

Czy oprogramowanie VPN jest bezpieczne?

# Wirtualne (nie)bezpieczeństwo sieci VPN

Bezpieczeństwo wirtualnych sieci prywatnych (ang. VPN – Virtual Private Network) jest często bardziej wirtualne niż rzeczywiste. Pomijając już nawet tzw. „wielką trójcę” (protokoły SSL, SSH i IPsec), wiele aplikacji VPN jest niedopracowanych i posiada poważne wady. W niniejszym artykule spróbujemy wyjaśnić istotę problemu bezpieczeństwa VPN oraz wskażemy zalecane rozwiązania. **PETER GUTMANN**



**O**bcenie istnieje wiele aplikacji VPN – zarówno Open Source, jak i komercyjnych. Jak jednak sprawdzić, które z nich zapewniają prawdziwe bez-

pieczeństwo? W praktyce, aby stworzyć bezpieczną wirtualną sieć prywatną (VPN), potrzeba czegoś więcej niż tylko oprogramowania wykorzystującego kodowanie Blowfish i wymiany kluczy RSA. W dalszej części tekstu opiszemy typowe pułapki czekające na administratorów i analityków projektujących i wdrażających sieci VPN. Podamy też kilka wskazówek i zaleceń dla osób chcących stworzyć własne oprogramowanie VPN lub w większym zakresie korzystać z funkcjonalności, jaką dają istniejące już aplikacje.

Spełnienie trzech tradycyjnych wymagań dla VPN, opisanych w ramce „Wymagania bezpieczeństwa dla sieci VPN,” wciąż jest bardzo trudne, mimo pojawiających się nowych rozwiązań i technologii. Na Rysunku 1 można znaleźć ogólny schemat zastosowania sieci VPN, składającej się z dwóch elementów: „czarnego pudełka” dokonującego uzgadniania parametrów komunikacji (tzw. handsha-

ke) oraz transmisji samych danych. Handshake może być wykonywany przy użyciu dowolnego z protokołów zabezpieczających (TLS, SSH [11], a nawet IKE IPsec [12]), klucze uzyskane na tym etapie służą do szyfrowania przesyłanych danych.

Każdy z pakietów uzyskuje ochronę poufności (warstwa wewnętrzna – przy użyciu algorytmu 3DES), ochronę integralności (warstwa pośrednia – wykorzystująca algorytm MAC-HA1) oraz zabezpieczenie integralności przesyłanych danych (warstwa zewnętrzna wykorzystująca protokół IPsec).

## Poufność danych

Przyjrzyjmy się najczęstszemu błędowi popełnianemu przez osoby projektujące i dokonujące implementacji sieci VPN. Pierwszym przykładem jest, wciąż wykorzystywany przez niektóre rozwiązania VPN, algorytm RC4. Począwszy od MS Windows 3.1 aż do

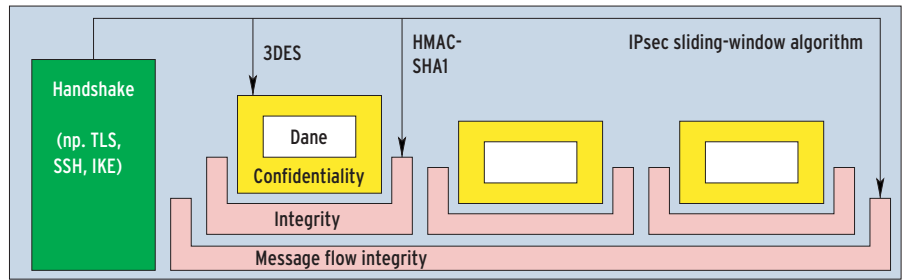
**AUTOR**

Peter Guttman jest pracownikiem wydziału informatyki na Uniwersytecie w Auckland i pracuje nad projektem związanym z analizą kryptograficznych architektur bezpieczeństwa. Pomógł w opracowaniu popularnego pakietu szyfrowania PGP, jest autorem wielu opracowań i referatów dotyczących protokołu RFC oraz ochrony i szyfrowania, m.in. opis certyfikatów X.509, a ponadto jest autorem zestawu narzędzi na cryptlib licencji OpenSource. Napisał też książkę „Cryptographic Security Architecture Design” (Springer-Verlag, 2003). W wolnym czasie Peter szuka błędów w systemach i aplikacjach, które go zainteresują.

późnych lat 90-tych koncern Microsoft korzystał z algorytmu RC4 wszędzie, gdzie to było możliwe, jednak niemal każda z ich implementacji była tak wadliwa, że sam Microsoft zaprzestał stosowania RC4. Jednakże niektóre aplikacje, nawet powstałe poza firmą Microsoft, takie jak ECLiPt <http://freshmeat.net/projects/ecliptsecureunnel/> czy Mirrordir <http://mirrordir.sourceforge.net>, nadal wykorzystują algorytm RC4.

RC4 jest prostym szyfrem strumieniowym generującym liczby pseudolosowe (tzw. key stream), tak pozyskany strumień-klucz jest łączony z tekstem do zaszyfrowania przy pomocy operacji logicznej XOR (różnicy symetrycznej). W efekcie otrzymujemy zaszyfrowane dane. Z kolei, aby rozszyfrować tekst należy wygenerować ten sam strumień-klucz (key stream) i używając go wykonać operację różnicy symetrycznej (XOR) na zaszyfrowanych danych. Jest to bardzo łatwe w implementacji rozwiązanie i dlatego algorytm RC4 stał się tak popularny. Problemem RC4 jest to, że różnica symetryczna (XOR) jest zmienna. Używając strumienia-klucza i zaszyfrowanego tekstu można odzyskać tekst oryginalny. Jednakże porównanie tekstu oryginalnego i tekstu zaszyfrowanego (np. gdy do użytkownika sieci VPN wysłano kilka wiadomości szyfrowanych i nieszyfrowanych i ktoś w tym samym czasie obserwował przepływ danych) może spowodować przechwycenie klucza. Teraz już, używając pozyskanego strumienia, można wykonać operację XOR na zaszyfrowanych danych, a ich treść stanie się jawna.

