Atak tabelą decymalizacyjną dla złamania PIN

Streszczenie

Prezentujemy atak na moduły bezpieczeństwa sprzętowego, z których korzystają banki detaliczne ,bezpieczne przechowywanie i weryfikacja kodów PIN klientów w bankomatach. Korzystając z adaptacyjnej tabeli decymlizacji i odgadywania , maksymalna liczba informacji jest poznawana na temat prawdziwego numeru PIN po każdej próbie. Aby ustalić czterocyfrowy kod PIN za pomocą tej techniki, potrzeba średnio 15 prób, zamiast 5000 zamierzonych. W pojedynczej 30-minutowej przerwie na lunch osoba atakująca może odkryć około 7000 kodów PIN zamiast 24 za pomocą metody brute force. Dzięki 300$ z limitem na karcie, potencjalna nagroda zostanie zwiększona z 7200 $ do 2,1 miliona USD, a jeden zmotywowany atakujący może wypłacić 30 - 50 000 dolarów .Atak ten stanowi zatem poważne zagrożenie dla bezpieczeństwa bankowego.

1. Wstęp

Bankomaty (ATM) są wykorzystywane codziennie przez miliony klientów do wypłat gotówki z ich kont. Jednak szerokie rozmieszczenie i czasami ustronne lokalizacje bankomatów czynią je idealnymi narzędziami dla przestępców, aby zamienić identyfikowalne pieniądze elektroniczne na czystą gotówkę. Kod PIN klienta jest podstawowym środkiem bezpieczeństwa przeciwko oszustwom; fałszowanie paska magnetycznego na kartach jest trywialne w porównaniu z przejęciem PIN. Przestępca uliczny może łatwo ukraść kartę gotówkową, ale jeśli nie zauważy, że klient wpisuje kod PIN w bankomacie, może on zgodzić się tylko z trzema domyślnymi z 10 000 PIN-ów i rzadko by mu się to udało. Nawet jeśli zakończy się sukcesem, jego kradzież wciąż nie może przekroczyć dziennego limitu wypłat wynoszącego około 300 $. Jednak programiści bankowi mają dostęp do systemów komputerowych, których zadaniem jest bezpieczne przechowywanie kodów PIN, które zwykle składają się z komputera typu mainframe połączonego z Hardware Security Module (HSM), który jest odporny na manipulacje i ma ograniczony interfejs API taki, że będzie tylko odpowiedzieć odpowiedzią TAK / NIE na domysły klienta. Niezręczną metodą ataku jest, aby skorumpowany programista bankowy napisał program, który próbuje wszystkich PIN-ów dla danego konta, a przy przeciętnym szczęściu wymagałoby to odkrycia około 5000 transakcji Typowy HSM może sprawdzić 60 próbnych kodów PIN na sekundę oprócz normalnego obciążenia, więc skorumpowany pracownik wykonujący program podczas 30-minutowej przerwy na lunch może uzyskać tylko około 25 PIN-ów, ale HSM-y stosują kilka wspólnych PIN-ów. Pierwsze bankomaty to maszyny IBM 3624, szeroko wprowadzone w Stanach Zjednoczonych około 1980 r., a większość metod generowania kodu PIN opiera się na ich podejściu, które obliczały oryginalny kod PIN klienta zaszyfrowany numer konta wydrukowany na przedniej stronie karty klienta tajnym kluczem DES nazywanym „kluczem generowania kodu PIN ". Powstały zaszyfrowany tekst jest konwertowany na szesnastkowy, a pierwsze cztery cyfry są pobierane. Każda cyfra ma zasięg `0'-` F '. Aby przekonwertować tę wartość na kod PIN, który można wpisać na klawiaturze dziesiętnej, używana jest "tabela decymalizacji", czyli mapowanie wielu do jednego między cyframi szesnastkowymi i cyframi numerycznymi. Lewa tabela decymalizacji na rysunku jest typowa.

0123456789ABCDEF	0123456789ABCDEF
0123456789012345	0000000100000000

Ta tabela nie jest uważana za wrażliwą przez wiele HSM, więc można podać dowolną tabelę wraz z numerem konta i próbnym PIN-em. Ale manipulując zawartością tabeli, można dowiedzieć się znacznie więcej o wartości kodu PIN niż po prostu wykluczać pojedynczą kombinację. Na przykład, jeśli używana jest prawostronna tabela, dopasowanie z próbnym pinem 0000 potwierdza, że kod PIN nie zawiera

liczby 7, eliminując w ten sposób ponad 10% możliwych kombinacji. Najpierw przedstawimy prosty schemat, w którym można uzyskać większość kodów PIN w około 24 domysłach, a następnie schemat adaptacyjny, który maksymalizuje ilość informacji uzyskanych z każdego odgadnięcia i zajmuje średnio 15 odgadnień. Na koniec przedstawiono trzeci schemat, który pokazuje, że atak jest nadal możliwy, nawet gdy atakujący nie może kontrolować domysłów, przeciwko którym dopasowywany jest kod PIN. Sekcja 2 artykułu określa atak w kontekście środowiska bankowości detalicznej i wyjaśnia, dlaczego nie można go wykryć za pomocą typowych środków bezpieczeństwa. Sekcja 3 opisuje metody generowania i weryfikacji kodów PIN, a Sekcja 4 opisuje algorytmy, które zaprojektowaliśmy w szczegółach. Prezentujemy wyniki z autentycznych badań w sekcji 5, omawiamy środki zapobiegawcze w rozdziale 6 i wyciągamy wnioski w sekcji 7.

