż do momentu uzyskania stanu spełniającego kryterium punktu rozróżnialnego. Dopiero wtedy wykonywane jest wyszukanie otrzymanej wartości w tablicy, co istotnie zmniejsza liczbę koniecznych odczytów pamięci. Natomiast w fazie obliczeń wstępnych w sytuacji, gdy generowanie łańcucha nie kończy się po wygenerowaniu spodziewanej (omówione szerzej w 3.4) liczby stanów,

podejrzewamy, że wykryliśmy pętlę i przerywamy dalsze generowanie łańcucha. W ten sposób wygenerowane tablice są wolne od pętli (krótszych od 2^d). Dzięki tej metodzie także złączenia zostaną wykryte w fazie obliczeń wstępnych. Dodatkowo nie bez znaczenia jest fakt, że potrzebujemy mniej pamięci, żeby przechowywać tak zbudowane pary punktów początkowych i końcowych, bo rozróżnialny punkt końcowy jest krótszy o d bitów.

3.4. Modyfikacja koncepcji punktu rozróżnialnego Rivesta

Oryginalna metoda Rivesta zakłada zliczanie, w ilu krokach punkt rozróżnialny został osiągnięty, oraz przechowywanie tej liczby. Łańcuchy, które nie osiągają punktu rozróżnialnego w zakładanej liczbie kroków, są przerywane. W ten sposób faza obliczeń wstępnych chroniona jest przed wystąpieniami cykli, zanim punkt rozróżnialny zostanie osiągnięty [4, 5].

Cykl to sytuacja polegająca na istnieniu i, j, gdzie $i \neq j$, dla których $X_{i,m} = X_{j,m}$.

W opisywanym ataku nie zliczamy i nie zapamiętujemy liczby kroków, po których punkt rozróżnialny został osiągnięty. Pozwala nam to ograniczyć pamięciową złożoność fazy obliczeń wstępnych, a zabezpieczenie fazy przed cyklem realizujemy poprzez wybór długości punktu rozróżnialnego, tzn. przyjęliśmy, że punkt rozróżnialny powinien zostać wybrany w sposób, który zapewni znikome prawdopodobieństwo wystąpienia cyklu w rundzie [6]. Niech $P_1(l)$ będzie prawdopodobieństwem osiągnięcia punktu rozróżnialnego w mniej niż l krokach, a $P_2(l)$ będzie oznaczało prawdopodobieństwo, że punkt rozróżnialny nie zostanie osiągnięty w l krokach. $P_1(l) = 1 - P_2(l)$.

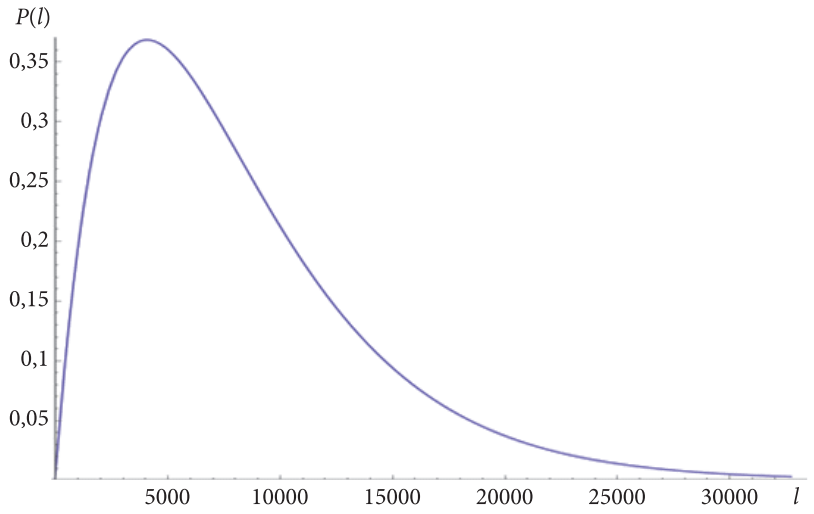
$$P_2(l) = \prod_{i=0}^{l-1} \left(1 - \frac{2^{k-d}}{2^k - i}\right). \tag{3.4.1}$$

Przyjmując, że $i = \frac{l-1}{2}$, bo $i \ll 2^k$, otrzymamy:

$$P_2(l) \approx \left(1 - \frac{2^{k-d}}{2^k - \frac{l-1}{2}}\right)^l \text{ oraz } P_1(l) \approx 1 - \left(1 - \frac{2^{k-d}}{2^k - \frac{l-1}{2}}\right)^l.$$

Zatem prawdopodobieństwo osiągnięcia punktu rozróżnialnego w dokładnie l krokach wynosi $P(l) = P_1(l) - P_1(l-1)$, zakładając, że $P_1(0) = 0$.