W pewnym momencie Microsoft próbował zręcznym ruchem „poprawić” swoje aplikacje korzystające z RC4, przechodząc z 32-bitowego klucza RC4 na klucz 128-bitowy. Oczywiście było to błędem, gdyż długość klucza nie

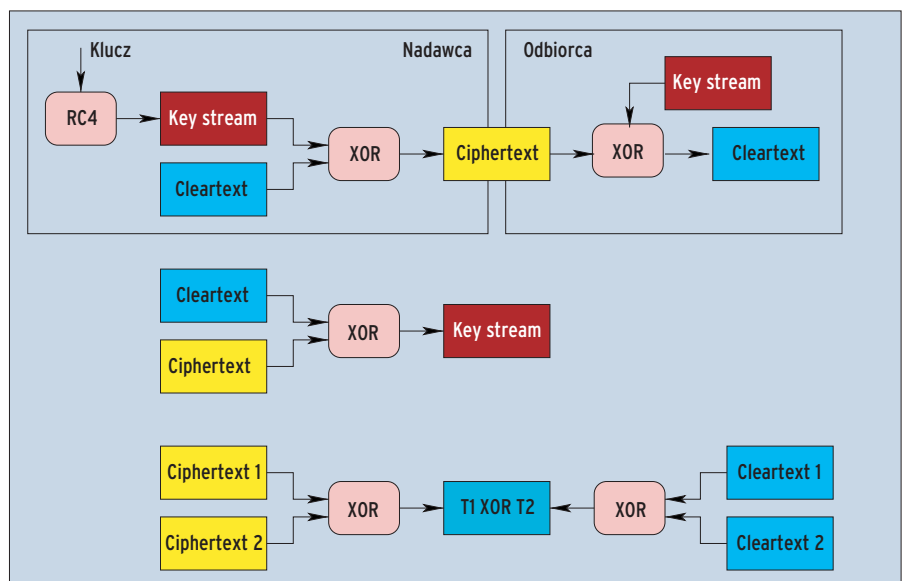


Rysunek 1. Bezpieczna implementacja VPN składa się z fazy handshake (uzgadniania, przy pomocy protokołów TLS, SSH i IKE), która służy do wymiany kluczy 3DES; ochrony integralności (HMAC-SHA1) oraz samego transferu danych.

jest tu istotna – wszystko czego potrzeba, aby odszyfrować zawartość dwóch pakietów, to wykonać na nich operację XOR.

## Proste ataki na RC4

Oto bardzo prosty atak na sekwencję challenge/response, używającą algorytmu RC4:



Rysunek 2. Algorytm RC4 nieprawidłowo używany może być niebezpieczny. Tutaj nadawca generuje strumień klucza i oblicza różnicę symetryczną (XOR) z tekstem jawnym (na górze). Aby uzyskać strumień klucza wystarczy wprowadzić tekst jawny i odczytać tekst zaszyfrowany (w środku). Obliczenie różnicy symetrycznej (XOR) dwóch pakietów zaszyfrowanych spowoduje uzyskanie różnicy symetrycznej tekstów jawnych.

## Wymagania bezpieczeństwa dla sieci VPN

Sieci VPN powinny spełniać wiele wymagań, znacznie przewyższających wymagania stawiane protokołom stosowanym w sieci Internet. Do podstawowych wymagań z zakresu bezpieczeństwa i ochrony danych zaliczamy:

**Poufność:** Atakujący nie powinien być w stanie określić zawartości wiadomości. Zakładając, że nie chodzi wyłącznie o rozszyfrowanie danych, atakujący może wyrządzić szkody znając wyłącznie rozmiar szyfrowanych danych, obserwując wzorce bitów w danych szyfrowanych lub też uzyskując dostęp do chociażby jednego bitu danych.

**Uwierzytelnianie i kontrola dostępu:** Atakujący nie powinien być w stanie dodać żadnych danych mogących udawać dane prawidłowe. Mo-

że to być trudne – atakujący może powtórzyć wiadomość autentyczną, którą należy wykryć.

**Ochrona integralności:** Atakujący nie powinien być w stanie zmodyfikować zawartości wiadomości. Należy tu także pamiętać o ochronie integralności przepływu danych, w których atakujący nie powinien być w stanie umieścić ('Zapłać 10000 dolarów na moje konto'), usunąć ('Patrz, ktoś zmienia nasze wiadomości!') lub zmienić kolejności naszych wiadomości. Pamiętaj, że atakujący nie musi pokonać w zasadzie żadnych zabezpieczeń (poufność danych, uwierzytelnianie, integralność danych), aby wykonać wiele rodzajów ataków modyfikujących przepływ danych w sieci VPN.

**Dostępność:** Istnieje wiele zabezpieczeń

o mniejszych wymaganiach, takich jak ochrona DoS (odmowa usługi). Znajdują się one z reguły poza kontrolą bezpieczeństwa sieci VPN. Zapewnienie ochrony DoS w kodzie zabezpieczeń jest również dobrym pomysłem, ponieważ wiele protokołów bezpieczeństwa wykonuje całkiem sporo operacji szyfrowania, które przerastają możliwości serwera obsługującego duże ilości danych od klientów. Wygenerowanie dużego ruchu w sieci nie jest dla atakującego problemem (będą to po prostu śmieci – skrawki plików itd.), a wymaga ogromnych nakładów na wykonanie operacji typu RSA lub DH na serwerze, tylko po to, aby przetworzone informacje okazały się po prostu niezrozumiałym i niepotrzebnym bełkotem informatycznym.

wystarczy operacja XOR na sekwencji challenge (wywołanie) i zakodowanej odpowiedzi (response), żeby odzyskać klucz szyfrujący. Nawiasem mówiąc, właśnie dlatego twórcy standardu 802.11 wycofali się z próby implementacji RC4 do autoryzacji WEP. Algorytm RC4 ma także inne słabości kryptograficzne (niektóre z nich są znane od wielu lat) – zbyt skomplikowane, aby je tutaj opisywać, ale przy projektowaniu nowych rozwiązań należy zdecydowanie brać je pod uwagę. Ponadto, algorytm RC4 umożliwia atakującemu modyfikację komunikatów w dowolny sposób (opisano to w dalszej części artykułu) – zatem nie zapewnia integralności.