2 Bezpieczeństwo bankowe

Banki tradycyjnie odgrywały rolę w zwalczaniu oszustw zarówno wewnętrznych, jak i zewnętrznych. Opracowały metody ochrony przed oszustwami poufnymi, w tym księgami podwójnego zapisu, separacją funkcjonalną i obowiązkowymi okresami świątecznymi dla pracowników, i uznają potrzebę regularnych audytów zabezpieczeń. Metody te skutecznie ograniczają oszustwa do akceptowalnego poziomu dla banków, a także w połączeniu z odpowiednimi ramami prawnymi dotyczącymi odpowiedzialności, mogą również chronić klientów przed konsekwencjami oszustwa. Jednakże rosnącej złożoności systemów komputerowych banków nie towarzyszyły wystarczające postępy w zrozumieniu metod zapobiegania oszustwom. Wprowadzenie HSM w celu ochrony kodów PIN klienta było krokiem we właściwym kierunku, ale w 2002 r. urządzenia te nie zostały powszechnie przyjęte, a te, które są używane, pokazywano raz po raz, aby nie były odporne na atak. Typowa praktyka bankowa ma jedynie na celu ograniczenie nadużyć do akceptowalnego poziomu, ale to źle przekłada się na wymogi bezpieczeństwa; niemożliwe jest dokładne oszacowanie narażenia na bezpieczeństwo danej usterki, które może być izolowanym incydentem lub wierzchołkiem ogromnej góry lodowej. Ten rodzaj zarządzania ryzykiem jest bezpośrednio w konflikcie z nowoczesną praktyką projektowania zabezpieczeń, w której ważna jest solidność. Istnieją użyteczne analogi w projektowaniu algorytmów kryptograficznych. Projektanci, którzy robią "tylko wystarczająco silne" algorytmy i zapewniają solidność w handlu, aby uzyskać zatwierdzenie prędkości lub eksportu, grają niebezpieczną grę. Złamanie szyfru telefonii komórkowej GSM A5 jest tylko jednym z przykładów. Ponieważ nadal stosuje się algorytmy kryptograficzne "tylko wystarczająco silne", ryzyko oszustwa wynikające z domniemanych fałszywych kodów PIN jest nadal uważane za akceptowalne, ponieważ odgadnięcie pojedynczego kodu PIN z maksymalną szybkością transakcji powinno zająć co najmniej 10 minut. przy modułach wdrożonych w latach 80. Klienci powinni zauważyć wypłaty fantomowe i zgłosić je, zanim napastnik zdobędzie wystarczającą liczbę PIN-ów, aby wygenerować istotną odpowiedzialność za banki. Nawet z najnowszymi HSMem, który obsługuje stawkę transakcji dziesięć razy wyższą, sumy pieniędzy, które napastnik może ukraść, są niewielkie z punktu widzenia banku. Ale teraz, gdy tabela decymalności PIN została zidentyfikowana jako element danych istotnych dla bezpieczeństwa, a ataki opisane w tym dokumencie pokazują, jak wykorzystać niekontrolowany dostęp do niej, brut-force jest szybsze o dwa rzędy wielkości. Wystarczająco dużo kodów PIN, aby odblokować dostęp do ponad 2 milionów dolarów, można sprawdzić w czasie jednej przerwy na lunch! Bardziej złowrogim zagrożeniem jest popełnianie mniejszej kradzieży, w której konieczne transakcje są dobrze przeprowadzane w obrębie ścieżek audytu banków. Weryfikacje PIN niekoniecznie są kontrolowane centralnie, a jeśli przyjmiemy, że takowe są, to około 15 wymaganych transakcji będzie trudnym zadaniem dla audytora, aby znaleźć się wśród milionów strumieni. Systemy wykrywania włamań nie radzą sobie znacznie lepiej (przypuśćmy, że bank ma wyjątkowo rygorystyczny system kontroli, który śledzi liczbę nieudanych prób dla każdego konta, podnosząc alarm, jeśli wystąpią trzy awarie z rzędu. Osoba atakująca może wykryć PIN bez podnoszenia alarmu, wprowadzając transakcje ataku tuż przed