Dla $k = 64$ i $d = 12$ otrzymamy wykres zilustrowany na rysunku 2.



Rys. 2. Wykres prawdopodobieństwa osiągnięcia punktu rozróżnialnego w dokładnie *l* krokach

Eksperymentalna długość cyklu w przypadku A5/1 wynosi $2^{20,4}$ kroków, stąd wniosek, że zaproponowany wybór punktu rozróżnialnego zapewnia znikome prawdopodobieństwo wystąpienia cyklu. Kryterium punktu rozróżnialnego stosowane jest w każdej, a nie tylko ostatniej, jak proponował Rivest, rundzie, tzn. generowanie łańcucha osiąga stan pośredni (po którym przechodzi do kolejnej rundy) wtedy, gdy spełnia kryterium punktu rozróżnialnego.

3.5. Pominięte udoskonalenia

W pracy pominięte zostały udoskonalenia, których wpływ na wybór parametrów ataku, czyli sposób poszukiwania kompromisu czasu i pamięci, jest niewielki, a zostaje on w dalszych obliczeniach uwzględniony, bo ma wpływ na kształt danych. Udoskonalenia te wpływają na skuteczność ataku, ale nie zmieniają metody projektowania ataków z fazą obliczeń wstępnych.

Udoskonalenia te to:

— szybsze wykrywanie fałszywych alarmów za pomocą punktów sprawdzeń Avoine'a w fazie ataku w czasie rzeczywistym [7],
— zmienny punkt rozróżnialny Hongy'ego [8],
— kwestie dotyczące zapisu rezultatu fazy obliczeń wstępnych (kompresja pamięci potrzebnej do zapisania pary <punkt końcowy, punkt początkowy>),
— metody szybszego przeszukiwania tablic (rodzaj indeksu).

4. Projektowanie ataku

Projektowanie fazy obliczeń wstępnych polega na wyborze zestawu wielu parametrów, których podzbiór wyodrębniony ze względu na istotność został tutaj przedstawiony. Poczynione uproszczenia nie wpływają na wynik optymalizacji w sposób zasadniczy.

4.1. Opis dziedziny problemu, ustalenie zbiorów możliwych wartości zmiennych w podziale na zmienne decyzyjne, wskaźniki oraz dane

Na wybór parametrów ataku można patrzeć jak na zadanie optymalizacji, w ramach którego należy ustalić zbiór zmiennych decyzyjnych, zbiór ograniczeń, zbiór kryteriów oceny [9].

Wyróżniamy następujące zmienne decyzyjne:

— długość łańcucha $t \in Z_+$ (zbioru liczb całkowitych dodatnich) — wartość ta zależy bezpośrednio od dwóch innych wartości, tj. długości punktu rozróżnialnego d oraz liczby rund $r \in Z_+$, ale dla uproszczenia zadania zredukujemy ten podział, a tym samym liczbę zmiennych decyzyjnych, i będziemy się posługiwać tylko długością łańcucha $t = r \times 2^d$. Wybór długości punktu rozróżnialnego d opisany został w punkcie 3.3,

— liczba łańcuchów w jednej tablicy $m \in Z_+$,

— liczba tablic $l \in Z_+$

oraz wskaźnik $k \in Z_+$ oznaczający wymaganą liczbę tekstów jawnych. Warto tu zauważyć, że zgodnie z algorytmem ataku w czasie rzeczywistym liczba przeszukiwań tablic jest iloczynem liczby rund, liczby tablic oraz liczby tekstów jawnych wynosi zatem:

$$rlk = \frac{tlk}{2^d}. \tag{4.1.1}$$

Do danych w zadaniu optymalizacji zaliczać będziemy:

— m_0 — ilość pamięci (średnia liczba bitów) koniecznej do zapisania pary<punkt końcowy, punkt startowy>,

— e_0 — czas potrzebny na wygenerowanie pojedynczego bloku strumienia klucza wyrażony w sekundach,

— k_0 — liczba tekstów jawnych uzyskanych z jednej przechwyconej jednostki (np. ramki),

— s_0 — czas pojedynczego wyszukania punktu startowego w jednej tablicy (czas odczytu z dysku) i odtworzenia z niego łańcucha wyrażony w sekundach,

— M — ograniczenie dostępnej do zapisania fazy obliczeń wstępnych pamięci wyrażone w bajtach,

— T — ograniczenie złożoności czasowej T fazy obliczeń wstępnych wyrażone w sekundach,
— $C_{\text{on line}}$ — ograniczenie złożoności czasowej fazy ataku w czasie rzeczywistym wyrażone w sekundach,
— d — wybrana długość punktu rozróżnialnego (3.3),
— $r_0 = e_0 \times 2^d$ — czas potrzebny do wygenerowania pojedynczego stanu rundy.

W chwili podejmowania decyzji decydent musi znać wartości wszystkich danych.

4.2. Określenie dopuszczalnych wartości zmiennych decyzyjnych

Jeżeli chodzi o zbiór początkowych ograniczeń zbioru decyzji dopuszczalnych, to mamy:

- ograniczenie pamięci koniecznej do zapisania fazy obliczeń wstępnych M:

$$M \geq m \cdot l \cdot m_0, \tag{4.2.1}$$

- ograniczenie złożoności czasowej T fazy obliczeń wstępnych:

$$T \geq t \cdot l \cdot m \cdot e_0, \tag{4.2.2}$$

- ograniczenie złożoności czasowej fazy ataku w czasie rzeczywistym $C_{\text{on line}}$:

$$C_{\text{on line}} \geq \frac{r+1}{2} \cdot l \cdot k_0 \cdot s_0. \tag{4.2.3}$$

Ostatecznie złożoność obliczeniowa fazy ataku w czasie rzeczywistym zależeć będzie wprost od liczby wymaganych tekstów jawnych oraz liczby tablic. Ze wzrostem liczby tablic rośnie prawdopodobieństwo sukcesu dla jednego tekstu jawnego, ale także złożoność tej fazy. Stąd wniosek, że nie ma potrzeby zwiększania prawdopodobieństwa sukcesu przez podnoszenie liczby tablic w sytuacji, gdy minimalna liczba tekstów jawnych wynika ze specyfiki ich źródła. To znaczy, jeżeli wiemy, że przechwycenie jednej ramki komunikacji dostarcza kilkuset tekstów jawnych, to zwiększanie prawdopodobieństwa sukcesu powyżej odwrotności liczby tekstów jawnych przez zwiększanie liczby tablic nie poprawi skuteczności ataku, a istotnie zwiększy złożoność fazy ataku w czasie rzeczywistym. Liczba tablic musi zatem zostać z góry ograniczona, a ograniczenie to powinno zależeć od specyfiki źródła tekstów jawnych.

- Dodatkowo wprowadzamy ograniczenia wynikające z utrzymania wysokiego prawdopodobieństwa sukcesu dla pojedynczej tablicy [10, 11].

Chcąc przekształcić prawdopodobieństwo sukcesu dla jednej tablicy Hellmana podane wzorem 3.1.3, przyjmujemy, że $e^{-x} \cong 1-x$, i otrzymujemy przybliżenie:

$$\left(1-\frac{it}{N}\right) \cong e^{\frac{-it}{N}} \tag{4.2.4}$$

oraz:

$$\left(1-\frac{it}{N}\right)^{j} \cong e^{\frac{-ijt}{N}}. \tag{4.2.5}$$

Jeżeli $e^{\frac{-ijt}{N}} \cong e^{\frac{-mt^2}{N}}$ oraz $mt^2>>N$, to składniki sumy: $\frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{t} e^{\frac{-ijt}{N}}$ będą szybko maleć [12]. Stąd wniosek, że nie ma zysku ze zwiększania m i t poza granicę $mt^2 = N$. Jeżeli natomiast $mt^2<<N$, $i >> 1$oraz $t >> 1$, to:

$$P(S) \geq \frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{t}\left(1-\frac{it}{N}\right)^{j} \cong \frac{1}{t}\sum_{i=1}^{m}\frac{1-e^{\frac{-it^2}{N}}}{\frac{it}{N}}\frac{t}{N} \cong \frac{1}{t}\int_{0}^{\frac{mt^2}{N}}\frac{1-e^{-tx}}{x}dx \cong \frac{g(u)mt}{N}, \tag{4.2.6}$$

gdzie $u=\frac{mt^2}{N}$, $g(u)=\frac{1}{u}\int_{o}^{u}\frac{1-e^{-x}}{x}dx$,

$g(u)$ oznacza dolne ograniczenie „pokrycia" przestrzeni klucza.

Tabela 1

Dolne ograniczenie „pokrycia" przestrzeni klucza

u	$g(u)$	u	$g(u)$
2^{-4}	0,9840	2^{0}	0,7965
2^{-3}	0,9688	2^{1}	0,6596
2^{-2}	0,9408	2^{2}	0,4918
2^{-1}	0,8876	2^{3}	0,3320

Z tabeli 1 wynika, że wartość funkcji pokrycia gwałtownie spada, gdy $u > 2^{-1}$, stąd wniosek, że aby zachować wysokie prawdopodobieństwo sukcesu, atakujący musi wybrać m i t tak, aby spełniona została nierówność:

$$N > 2mt^2. \tag{4.2.7}$$

4.3. Kryterium oceny

Chcąc rozumowanie uogólnić dla przypadku, gdy mamy wiele tablic, otrzymamy:

$$P(S) = 1 - (1 - P_{\text{single}}(S))^l \geq 1 - (1 - g(u)\frac{mt}{N})^l \approx 1 - e^{-g(u)\frac{lmt}{N}} = P(u,v),$$

gdzie $v = \frac{lmt}{N}$.

Wiemy, że $g(u) < 1$ dla każdego $u > 0$. Stąd

$P(u, v) < 1 - e^{-v}$. Dla $v = 1$ mamy $P(u, 1) < 0{,}63$.

Stąd wniosek, że v powinno być istotnie większe od 1, np. $P(u, 2) < 0{,}86$; $P(u, 3) < 0{,}95$; $P(u, 4) < 0{,}98$, co implikuje ograniczenie postaci:

$$v > 2 \Rightarrow \frac{lmt}{2} > N.$$

Jednak w praktyce tak opisane kryterium jest niemożliwe do spełnienia. Dlatego złagodzone kryterium przyjmie postać:

$$P > \frac{2^d}{tlk_0}, \tag{4.3.1}$$

gdzie k_0 oznacza liczbę możliwych do zdobycia w praktyce tekstów jawnych, czyli prób przeszukiwań wszystkich tablic [13, 14].

Ponieważ wybranym kryterium optymalizacji jest liczba tekstów jawnych (a pośrednio złożoność obliczeniowa fazy ataku w czasie rzeczywistym), przyjmujemy funkcję celu postaci:

$$k(l,m,t) = \frac{2^d}{\left(1 - e^{-\frac{lmt}{N}}\right)lt}. \tag{4.3.2}$$

4.4. Sformułowanie zadania

Dla danych

$$a \in A = \left\{m_0, e_0, k_0, s_0, d, M, T, C_{\text{on line}} \in \mathbb{Q}_+ \times \mathbb{Q}_+ \times Z_+ \times \mathbb{Q}_+ \times Z_+ \times Z_+ \times Z_+ \times \mathbb{Q}_+\right\}$$

wyznaczyć

$$\left(l^*,m^*,t^*\right)\in X_0=\left\{l,m,t\in Z_+ : M\geq mlm_0,\ T\geq tlme_0, C_{\text{on line}}\geq \frac{\frac{t}{2^d}+1}{2}lk_0s_0, N>2mt^2\right\}$$

takie, że

$$\frac{2^d}{\left(1-e^{-\frac{l^*m^*t^*}{N}}\right)l^*t^*}=\min_{l,m,t}\frac{2^d}{\left(1-e^{-\frac{lmt}{N}}\right)lt}, \tag{4.4.1}$$

co oznacza, że poszukujemy takiego zestawu zmiennych decyzyjnych spełniających wymienione ograniczenia, dla których liczba znanych tekstów jawnych dająca wysokie prawdopodobieństwo na odnalezienie pierwotnego stanu rejestrów (co jest równoznaczne z odnalezieniem klucza) jest najmniejsza, tj. faza ataku w czasie rzeczywistym jest najkrótsza. Szukana zatem wartość to min $k(l, m, t)$, a [min $k(l, m, t)$] oznaczać będzie spodziewaną liczbę tekstów jawnych, dla których atak się powiedzie.