Alternatywą dla szyfrowania strumieniowego jest szyfrowanie blokowe, którego najbardziej znanym przykładem jest szyfrowanie algorytmem DES i jego zmodyfikowaną odmianą – algorytmem potrójnym DES (czyli 3DES). Inne rodzaje szyfrowania blokowego to niedawno opracowany algorytm AES, IDEA (stosowany np. w kluczach PGP 2), Blowfish (patrz książka Bruce'a Schneiera „Applied Cryptography”) oraz CAST (stosowany np. w PGP 5). Wszystkie te algorytmy są bardzo podobne do siebie – algorytm 3DES jest spośród nich najbardziej konserwatywny, najlepiej sprawdzony, używany od dawna i obsługiwany praktycznie przez dużą ilość programów i sprzętu. Jego następca – algorytm AES jest dużo szybszy, ale także dużo młodszy niż 3DES, więc czas nie zweryfikował jeszcze jego użyteczności. Mimo że nowsze algorytmy, takie jak AES, zostały opracowane przede wszystkim z myślą o zwiększeniu prędkości wykonywanych operacji, wszystko jest względne – algorytm 3DES – nawet na starszych komputerach z procesorem 1 GHz – osiąga prędkość około 50MB/s, a algorytm AES – około 100 MB/s. Wydajność oprogramowania Free S/WAN przetestowano i opisano na stronie [1].

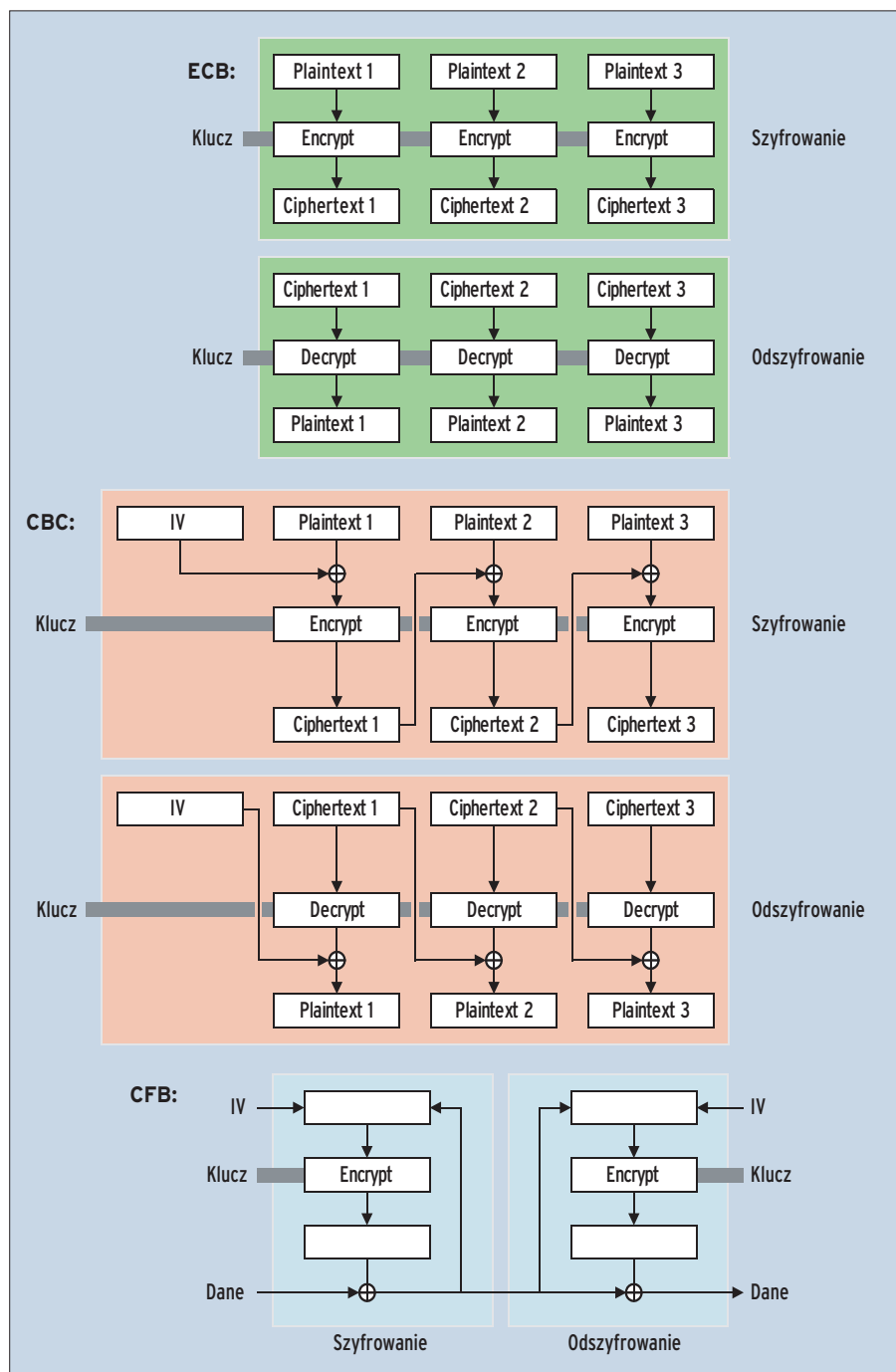
Szyfrowanie blokowe, jak sama nazwa wskazuje, koduje cały blok danych zwykle przy użyciu 64 bitów, chociaż ostatnio coraz popularniejsze staje się kodowanie 128-bitowe. W tej sytuacji użycie szyfrowania blokowego jest trudniejsze niż szyfrowanie algorytmem RC4, gdyż wymaga ono pewnego przygotowania danych wejściowych (bo przecież zazwyczaj nie są one dokładną wielokrotnością słowa 64 lub 128-bitowego). Po standaryzacji algorytmu DES zdefiniowano dla niego kilka trybów działania o różnych właściwościach, takich jak ukrywanie wzorców danych i tryb pracy „bit po bicie”. Jednym z nich jest standardowy tryb ECB (electronic codebook), który szyfruje bloki po 64-bity. Nawiasem mó-