faktycznymi transakcjami od klienta, które zresetują licznik. Bez względu na politykę systemu wykrywania włamań nie można ich utrzymać w tajemnicy, dlatego kompetentny programista mógłby ich uniknąć. Właśnie dlatego, że HSM zostały wprowadzone do banków, systemy operacyjne typu mainframe tylko w zadowalający sposób chroniły integralność danych i nie można było ufać, że zachowują one dane od programistów. Tak więc, jak ekonomia bezpieczeństwa wygląda na to, że rozwijają się one w dojrzałe pole, wydaje się, że banki muszą zaktualizować swoje strategie zarządzania ryzykiem, aby uwzględnić zmienny charakter branży bezpieczeństwa. Na ich klientach ciąży również obowiązek ponownej oceny odpowiedzialności za oszustwa w indywidualnych przypadkach, ponieważ zmiany w zakresie bezpieczeństwa komputerowego nieustannie zmieniają krajobraz, w którym toczą się spory prawne między bankiem a klientem.

3 Techniki generacji i weryfikacji PIN

Istnieje szereg technik generowania i weryfikacji kodów PIN, z których każdy jest zastrzeżony dla konkretnego konsorcjum banków, które zamówiły system przetwarzania kodu PIN od innego producenta. IBM CCA obsługuje próbkę reprezentatywną. Mamy bardziej szczegółowo ustaloną metodę IBM 3624-O-set, ponieważ jest ona typowa dla użycia tablicy decymalizacyjnej.

3.1 Metoda wyprowadzania kodu PIN 3624 IBM Offset

Metoda IBM 3624-Offset została opracowana w celu obsługi pierwszej generacji bankomatów i została w związku z tym powszechnie przyjęta i naśladowana. Ta metoda została zaprojektowana w taki sposób, aby bankomaty w trybie offline mogły weryfikować numery PIN klientów bez potrzeby posiadania mocy obliczeniowej i pamięci masowej do manipulowania całą bazą danych rekordów kont klientów. Zamiast tego opracowano schemat, w którym kod PIN klienta można obliczyć na podstawie jego numeru konta poprzez szyfrowanie tajnym kluczem. Numer konta został udostępniony na pasku magnetycznym karty, więc bankomat musiał tylko bezpiecznie przechowywać pojedynczy klucz kryptograficzny. Przykładowa kalkulacja PIN jest pokazana na rysunku

```
Account Number        4556 2385 7753 2239
Encrypted Accno       3F7C 2201 00CA 8AB3

Shortened Enc Accno   3F7C

                 0123456789ABCDEF
                 0123456789012345

Decimalised PIN       3572

Public Offset         4344

Final PIN             7816
```

Numer konta jest reprezentowany za pomocą cyfr ASCII, a następnie interpretowany jako szesnastkowe wejście do szyfru blokowego DES. Po zaszyfrowaniu kluczem tajnym ”generowania PIN” , dane wyjściowe są konwertowane na szesnastkowe i wszystkie oprócz czterech pierwszych cyfr są odrzucane. Jednak te cztery cyfry mogą zawierać cyfry szesnastkowe "A" -F ", które nie są dostępne na

standardowej klawiaturze numerycznej i byłyby mylące dla klientów, więc są odwzorowywane z powrotem na cyfry dziesiętne za pomocą "tabeli decymalizacji"

```
0123456789ABCDEF
0123456789012345
```

Przykładowy kod PIN 3F7C zmienia się w 3572. Wreszcie, aby umożliwić posiadaczom kart zmianę kodów PIN, dodaje się przesunięcie, które jest przechowywane w bazie danych na komputerze mainframe wraz z numerem konta. Kiedy bankomat sprawdza wprowadzone kody PIN, po prostu odejmuje przesunięcie od karty przed sprawdzeniem wartości w stosunku do zdziesiątkowanego wyniku szyfrowania.