5. Przykład zastosowania metody ataku dla algorytmu A5/1

Na tym etapie musimy opisać dostępne zasoby, z których wynikać będą ograniczenia. Przykład zakłada wykorzystanie zasobów sprzętowych dostępnych tzw. użytkownikowi nieinstytucjonalnemu, tj. takiemu, który ma ograniczone środki i dostęp do zasobów.

Atak z fazą obliczeń wstępnych na A5/1 został już raz przeprowadzony przez grupę Karstena Nohla [15]. Ograniczenia przyjęte przez grupę Nohla zostały w niniejszym przykładzie wykorzystane, a sam atak jest podobny, choć różni się m.in. wykorzystaniem punktu rozróżnialnego oraz oczywiście parametrami, których optymalizacji dotyczy artykuł.

Zakładamy zbiór ograniczeń:

- Ograniczenie pamięci. Zakładamy, że:
  - dostępna pamięć na zapisanie fazy obliczeń wstępnych wynosi $M = 2TB$ (dostępny dysk na rynku), jednak należy mieć na uwadze, że z przyczyn praktycznych/technicznych całego dysku nie zapiszemy, stąd realnie przestrzeń wyniesie około

$$M\ 0{,}85\times 2TB,$$

  - liczba bitów potrzebnych do zapisania łańcucha (pary punkt startowy–punkt końcowy) wynosi $m_0 = 60$ bitów (mógłby wynosić około 55 bitów, jednak tracimy kilka bitów na nieopisane w artykule udoskonalenia). Ograniczenie (4.2.1) przyjmie postać:

$$lm < 2^{37,86}.$$

- Ograniczenie złożoności czasowej fazy obliczeń wstępnych. Zakładamy, że:
  — czas potrzebny na wygenerowanie pojedynczego bloku strumienia klucza wynosi $e_0 = 2^{-29,6}$s przy wykorzystaniu dostępnych układów GPU, a czas potrzebny do wygenerowania pojedynczego stanu rundy wynosi

$$r_0 = 2^{-17,6}s,$$

  — czas, który możemy poświęcić na przeprowadzenie fazy obliczeń wstępnych, wynosi $T = 120$ dni. Ograniczenie (4.2.2) przyjmie postać:

$$tlm < 2^{52,91},$$

- Ograniczenie złożoności czasowej fazy ataku w czasie rzeczywistym. Zakładamy, że:
  — liczba dostępnych tekstów jawnych z jednej ramki LAPDm

$$k_0 = 200,$$

  — czas wygenerowania punktów końcowych ze znanych tekstów jawnych, ich wyszukania w tablicy oraz odtworzenia łańcucha wynosi (HDD):

$$s_0 = 2^{-7}\text{s},$$

  — $C_{\text{on line}} = 5$ min

$$l \leq 42,67 \approx 2^{5,41}.$$

- Ograniczenie wynikające *z* zachowania wysokiego „pokrycia" w pojedynczej tablicy (4.2.7):

$$mt^2 < 2^{63},$$

- $m, t, r, l \in Z+$.

Natomiast funkcja celu (4.3.2) przyjmie postać ($N = 64$):

$$f(l,m,t) = \frac{2^d}{\left(1 - e^{-\frac{lmt}{2^{64}}}\right) lt}.$$

Z ograniczeń czasu i pamięci, zakładając maksymalne wykorzystanie dostępnej na fazę obliczeń wstępnych pamięci oraz czasu, wynika, że $t \leq 2^{15,05} \approx 2^{15}$. Stąd, wiedząc z uprzednich wyliczeń wynikających z modyfikacji punktu rozróżnialnego Rivesta, że $d = 12$, otrzymujemy, że liczba rund $r \leq 8$. Wtedy optymalizowana funkcja celu zakładająca maksymalną możliwą liczbę rund przyjmie postać:

$$f\left(l,m,t=2^{15}\right)=\frac{2^{12}}{2^{15}\left(1-e^{-\frac{lm2^{15}}{2^{64}}}\right)l}=\frac{1}{2^{3}\left(1-e^{-\frac{lm}{2^{49}}}\right)l}.$$