wiąc, każda książka o kryptografii ostrzega przed stosowaniem ECB (patrz opis poniżej), jest on np. używany przez program vtun, znajdujący się na stronie <http://vtun.sourceforge.net>.

## Tryby działania

Jako że każdy blok danych jest szyfrowany niezależnie, określony fragment tekstu jawnego jest zawsze szyfrowany w ten sam sposób, tworząc zawsze ten sam tekst zaszyfrowany. Zatem jeżeli zaszyfrowanie bloku danych ABCD stworzy tekst zaszyfrowany

WXYZ, to pojawienie się w sieci VPN bloku zaszyfrowanego WXYZ będzie dla nas jednoznaczne z faktem, że po rozszyfrowaniu blok ten będzie miał postać ABCD. W ten sposób, metodą prób i błędów, można poznać fragmenty przesyłanych informacji innych użytkowników sieci bez znajomości klucza (tak samo jak w przypadku szyfrowania haseł w systemie Unix bez modyfikatora, czyli tzw. salt). Co gorsza, jako że bloki są niezależne, mogą one podlegać atakom polegającym na zwykłym wycinaniu i wklejaniu,



Rysunek 3. Tryby szyfrowania blokowego. Oba tryby, CBC (w środku) i CFB (na dole) tworzą kolejne bloki uzależniając je od poprzednich bloków. Tryb ECB szyfruje każdy blok niezależnie.

co zostało opisane w dalszej części artykułu. Dla specjalisty z dziedziny kryptografii oprogramowanie korzystające z trybu ECB przypomina opisanie w życiorysie znajomości języka Visual Basic i HTML-a jako „umiejętności programistycznych”...

Oczywiście istnieją inne tryby szyfrowania, które nie powodują takich problemów. Jednym z najczęściej stosowanych jest tryb łańcuchowy (CBC). Tryb CBC (i inne tryby takie jak CFB, czyli sprzężenie zwrotne) uzależnia blok n od bloku n-1 i wykorzystuje losowy blok -1 (tzw. wektor początkowy zwany także IV) do zwiększenia „losowości” pierwszego bloku. Wszystkie te tryby pokazano na Rysunku 3. Oznacza to, że nawet powtarzanie szyfrowania tych samych danych tym samym kluczem (ale z wektorem innym niż początkowy) będzie powodowało uzyskiwanie za każdym razem innego wyniku – dzięki temu proste ataki na tryb ECB nie będą skuteczne, chociaż nie będą niemożliwe. Wynika to ze sprzecznego z intuicją „paradoksu dnia urodzin” – dla danych szyfrowanych w blokach o długości 64 bitów szansa powtórzenia się dwóch bloków po zaszyfrowaniu  $2^{32}$  bloków jest niemal pewna). Dlatego też algorytm AES korzysta z bloków o wielkości 128 bitów.

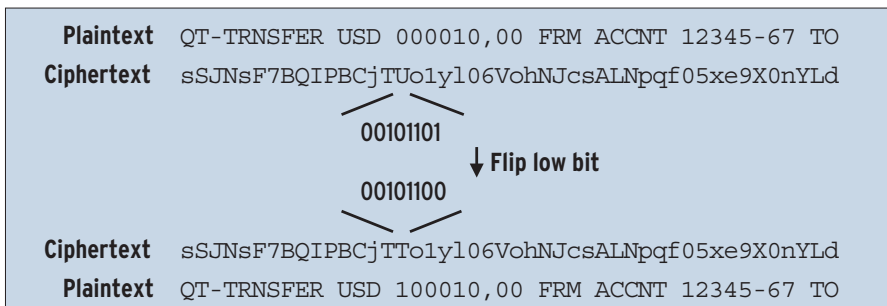
## Słabości algorytmów

Kolejny problem jest znany kryptografom pod nazwą słabości algorytmów – wynika on z rozwoju wiedzy i technologii. Nie jest wprawdzie zbyt często wykorzystywany przez hackerów, jednak w sytuacji wyboru pomiędzy dwoma algorytmami, wybierzemy taki, który będzie wolny od zagrożeń.

### Paradoks dnia urodzin

Ile osób musi się zebrać, aby prawdopodobieństwo tego, że co najmniej dwie spośród nich obchodzą urodziny tego samego dnia, było większe niż 50%? Szanse na to, że ktoś urodził się w tym samym dniu co my, są raczej niewielkie i wynoszą 1/365 (czyli 0,3%). Jednakże szanse kolejnych dwojga osób na urodziny w tym samym dniu rosną o wiele szybciej niż 0,3%, ponieważ każda z nowych osób usuwa swoją datę urodzin z puli dostępnych do wykorzystania dat.