3.2 Interfejsy API modułu bezpieczeństwa sprzętu

Centra kontroli bankowej i bankomaty używają sprzętowych modułów zabezpieczeń (HSM), które są chronione kluczami do ochrony kodu PIN przed skorumpowanymi pracownikami i atakującymi fizycznie. HSM to koprocesor odporny na manipulacje, który uruchamia oprogramowanie zapewniające usługi kryptograficzne i związane z bezpieczeństwem. Jego interfejs API został zaprojektowany w celu ochrony poufności i integralności danych, jednocześnie umożliwiając dostęp zgodnie z zasadami użytkowania. Typowe finansowe API zawierają transakcje do generowania i weryfikowania kodów PIN, translacji odgadniętych kodów PIN między różnymi kluczami szyfrowania podczas ich podróży między bankami oraz obsługi wielu kluczowych funkcji zarządzania. Zasady użytkowania są zwykle ustawione tak, aby każdy, kto ma dostęp do komputera hosta, wykonywał codzienne polecenia, takie jak weryfikacja PIN, ale aby zapewnić, że wrażliwe funkcje, takie jak ładowanie nowych kluczy, mogą być wykonywane tylko za pomocą autoryzacji z wielu źródeł przez pracowników, którym ufają, że nie zmawiają się. "Common Cryptographic Architecture" firmy IBM to finasowe API implementowany przez szereg IBM HSM, w tym 4758, oraz koprocesor szyfrujący CMOS (odpowiednio dla komputerów PC i komputerów typu mainframe). Przykład kodu do weryfikacji PIN CCA pokazano na rysunku

```
Encrypted_PIN_Verify (

A_RETRES, A_ED, // kody powrotu 0,0 = tak 4,19 = nie

trial_pin_kek_in, pinver_key, // klucze szyfrowania dla wejść enc

(UCHAR *) "3624" "NONE" // Format bloku PIN

"F" // cyfra bloku blokującego PIN

(UCHAR *) "",

trial_pin, // encrypted_PIN_block

I_LONG (2),

(UCHAR *) "IBM-PINO" "PADDIGIT", // metoda weryfikacji PIN

I_LONG (4), // Liczba cyfr kodu PIN = 4

"0123456789012345" // Tabela decymalizacji

"123456789012" // PAN_data (numer konta)
```

"0000" // dane przesunięcia

Najważniejsze dane wejściowe do Encrypted_PIN_Verify to tabela decimalisation, PAN_data i encrypted_PIN_block. Pierwsze dwa są dostarczane w przejrzysty sposób i są łatwe do manipulowania przez atakującego, ale uzyskanie encrypted_PIN_block reprezentującego wybrany próbny PIN jest raczej trudniejsze

3.3 Uzyskanie wybranych zaszyfrowanych próbnych PIN-ów

Niektóre systemy bankowe zezwalają na wyraźne wprowadzanie próbnych kodów PIN z oprogramowania hosta. Na przykład ta funkcja może być wymagana do wprowadzania losowych kodów PIN podczas generowania bloków PIN dla schematów, które nie używają tabel decymalizacji. Odpowiednią komendą CCA jest Clear_PIN_Encrypt, która przygotuje zaszyfrowany blok PIN z wybranego PIN-u. Należy zauważyć, że włączenie tego polecenia niesie inne ryzyko, a także pozwala na nasze ataki. Jeśli nie ma randomizowanego wypełniania numerów PIN przed ich zaszyfrowaniem, osoba atakująca może utworzyć tabelę znanych próbnych zaszyfrowanych kodów PIN, porównać każdy przychodzący zaszyfrowany kod PIN z tą listą, a tym samym łatwo określić jego wartość. Jeśli nadal konieczne jest włączenie wyraźnego wprowadzenia kodu PIN w przypadku braku losowego wypełnienia, niektóre systemy mogą egzekwować to, że wyraźne kody PIN są szyfrowane tylko za pomocą klucza do tranzytu do innego banku (w takim przypadku atakujący nie może wykorzystać tych przypuszczeń jako danych wejściowych do lokalne polecenie weryfikacji. Tak więc, przy założeniu, że osoba atakująca nie ma dostępu do wyraźnego kodu PIN, jego drugą opcją jest wprowadzenie wymaganych domniemań w prawdziwym bankomacie i przechwycenie zaszyfrowanego numeru PIN, odpowiadającego każdemu domysłowi, gdy dotrze do banku. Nasz adaptacyjny stół decyzyjny wymaga tylko różnych próbnych kodów PIN {0000, 0001, 0010, 0100, 1000. Jednak atakujący może uzyskać tylko kodowane kody PIN w formacie blokowym, takim jak ISO-0, gdzie numer konta jest osadzone w bloku. Wymagałoby to od niego ręcznego wprowadzania kodów PIN próbnych w bankomacie dla każdego konta, które może zostać zaatakowane (ogromne przedsięwzięcie, które całkowicie pokonało strategię). Trzecim i bardziej niezawodnym sposobem działania atakującego jest wykorzystanie możliwości przesunięcia kodu PIN w celu przekonwertowania pojedynczego znanego kodu PIN na wymagane domysły. Ten znany kod PIN może zostać wykryty przez wymuszone zgadywanie lub po prostu otwarcie konta w tym banku. Pomimo wszystkich opcji uzyskiwania zaszyfrowanych próbnych PIN-ów można argumentować, że atak tabelą dziesiętnej nie jest możliwy do wykorzystania, chyba że można go wykonać bez jednego znanego próbnego kodu PIN. Aby rozwiązać te problemy, stworzyliśmy trzeci algorytm (opisany w następnej sekcji), który ma taką samą prędkość jak inne, i nie wymaga żadnych znanych lub wybranych próbnych PIN-ów.