Natomiast z ograniczenia dotyczącego zachowania wysokiego „pokrycia" w pojedynczej tablicy wynika, że $m < 2^{33}$. Jednak węższym ograniczeniem na m jest przecięcie ograniczeń pamięci i złożoności czasowej fazy ataku w czasie rzeczywistym, z których wynika, że $m < 2^{32,45}$. *Z* ograniczenia dotyczącego liczby dostępnych w praktyce tekstów jawnych wynika, że $l \leq 42$.

$$f\left(l=42,m=2^{32,45},\, t=2^{15}\right)=\frac{1}{2^{3}\left(1-e^{-\frac{42\cdot 2^{32,45}}{2^{49}}}\right)42}\approx 6{,}8.$$

Natomiast $[\min f(l, m, t)] = 7$.

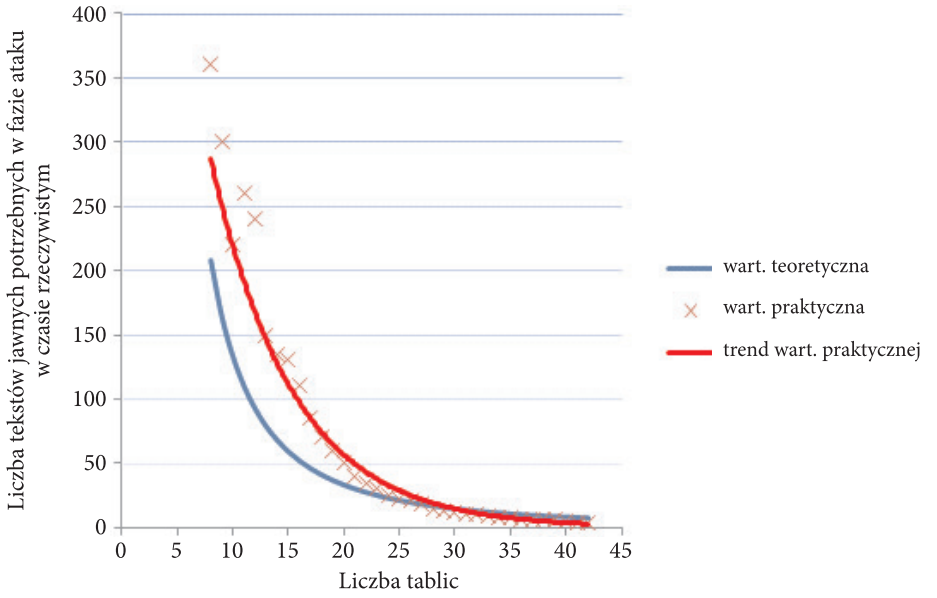
Należy zwrócić uwagę, że prowadzone obliczenia pomijają mniej ważne parametry ataku, o które powinien on zostać uzupełniony po uprzednim ustaleniu jego szkieletu, i mogą one ten szkielet nieznacznie modyfikować. I tak np. otrzymana wielkości tablicy wynosi $2^{32,45}$, co znaczy, że do zapisu w tablicy punktu startowego potrzebne będą 33 bity. Modyfikacja tej wartości do wielkości 2^{32} pozwoliłaby zapisać punkt startowy w 32 bitach, co oznacza sporą oszczędność pamięci całości fazy obliczeń wstępnych.