Oznacza to, że zakres możliwych wyborów spada z 1/365 do 1/364 a następnie do 1/363 i tak dalej. Po zebraniu grupy 23 osób szansa na to, że co najmniej dwie spośród nich obchodzą urodziny tego samego dnia, wynosi już powyżej 50%. Ten paradoks ma zastosowanie w bezpiecznych wyborze wielkości bloków do szyfrowania blokowego oraz w algorytmach kodu MAC.



Rysunek 4. Odwrócenie jednego bitu zaszyfrowanych algorytmem RC4 danych, może mieć istotny wpływ: zamiast 10 dolarów, fundusz przeleje na nasze konto 100010 dolarów.

W przyszłości ktoś przecież może znaleźć zastosowanie i sposób wykorzystania tego problemu. Wiele z ataków na protokoły szyfrowania danych na świecie w ostatnich latach wynikało ze słabości algorytmów. Z tego powodu projektanci protokołów wolą zabezpieczać protokoły bezpieczeństwa nadmiernie, aby oparły się wszystkim atakom, jakie tylko przyjdą do głowy kryptografom: chcą tak zabezpieczyć dany protokół, żeby po roku lub dwóch nie istniała konieczność wnoszenia poprawek do projektu, bo właśnie ktoś opracował sposób złamania zabezpieczeń i może go praktycznie zastosować.

Oczywiście istnieje także kilka innych potencjalnych problemów, także związanych z wektorem początkowym (IV). Aby wektory początkowe były skuteczne, muszą być po prostu używane (popularne aplikacje VPN takie jak TunnelVision <http://open.nit.ca/wiki/?page=TunnelVision>, a także Zebedee <http://www.win-ton.org.uk/zebedee/> nie robią tego). Ponadto wektor początkowy musi być wybrany losowo i być nieprzewidywalny dla atakującego [2] [3]. Protokoły SSL i SSH wykorzystywały ostatni blok pakietu n-1 jako wektor początkowy dla pakietu n, co oznacza, że atakujący, który odczyta pakiet n-1, może już zawczasu określić wektor początkowy dla pakietu n. Kolejna wersja protokołu SSL o nazwie TLS 1.1 usuwa ten problem (programiści protokołu IPsec znali ten problem od roku, więc w protokole IPsec jest on rozwią-

zany [4]). Jest to kolejna słabość algorytmu – jak dotąd nikomu nie udało się opracować sposobu wyrządzenia poważnych szkód w ten sposób, ale nie oznacza to, że możemy ignorować ten problem.

## Integralność wiadomości

Niektórzy programiści mylnie zakładają, że szyfrowanie danych zapewnia ich integralność. Nic bardziej błędnego – dość łatwo można dowolnie zmodyfikować zaszyfrowane dane bez znajomości klucza szyfrującego. Rozważmy przykład z opisanym w poprzedniej części algorytmem RC4, używanym w programach ECLiPt, mirrordir i wielu aplikacjach firmy Microsoft. W przykładzie pokazanym na Rysunku 4 zamiana jednego bitu zmienia całkowicie znaczenie wiadomości – zamiast przesłania 10 dolarów na nasze konto, fundusz dokonałby przelewu na znacznie większą kwotę.

Zmiana bitu w innych rodzajach szyfrowania strumieniowego (np. CFB) jest również prosta, ponieważ wiele sieci VPN korzysta z algorytmu RC4 lub trybu CFB bez zabezpieczenia integralności danych.

## Wycinanie i wklejanie

W poprzedniej części wspomnieliśmy o problemach z trybem ECB użytym w programie vtun. Na Rysunku 5 pokazano sposób, w jaki można złamać bezpieczeństwo sieci

Deposit	\$10,000	in acct.	number	12-3456-	789012-3
H2nx/GHE	KgvldSbq	GQHbrUt5	tYf6K7ug	S4CrMTvH	7eMPZcE2
H2nx/GHE	5guZEHVr	GQHbrUt5	tYf6K7ug	Pts21LGb	a8oaNWPj
H2nx/GHE	5guZEHVr	GQHbrUt5	tYf6K7ug	S4CrMTvH	7eMPZcE2

Rysunek 5. Szyfrowanie blokowe w trybie ECB jest podatne na proste ataki wycinania i wklejania. Wystarczy skopiować numer konta ze standardowej transakcji (na górze) do transakcji przeprowadzanej przez kogoś innego (na dole), aby pieniądze dotarły na inne konto.

VPN wykorzystującej np. algorytm 3DES – oczywiście przy założeniu, że nie znamy klucza szyfrującego.

Zacznijmy od tego, że w 3DES każdy blok jest szyfrowany niezależnie. Na początek wykonaliśmy transakcję przelewu pieniędzy na serwerze bankowym. Obserwując ruch w sieci VPN, możemy określić jak została zaszyfrowana nasza informacja o przelewie. Później, w ciągu dnia zauważyliśmy kolejną zaszyfrowaną wiadomość, tak więc kopiując bloki zawierające nasz numer konta wkleiliśmy go do tej wiadomości – w ten sposób pieniądze zostały przelane na nasze konto, mimo że nie znaliśmy klucza szyfrującego.

Nie jesteśmy pewni w tym miejscu, jakiego wpływu gotówki mamy się spodziewać, gdyż informacja ta jest zaszyfrowana. Musimy więc poczekać do momentu pojawienia się wpływu na naszym koncie. Jeżeli kwota nie będzie nas satysfakcjonować – zawsze możemy powtórzyć opisaną operację dowolną ilość razy.