4.Ataki Tabelą Decymalizacyjną

W tej sekcji opiszemy trzy ataki. Po pierwsze, możemy przedstawić prosty schemat statyczny 2-stopniowy, który wymaga jedynie średnio około 24 odgadnień. Wadą tej metody jest to, że potrzebuje prawie dwa razy tyle odgadnień w najgorszym przypadku. Pokażemy, jak przezwyciężyć tę trudność przez wykorzystanie adaptacyjnego podejścia i zmniejszyć liczbę koniecznych prób do 22. Na koniec przedstawiamy algorytm, który wykorzystuje zaszyfrowane przesunięcia PIN aby wydedukować PIN z jednego prawidłowego odgadnięcia, jak to jest zazwyczaj dostarczane przez klienta z bankomatu.

4.1 Schemat początkowy

Początkowy schemat składa się z dwóch etapów. Etap pierwszy określa, które cyfry są obecne w PIN'ie Drugi etap polega na wypróbowaniu wszystkich możliwych szpilki, składające się z tych cyfr. Niech D_{origo} będzie tabelą decymzalizacyjną. Dla danej cyfry i , rozważyć binarną tabelę decymalizacyjną D_i z następującymi właściwościami. D_i posiada 1 w pozycji x wtedy i tylko wtedy, gdy D_{origo} ma cyfrę i na tej pozycji. Innymi słowy,

$$D_i[x] = \begin{cases} 1 & \text{jeśli } D_{\text{orig}}[x] = i, \\ 0 & \text{w przeciwnym razie} \end{cases}$$

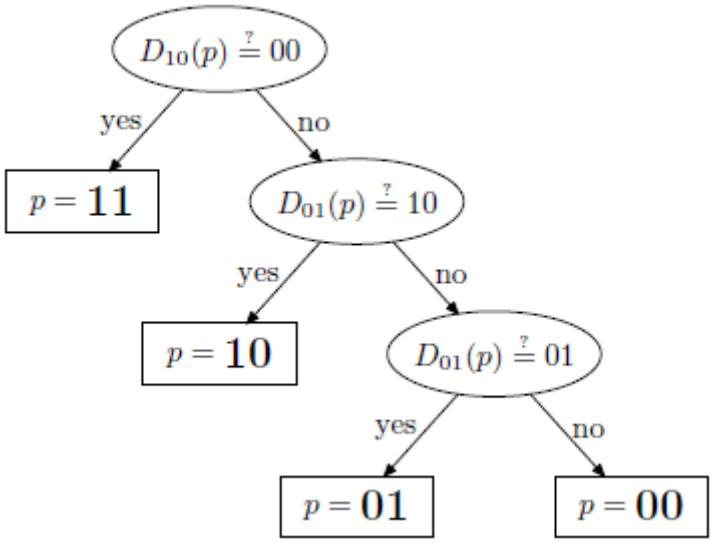
Na przykład, w standardowej tabeli Dorigo = 0123 4567 8901 2345, wartość D3 0001 0000 0000 0100. W pierwszej fazie, dla każdej cyfry i sprawdzamy oryginalny pin przeciwko tabeli decymalizacji Di z próbnym pinem 0000. Łatwo jest zauważyć, że test nie powiedzie się dokładnie wtedy, gdy oryginał pin zawiera i. W ten sposób, przy użyciu tylko co najwyżej 10 odgadnień , ustaliliśmy wszystkie cyfry, które stanowią oryginalny kod PIN. W drugim etapie staramy się każdą możliwą kombinację tych cyfr. Ich rzeczywista liczba zależy od tego jak wiele różnych cyfr zawiera kod PIN. Poniższa tabela podaje szczegóły.

Cyfry	Możliwości
A	AAAA (1)
AB	ABBB (4), AABB (6) AAAB (4)
ABC	AABC (12), Abbe (12) ABCC (12)
ABCD	ABC (24)

Tabela pokazuje, że w drugim etapie co najwyżej 36 odgadnień musi zgadywać (gdy oryginalny PIN zawiera 3 różne cyfry), co daje w sumie 46 odgadnień. Spodziewana liczba prób jest jednak tak mała, jak około 23: 5.

4.2 System adaptacyjny

Proces łamania PIN może być reprezentowany przez przeszukiwania binarnego drzewa. Każdy węzeł V zawiera przypuszczenie, to jest tabela decymalizacyjna D_V i pin p_V. Zaczynamy w węźle głównym i schodzimy z drzewa wzdłuż drogi, która jest określana przez wyniki naszych prób. Niech p_{orig} będzie oryginalnym pinem. W każdym węźle, sprawdzamy, czy $D_V(p_{orig}) = p_v$. Następnie ruszamy w prawo dziecka jeśli tak, i w lewo dziecka, jw. przeciwnym razem. Każdy węzeł v w drzewie może być powiązany z listą P_V oryginalnych kodów PIN w taki sposób, że $p \in P_V$ wtedy i tylko wtedy gdy v jest osiągnięta w procesie opisanym w poprzednim ustępie, jeżeli weźmiemy p jako pierwotnego kodu PIN. W szczególności wymienia związane z rootem, węzły zawierające wszystkie możliwe piny a lista każdego liścia powinna zawierać tylko jeden z elementów: oryginalny PIN p_{orig}. Rozważmy wstępny schemat opisany w poprzednim punkcie, na przykład. Dla uproszczenia zakładamy, że oryginalny pin składa się z dwóch cyfr binarnych i tabela decymalizacyjnej, jest trywialne i mapuje 0→ 0 i 1→ 1. Rysunek poniżej przedstawia drzewo wyszukiwania tych ustawień.