6. Wyniki

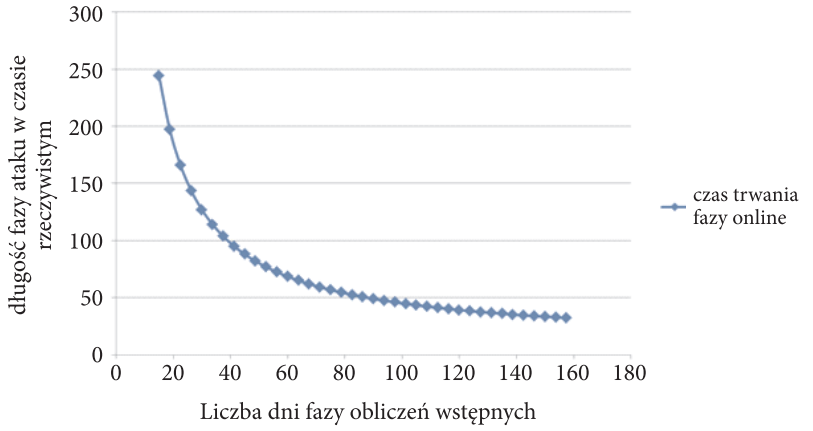
W czasie realizacji ataku dokonano pewnych zmian wobec projektu, wynikających z nieprzewidzianych wcześniej okoliczności. W czasie budowania pojedynczej tablicy zaskakująco dużo łańcuchów kończyło się tym samym punktem końcowym. Wygenerowanie dopiero około 2^{33} łańcuchów po odrzuceniu duplikatów na etapie sortowania łańcuchów po ich punktach końcowych dawało planowaną wielkość pojedynczej tablicy $2^{33,45}$, co znaczy, że około 31% wygenerowanych łańcuchów było duplikatami. Podwyższenie liczby łańcuchów w pojedynczej tablicy skutkuje wzrostem złożoności obliczeniowej fazy obliczeń wstępnych oraz koniecznością zapisywania punktu startowego na 33 bitach.

Wyniki eksperymentalne skuteczności całego ataku (mimo że po całej fazie obliczeń wstępnych wygenerowanych zostało około $2^{37,25}$ łańcuchów, co jest nieznacznie niższą wartością wobec projektu) są minimalnie lepsze od teoretycznych, co wynika z niedoskonałości szyfru A5/1 polegającej na tym, że prawdziwa przestrzeń klucza jest mniejsza od zakładanej. Porównanie wyników opisuje rysunek 3. Jako że suma bitów trzech rejestrów wynosi 64, to do wyliczeń na etapie projektu używano wielkości 2^{64}.

Jednak sposób inicjowania algorytmu A5/1 (100 pierwszych taktów jest pomijanych i dopiero kolejne 228 bitów wyjścia stanowi strumień klucza) powoduje, że realna przestrzeń klucza (po inicjacji) spada do około $2^{61, 63}$.



Rys. 3. Porównanie teoretycznej i praktycznej liczby tekstów jawnych koniecznych do uzyskania klucza



Rys. 4. Wykres relacji długości fazy ataku w czasie rzeczywistym do fazy obliczeń wstępnych

Charakterystyka spadku czasu trwania fazy w czasie rzeczywistym, która jest odwrotnie proporcjonalna do wzrostu prawdopodobieństwa skuteczności ataku względem wzrostu liczby tablic pokazuje, że wybrana funkcja celu minimalizująca jedynie liczbę potrzebnych tekstów jawnych jest nie dość dobra, bo przyrost prawdobobieństwa dla tablic liczby tablic większej od 30 jest znikomy. Dalsze badania powinny być zatem prowadzone w kierunku poszukiwania funkcji, która zapewni kryterium ograniczające spadek przyrostu prawdopodobieństwa dla kolejnych tablic.

7. Podsumowanie

W pracy sformułowano wnioski dotyczące projektowania ataków z fazą obliczeń wstępnych oraz przedstawiono wyniki zebrane z realizacji ataku na szyfr strumieniowy A5/1.

Otrzymane wyniki wskazują, że najbardziej nieprzewidzianym zjawiskiem była duża liczba duplikatów, przez co aby zachować pokrycie zgodne z projektem, wydłużona została faza obliczeń wstępnych. Z tego faktu wynika, że albo zbyt łagodne było kryterium skuteczności dla jednej tablicy, albo należy szukać modyfikacji ataku, który by ten parametr poprawił. Ponadto stwierdzono niedoskonałość metody projektowania ataków z fazą obliczeń wstępnych polegającą na małym przyroście prawdopodobieństwa skuteczności ataku dla liczby tablic większej od 30. Niedoskonałość ta to temat badań w najbliższym czasie. Prace będą prowadzone w kierunku optymalizacji innych kryteriów ataku — opisana metoda optymalizacji minimalizuje liczbę tekstów jawnych i pośrednio fazę ataku w czasie rzeczywistym, traktując czas potrzebny do realizacji fazy obliczeń wstępnych jako ograniczenie. Dodatkowo opisana metoda poszukiwania kompromisu czasu i pamięci zostanie poddana analizie skuteczności na innym szyfrze.