Co jednak zrobić, gdy użyty został tryb szyfrowania CBC, w którym każdy blok jest uzależniony od bloku poprzedniego? Niektóre sieci VPN wykorzystują tryb CBC lub CFB w zastępstwie trybu ECB dla uniknięcia słabości tego rozwiązania. Faktycznie tryby te są lepsze niż ECB, ale tylko nieznacznie. Tryby CBC i CFB mają bardzo pożyteczną cechę – ponownej synchronizacji, dzięki której zmiana w jednym bloku wpłynie na blok kolejny, ale dla trzeciego bloku szyfrowanie zostanie przywrócone do stanu początkowego. Oznacza to, że mimo użycia lepszych trybów szyfrowania, nadal możemy wykonać próbę wycinania i wklejania fragmentów bloków, opisaną wcześniej. Tryb CFB umożliwia ponadto niewykrywalną modyfikację ostatniego bloku w taki sam sposób, jak to miało miejsce z algorytmem strumieniowym RC4.

## Zapewnienie bezpiecznej integralności szyfrowanych danych

Jedynym sposobem zapewnienia prawdziwej integralności danych jest badanie integralności w sposób klarowny. Poza kilkoma rozwiązaniami, które pojawiły się ostatnio, żaden z opatentowanych i trudnych trybów szyfrowania nie zapewnia integralności zaszyfrowanych danych.

Pierwszym stopniem zapewnienia integralności danych byłoby dodanie do wiadomości prostej sumy kontrolnej, takiej jak CRC (część sieci VPN nie stosuje nawet tak prostego zabezpieczenia...). Jednym z programów korzystających z sumy kontrolnej CRC jest CIPE

<http://sites.inka.de/sites/bigred/devel/cipe.html> (podobnie jak SSHv1). Niestety nie jest to rozwiązanie bezpieczne – sumy kontrolne CRC są wystarczające przy wykrywaniu przypadkowych niecelowych modyfikacji danych, ale okazują się kompletnie bezużyteczne do wykrywania modyfikacji celowych – można przecież utworzyć dane o dowolnej sumie kontrolnej CRC, także o sumie kontrolnej wiadomości oryginalnej. Ponadto, nic nie pomoże tutaj zaszyfrowanie samej sumy kontrolnej CRC. Więcej informacji o tym można znaleźć w opracowaniu Core SDI z 1998 roku [5] (przed opracowaniem SSHv1 użycie CRC-32 było w rzeczywistości słabością algorytmu w Kerberos). Ta słabość doprowadziła do wielu ataków na SSHv1, a w rezultacie odrzucenie tego protokołu przez użytkowników.

## Ochrona integralności przy użyciu kodu MAC

Aby zapewnić odpowiednią integralność wiadomości, musimy użyć specjalnej sumy kontrolnej, zwanej kodem autoryzacji wiadomości (MAC). Protokół cargo-cult stosowany w IPsec używa 96 bitów kodu MAC, gdyż dzięki temu nagłówek AH ma odpowiednią wielkość – jego długość wynosi dokładnie 128 bitów (cztery słowa 32-bitowe) – niektórzy podejrzewali, że liczba 96 ma jakieś magiczne znaczenie i kopiowali tę wartość do swoich projektów i rozwiązań.

Protokoły SSL/TLS i SSHv2 wykorzystują pełne 160-bitowe kodowanie MAC. Vpnd <http://sunsite.dk/vpnd/> także wykorzystuje kodowanie MAC, ale używa tylko 16 bitów. Według teorii paradoksu dnia urodzin, pierwszego powtórzenia kodu MAC można spodziewać się już po upływie pół sekundy

(przy łączu szerokopasmowym T1). Jako opcjonalną alternatywę dla kodu MAC, vpnd opracowuje własny 16-bitowy rodzaj sumy kontrolnej, bardzo osobliwy:

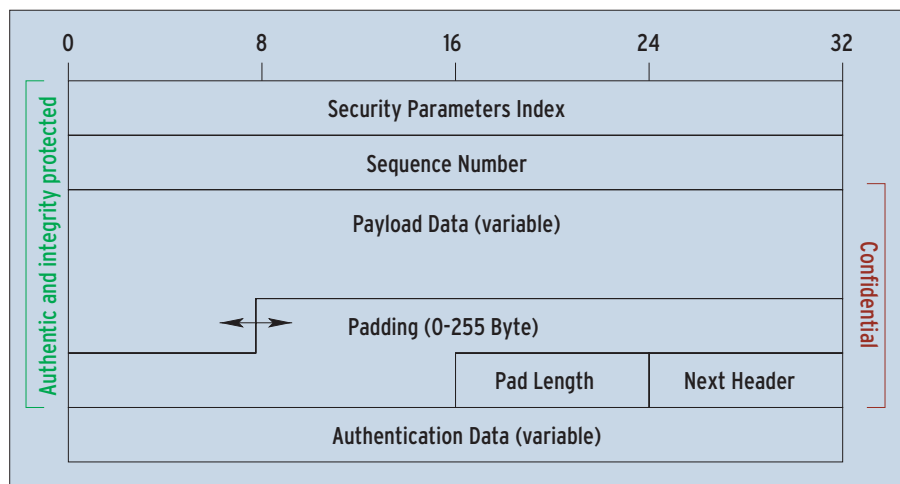
```
while(length--)  
    sum=((sum<1)|((sum>15)&1))  
    ^*data++;
```

Ma on bardzo słabe właściwości kryptograficzne, suma kontrolna powtarza się po 16 bitach danych, a po kolejnych 16 bitach liczni są zerowane i powtarzane).

## Integralność przepływu danych

Nawet jeśli zdołamy odpowiednio zabezpieczyć wiadomość przy pomocy szyfrowania i ochrony integralności, potencjalny atakujący może nadal spowodować chaos poprzez wstawienie, powtórzenie, usunięcie lub zmianę kolejności pakietów z chronionymi danymi. Prawie żadna z wirtualnych sieci prywatnych (VPN) nie zapewnia bezpieczeństwa przed tego typu atakami. Problem ten dotyczył także wczesnych wersji IPsec, które nie posiadały dobrej ochrony integralności przepływu danych, gdyż był to tunelowany ruch IP, a protokół IP sam radzi sobie z założenia niewiarygodnymi pakietami.