Główną wadą tego systemu jest to, że początkowy liczba wymaganych prób, zależy w dużym stopniu od oryginalnego sworznia p_{orig}. Na przykład, sposób wymaga tylko 9 przybliżeń dla p_{orig} = 9999 (ponieważ, po upewnieniu się, że nie występują w p_{orig} cyfry 0- 8 tylko to jest możliwe), ale w niektórych przypadkach konieczne są 46 odgadnienia. W rezultacie, drzewo wyszukiwania jest dość niezrównoważone, a tym samym nie jest optymalne. Jedna z metod wytwarzania idealnego przeszukiwania drzewa (to drzewo, które wymaga najmniejszych możliwych ilości prób w najgorszym przypadku) jest, aby rozważyć wszystkie możliwe drzewa wyszukiwania i wybrać najlepsze. Takie podejście jest jednak być zbyt nieefektywne ze względu na jego złożoność wykładniczą czasu w stosunku do liczby pinów i możliwych tabel decymalizacji .Okazuje się, że nie wiele tracimy, kiedy zamienimy wyszukiwanie wyczerpujący na prostą heurystykę. Będziemy wybierać wartości D_V i p_v dla każdego węzła w następujący sposób. Niech Pv będzie wykazem pinów związanych z węzłem v. Następnie patrzymy na wszystkie możliwe pary D_V i p_V wybieramy jedną, której prawdopodobieństwo $D_V(p)= p_V \in p_V$ jest zbliżona do 1/2, tak jak to możliwe. Gwarantuje to, że lewa i prawa poddrzewa są w przybliżeniu tej samej wielkości, więc całe drzewo powinno być bardzo wyważone. Program ten można jeszcze poprawić stosując następującą obserwację. Przypomnijmy, że oryginalny pin p_{orig} jest 4-cyfrowa liczba szesnastkowa. Jednak nie musimy go ustalić dokładnie; Wszystko czego potrzebujemy to, aby dowiedzieć się wartość p = $D_{origo}(p_{orig})$. Na przykład, nie musimy być w stanie odróżnić 012D i ABC3 bo dla nich obu, p = 0123. Można łatwo wykazać, że możemy zwiększyć drzewo przeszukiwania, które jest oparty na wartości p_V pod warunkiem, że zamiast p_{orig} zapewnia ,że tabele D_v, nie rozróżniają pomiędzy 0 a A ,1 i B i tak dalej. Na ogół, wymagamy aby każde D_V spełniało następującą właściwość: dla każdej pary cyfr w kodzie szesnastkowym x, y, $D_{origo}[x] = D_{origo}[Y]$, musi oznaczać $D_V[x] = D_V[r]$. Ta nieruchomość nie jest trudne do spełnienia, a w nagrodę możemy zmniejszyć liczbę pinów z 16^4 = 65 536 do 10^4 = 10 000. Rysunek poniższy przedstawia przebieg próbkowania algorytmem oryginalnego PIN p_{orig} = 3491.

No	Possible pins	Decimalisation table D_v	Trial pin p_v	$D_v(p_{\text{orig}})$	$p_v \stackrel{?}{=} D_v(p_{\text{orig}})$
1	10000	1000 0010 0010 0000	0000	0000	yes
2	4096	0100 0000 0001 0000	0000	1000	no
3	1695	0111 1100 0001 1111	1111	1011	no
4	1326	0000 0001 0000 0000	0000	0000	yes
5	736	0000 0000 1000 0000	0000	0000	yes
6	302	0010 0000 0000 1000	0000	0000	yes
7	194	0001 0000 0000 0100	0000	0001	no
8	84	0000 1100 0000 0011	0000	0010	no
9	48	0000 1000 0000 0010	0000	0010	no
10	24	0100 0000 0001 0000	1000	1000	yes
11	6	0001 0000 0000 0100	0100	0001	no
12	4	0001 0000 0000 0100	0010	0001	no
13	2	0000 1000 0000 0010	0100	0010	no