Przeprowadzone próby potwierdziły praktyczną realizowalność ataku na szyfr A5/1 (komunikację głosową w ramach GSM) przy ograniczonych zasobach. Należy zwrócić uwagę, że mimo różnic m.in. w podejściu do koncepcji punktu rozróżnialnego, wnioski z zaproponowanej metody projektowania kompromisu czasu–pamięci zastosowane do szyfru A5/1 nie są bardzo odległe od projektu Karstena Nohla.

LITERATURA

[1] M. Hellman, *A Cryptanalytic Time-Memory Tradeoff*, IEEE Transactions on Information Theory, 26, 1980.

[2] P. Oechslin, *Making* a *Faster Cryptanalytic Time-Memory Trade-Off*, Advances in Cryptology, Springer-Verlag, 2003.

[3] S. Babbage, *A Space/Time Tradeoff in Exhaustive Search Attacks on Stream Ciphers; European Convention on Security and Detection*, IEE Conference Publication 408, 1995.

[4] J. Borst, B. Preneel, J. Vandewalle, *On the Time-Memory Tradeoff between ex-haustive key search and table precomputation*, Proc. of the 19th Symposium in Information Theory in the Benelux, WIC, 1998, 111-118.4.

[5] D.E. Denning, *Cryptography and Data Security*, 100, Addison-Wesley, 1982.

[6] J. Borst, B. Preneel, J. Vandewalle, *A Time-Memory Tradeoffs using Distinguished Points*, Technical report ESAT-COSIC Report 98-1, 1998.

[7] K. Nohl, C. Paget, *GSM — SRSLY?*, 26th Chaos Communication Congress (26C3) Berlin, 2009.

[8] J. Hongy, K. Chul Jeongz, E. Young Kwony, *Variants of the Distinguished Point Method for Cryptanalytic Time Memory Trade-offs*, Cryptology ePrint Archive, Report 054, 2008.

[9] E. Barkan, E. Biham, A. Shamir, *Rigorous Bounds on Cryptanalytic Time-Memory Tradeoffs*, Advances in Cryptology, Proceedings of Crypto 2006, Springer-Verlag, 2006.

[10] G. Avoine, P. Junod, P. Oechslinl, *Characterization and Improvement of Time-Memory Trade-Off Based on Perfect Tables*, ACM Transactions on Information and System Security (TISSEC), 11, 4, July 2008, Article 17.

[11] K. Kusuda, T. Matsumoto, *Optimization of Time-Memory Trade-Off Cryptanalysis and Its Application to DES, FEAL-32, and Skipjack*, IEICE Transactions on Fundamentals, E79-A(1), 1996, 35-48.

[12] Kim T. Matsumoto, *Achieving higher success probability in time-memory trade-off cryptanalysis without increasing memory size*, IEICE Transactions on Communications/Electronics/Information and Systems, 1999.

[13] A. Biryukov, *Some Thoughts on Time-Memory-Data Tradeoff*, IACR ePrint Report 207, 2005.

[14] A. Fiat, M. Naor, *Rigorous Time-Space Tradeoffs for Inverting Functions*, SIAM J. Computing, 29(3), 1999, 790-803.

[15] A. Biryukov, A. Shamir, D. Wagner, *Real Time Cryptanalysis of A5/1 on a PC*, Proceedings of Fast Software Encryption 2000.

K. KANCIAK, M. MISZTAL

Time memory trade off attack on symmetric ciphers optimization

Abstract. The paper discusses variants of time memory tradeoff attack. The article uses A5/1 stream cipher as an example. The article describes known variants of the attack with their theoretical effectiveness. Results of tradeoff parameters optimization are presented. The article covers conclusions on tradeoffs design and comparison of obtained A5/1 algorithm cryptanalysis results with expected theoretical values.

Keywords: cryptology, cryptanalysis, stream cipher, TMTO, time memory tradeoff