Jednakże, podobnie jak to miało miejsce w przypadku sum kontrolnych CRC (sprawdzają się przy przypadkowych modyfikacjach danych, ale są bezużyteczne przy wykrywaniu celowych zmian danych), jest wiele sposobów na przygotowanie celowych „błędów” i wykorzystanie faktu, że protokoły po prostu nie zostały przygotowane do radzenia sobie z celowymi „błędami” transmisji. Steve Bellovin (je-



Rysunek 6. Nagłówek ESP (protokół IPsec) przedstawia pewne istotne dla bezpieczeństwa właściwości. Numer sekwencji jest wykorzystywany do ponownej ochrony, a dopełnienie maskuje prawdziwą długość wiadomości.

## Kilka wniosków

Opracowanie bezpiecznego protokołu sieci VPN jest naprawdę bardzo trudne. Opracowanie protokołu IPsec zajęło najlepszym na świecie kryptografom i inżynierom od spraw zabezpieczeń wiele lat, ponieważ za każdym razem, gdy już myśleli, że projekt jest gotowy, ktoś opracowywał nowy rodzaj ataku na ich dzieło.

W pewnych przypadkach, aby ocenić rzeczywistą wartość bezpieczeństwa IPsec, musiano stworzyć całkowicie nowe kryteria i techniki analiz. Podczas tego procesu tworzenia napotkano na problemy, o których większość nawet nie śniła. Poradzono sobie jednak i końcowy projekt był odporny na wszelkie ataki.

W wyniku pracy nad protokołami takimi jak IPsec, SSL i SSH w ciągu ostatnich lat zmieniły się znacznie standardy protokołów uwierzytelniania i wymiany danych. Szczególnie trudne do wykonania jest opracowanie uzgadniania klucza uwierzytelnionego (samo opracowanie jest proste, trudne jest dobre opracowanie uzgadniania klucza uwierzytelnionego). Nawet najprostsze modele zabezpieczeń podlegają nadal aktywnym badaniom na świecie.

Powodem, dla którego protokoły typu IPsec (bez części IKE), SSL i SSH znajdują się w tak dalekim stadium rozwoju jest fakt, że są to jedyne tak niewielkie protokoły, które są nadal bezpieczne.

den z twórców protokołu IPsec) przygotował referat zawierający zestawienie wszystkich problemów [6]. Wkrótce błędy zostały usunięte z protokołu IPsec, ale inne przytoczone wcześniej aplikacje nadal borykają się z nimi.

## Błędne założenia

Pojawia się także problem związany z oczekiwaniami użytkownika, który spodziewa się, że używanie sieci VPN z pełnym zestawem zabezpieczeń, daje mu większą gwarancję co do jakości obsługi niż kanał niechroniony. Rzeczywiście głównym powodem stosowania sieci VPN jest to, że zapewnienia dodatkowe gwarancje bezpieczeństwa przesyłanych danych w stosunku do standardowych protokołów TCP/IP. Oczywiście użytkownik może tunelować coś innego niż IP (np. projekt vpe <http://savannah.nongnu.org/projects/vpe> tuneluje ramki ethernetowe), nie podejmując nawet próby ochrony przepływu danych.

W takim przypadku twórca sieci VPN oczekuje, że użytkownik zapewni sobie odpowiednią ochronę danych na poziomie protokołu wymiany danych, a użytkownik oczekuje od twórcy sieci VPN, że ta zapewni mu bezpieczeństwo na wszystkich poziomach modelu OSI. Rozwiązanie tego problemu jest bardzo proste: wewnątrz czegoś, co można określić bezpieczną kopertą (Rysunek 6), należy umieścić numery sekwencji i polecić odbiorcy wiadomości ułożenie odebranych pakietów w podanej w niej kolejności (lub odpowiednio odrzucić powtórzone pakiety i wysłać żądanie ponownego wysłania brakujących, itd.). Dla protokołu TCP jest to wystarczający sposób (pakiety muszą docierać w odpowiedniej kolejności, więc jakiegokolwiek powtórzenia, brak pakietów lub inna ich kolejność oznaczają

atak z zewnątrz), problem pojawia się jednak przy zawodnym z natury protokole UDP.

Protokół IPsec rozwiązuje ten problem przy pomocy algorytmu sliding-window [7], który zapewnia odpowiednią niezawodność na poziomie protokołu IPsec. Alternatywą mogłaby być warstwa niezawodności protokołu UDP, a następnie potraktowanie go jak zwykłego łącza TCP ze standardowym zabezpieczeniem numerów sekwencji pakietów, opisanym powyżej. Oba opisane sposoby wymagają przemyślenia (i ostrożnego programowania).

## Niekontrolowany kanał kontrolny

Jednak wiele sieci prywatnych VPN, z protokołem innym niż IPsec, steruje danymi nie całkiem bezpiecznymi tunelami, w których dodatkowo kanał kontrolny nie obsługuje tunelowania IP – tak więc nie ma tutaj nawet minimalnej ochrony zapewnianej na poziomie IP.

Dane kontrolne sieci VPN są bardziej podatne na różnego typu ataki niż dane zawierające wiadomość, gdyż udany atak na wiadomość z kanału kontrolnego może narazić na

niebezpieczeństwo każdą kolejną wiadomość przesłaną kanałem danych. Właśnie z tego powodu protokoły SSL, SSH i IPsec wykorzystują pełną ponowną renegocjację protokołów, aktualizując parametry szyfrowania – nie przesyła się po prostu kolejnego klucza lub parametru ochrony tym tunelem. Jako że odpowiedzią na wiadomości z nieprawidłowymi sumami kontrolnymi jest ich odrzucenie, możliwe jest pozostawienie wiadomości o zmianie klucza (wymuszając ciągłe używanie narażonego na atak klucza), odwracając jeden lub dwa bity. Bez ochrony odpowiedzi atakujący może nawet podawać odpowiedzi zawierające polecenia powrotu do starych kluczy, wymuszając w ten sposób używanie złamanego klucza.