4.3 Schemat adaptacyjny przesunięcia PIN

Kiedy atakujący nie zna żadnego zaszyfrowanego próbnego PIN'u próbnych zaszyfrowane i nie może szyfrować swoich własnych odgadnień, może jeszcze udanie manipulując parametrem offsetu używanym dla zmiany PIN'u klienta. Nasz ostatni schemat ma takie same dwa etapy jak początkowy schemat, więc naszym pierwszym zadaniem jest określenie cyfr występujących w PIN'ie. Zakładamy, że szyfrowany blok PIN zawierający poprawny kod PIN do konta została przechwycona (większość przybywających zaszyfrowanych bloków PIN będzie spełniać to kryterium), a dla uproszczenia, że posiadacz rachunku nie zmieniła swojego numeru PIN i prawidłowe przesunięcie to 0000. Stosując następujący zestaw tabel deymalizacyjnych, atakujący może określić, jakie cyfry są obecne w prawidłowym PIN'ie.

$$D_i[x] = \begin{cases} D_{\text{orig}}[x] + 1 & \text{jeśli } D_{\text{orig}}[x] = i, \\ D_{\text{orig}}[x] & \text{w przeciwnym razie} \end{cases}$$

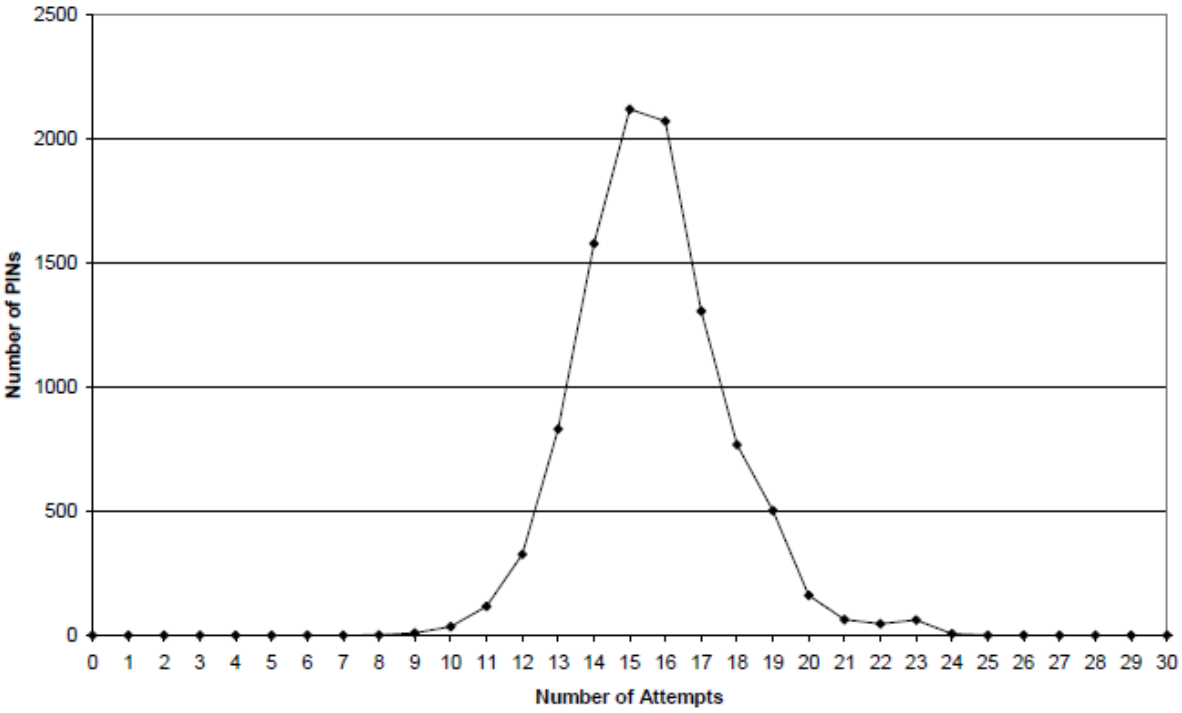
Na przykład, dla D_{origo} = 0123 4567 8901 2345, wartość D3 to 0124 4567 8901 2445. Dostarcza za każdym razem poprawny zaszyfrowany blok PIN i poprawne ustawienie. Podobnie jak w przypadku schematu początkowego, druga faza określa pozycje cyfr obecnych w kodzie PIN i jest ponownie zależny od liczby powtarzających się cyfr w oryginalnym PIN. Weźmy pod uwagę wspólny przypadek, w którym wszystkie cyfry PIN są różne, na przykład 1583. Możemy spróbować ustalić pozycję pojedynczej 8 cyfr, stosując przesunięcie do różnych cyfr i sprawdzając dopasowanie .

Guess Offset	Guess Decimalisation Table	Customer Guess	Customer Guess + Guess Offset	Decimalised Original PIN	Verify Result
0001	0123 4567 9901 2345	1583	1584	1593	no
0010	0123 4567 9901 2345	1583	1593	1593	yes
0100	0123 4567 9901 2345	1583	1683	1593	no
1000	0123 4567 9901 2345	1583	2583	1593	no

Każde inne odgadnięcie pozwala przypuszczać, że klient trafił na nowy kod PIN, który może być zgodny lub nie z oryginalnym kodem PIN po jego deklimalizacji za pomocą zmodyfikowanej tabeli. Tę procedurę powtarza się, dopóki znane jest położenie wszystkich cyfr. Przypadki zawierające wszystkie cyfry będą wymagać co najwyżej 6 transakcji w celu określenia wszystkich danych pozycji. Trzy różne cyfry będą wymagały maksymalnie 9 prób, dwie cyfry do 13 prób, a jeśli wszystkie cyfry będą takie same, żadne próby nie są wymagane, ponieważ nie ma żadnych permutacji. Po złożeniu części schematu przeciętnie wymagane jest 16,5 odgadnień w celu ustalenia danego kodu PIN.

5 Wyniki

Najpierw przetestowaliśmy algorytm adaptacyjny na wszystkich możliwych PIN-ach. Otrzymano rozkład na rysunku.



Najgorszy przypadek zmniejszył się z 45 odgadnięć do 24 ,a średnia spadła z 24 do 15. Następnie zaimplementowaliśmy ataki na IBM Common Cryptographic Architecture (wersja 2.41, dla IBM 4758) i pomyślnie wyodrębniliśmy kody PIN wygenerowane przy użyciu metody IBM 3624. Sprawdziliśmy również ataki na specyfikacje API dla VISA Security Module (VSM) i stwierdziliśmy, że są skuteczne. VSM jest prekursorem całego szeregu modułów bezpieczeństwa sprzętowego do przetwarzania kodu PIN i wierzymy, że ataki będą skuteczne również w stosunku do wielu jego następców.

6 Zapobieganie

Łatwo jest sprawdzić, czy tabela decymalizacji jest ważna. Niektóre metody weryfikacji PIN, które wykorzystują tabele decymalizacyjne, wymagają, aby tabela była zgodna z 0123456789012345, aby algorytm działał poprawnie. W takich przypadkach interfejs API musi jedynie egzekwować ten wymóg, aby odzyskać bezpieczeństwo. Jednak metody weryfikacji numerów PIN, które obsługują własne tabele decymalizacyjne, są trudniejsze do uzyskania . Procedura sprawdzania, która zapewnia odwzorowanie kombinacji wejściowych na maksymalną liczbę możliwych kombinacji wyników, będzie chronić przed

pierwszymi dwoma atakami z tabelą decymalizacyjną, ale nie przed atakiem, który wykorzystuje PIN i ustawia jedynie niewielkie modyfikacje tabela decymalizacji. Aby odzyskać pełne bezpieczeństwo, dane wejściowe tablicy decymalizacyjneh muszą być chronione kryptograficznie, aby można było używać tylko autoryzowanych tabel. Jedyną krótkoterminową alternatywą dla powyższych środków jest zastosowanie bardziej zaawansowanych metod wykrywania włamań i wydaje się, że długoterminowe przesłanie jest jasne: dalsze wspieranie tabel decymalizacji nie jest solidnym podejściem do weryfikacji PIN. Niewykorzystane losowo generowane kody PIN przechowywane w postaci zaszyfrowanej w bazie danych online, takie jak są już używane w niektórych bankach, są znacznie bardziej bezpieczne.

7. Wnioski

Zmodyfikowanie oprogramowania, które współdziała z HSM, jest bardzo kosztowne i podczas gdy aktualizacja oprogramowania HSM jest tańsza, system nadal będzie wymagał testów, a aktualizacja może wymagać kosztownej fazy ponownej inicjalizacji. Proste sprawdzanie poprawności tablic decymalizacyjnych powinno być łatwe do wdrożenia, ale pełna ochrona zachowująca kompatybilność z istniejącym oprogramowaniem mainframe będzie trudna do osiągnięcia. Będzie to zależeć od możliwości wykrywania włamań, o których mówi każdy poszczególny producent. Mamy nadzieję, że w pełni zrozumiemy wpływ tych ataków i optymalne środki zapobiegawcze w najbliższej przyszłości. Chociaż HSM istnieją od dwóch dekad, formalna analiza ich API bezpieczeństwa jest wciąż w powijakach. Trudno dostrzec, że jakakolwiek metodologia uzyskania pewności poprawności może zapewnić wartościowe gwarancje, biorąc pod uwagę różnorodność ataków na poziomie API. Konieczne są dalsze badania nad metodami analizy API, ale na razie musimy przyznać, że pisanie poprawnych specyfikacji API jest tak trudne, jak pisanie poprawnego kodu, i do tradycyjnego wyścigu zbrojeń między atakiem a obroną, że tak wiele produktów oprogramowania ma walczyć.