Większość aplikacji sieci VPN nie ukrywa długości wiadomości przy pomocy losowego dopełnienia, umożliwiając atakującemu szybką identyfikację najważniejszych pakietów zarządzających i wykonanie opisanych powyżej czynności. Ukrywanie długości wiadomości może także zapobiegać identyfikacji innych wiadomości o stałej długości, takich jak ACK TCP, żądań ARP oraz wiadomości DHCP lub identyfikacji zaszyfrowanych wiadomości/odpowiedzi ICMP, odszyfrowaniu zawartości i wprowadzeniu własnych wiadomości ICMP [8]. W celu ukrycia informacji o kluczowych pakietach danych (np. zawierających zaszyfrowane hasła), protokoły SSH przesyłają te dane za pomocą pakietów o stałej długości, uniemożliwiając w ten sposób określenie długości wiadomości (można też wygenerować odpowiednio duży ruch w sieci, ukrywając w ten sposób obecność pakietów zawierających istotne informacje).

## Wyłączenie szyfrowania

Kolejnym rodzajem ataku na kanał kontrolny (który jest w zasadzie atakiem na integralność wiadomości, a nie na przepływ danych) jest działanie na aplikacje sieci VPN, które przenoszą niewystarczająco chronione klucze

## Gotowe rozwiązania VPN

Jeżeli potrzebujesz gotowego oprogramowania sieci VPN, najlepszym rozwiązaniem jest implementacja IPsec, Free S/WAN [12] <http://www.freeswan.org/>. Niestety użytkownicy tego oprogramowania twierdzą, że jest ono dość trudne w użyciu, trudniejsze niż inne aplikacje IPsec. Pomóc w tej dziedzinie może nowa funkcjonalność IPsec umieszczana w jądrach Linuksa.

Inne aplikacje godne polecenia, które nie napotkały od razu problemów z ochroną

danych to OpenVPN <http://openvpn.sf.net/> i Yavipin <http://yavipin.sf.net/>. OpenVPN zbudowano w oparciu o protokół SSL (kanał kontroli, zastępujący IKE protokołu IPsec) i protokół ESP (kanał danych, na podobieństwo protokołu IPsec). Yavipin jest produktem całkowicie odrębnym, ale stworzony został we właściwy sposób i wydaje się być prawidłowo napisany. Oba programy są przyjazne dla użytkownika i nie mają wiele wspólnego ze złożonością programowania IPsec.

## Przygotowanie własnego oprogramowania VPN

Jeżeli naprawdę musisz uruchomić własne oprogramowanie dla sieci VPN lub dodać funkcjonalność takiego oprogramowania do istniejącej aplikacji, utwórz kanał kontroli przy pomocy protokołu SSL i SSH, a kanał danych przy pomocy opracowań wydanych przez programistów protokołu IPsec (na takiej zasadzie powstała i działa z powodzeniem oprogramowanie OpenVPN). Materiały źródłowe dotyczące SSL, SSH i innego pomocnego oprogramowania znajdziesz na stronach: OpenSSL <http://www.openssl.org> oraz OpenSSH [11] <http://www.openssh.org/> lub innych, takich jak cryptlib <http://www.cs.auckland.ac.nz/~pgut001/cryptlib/>.

Część IPsec można znaleźć w opracowaniu Kame (w nowszych jądrach Linuksa) lub użyć np. `ipsec_tunnel` ze strony [http://ringstrom.mine.nu/ipsec\\_tunnel/](http://ringstrom.mine.nu/ipsec_tunnel/). `ipsec_tunnel` jest wyjątkowo niewielki (tylko 38KB kodu źródłowego), ale brakuje mu pewnej funkcjonalności w zakresie ochrony powtórzeń, którą należy dołożyć do swojego oprogramowania, aby zapewniało kompletną ochronę.

Na stronie internetowej Gmane <http://news.gmane.org/gmane.network.vpn.theory> toczy się dyskusja na temat zdefiniowania standardów wirtualnych sieci prywatnych (VPN) opartych na protokole IPsec.

szyfrujące w pakietach. Najbardziej oczywisty atak to zamiana bitów w taki sposób, aby uzyskać klucz zerowy (lub inny znany czy też możliwy do przewidzenia klucz). Jednakże możliwe jest wykonanie prostszego ataku, który w zasadzie wymaga tylko umiejętności zamiany pojedynczego bitu. Nie jest istotne jaka będzie wartość tego bitu – byle była to wartość różna od oryginalnie wysłanej w wiadomości. Wykonanie takiego działania możliwe jest na algorytmie DES lub 3DES. Szyfr ten ma taką właściwość, że bity kluczowe są badane pod kątem parzystości, co było przedmiotem dyskusji we wczesnych latach siedemdziesiątych, kiedy opracowywano algorytm DES. Zgodnie z właściwościami tego algorytmu, nie musimy używać klucza, jeżeli bity parzystości są błędne, gdyż oznacza to uszkodzenie klucza podczas transmisji.

Atak zwykle się udaje, ponieważ wiele aplikacji sieci VPN nie zajmuje się sprawdzaniem wartości zwrotnych dla funkcji szyfrowania, zakładając że są one zawsze prawidłowe. Do tego typu aplikacji należą: TunnelVision, vpe oraz Zeebedee. Jedynie, co trzeba zrobić

w przypadku tych aplikacji, to zamienić bit w zaszyfrowanym pakiecie klucza. Oprogramowanie VPN podejmie próbę załadowania klucza – oczywiście ładowanie nie uda się z powodu niewłaściwej wartości bitu parzystości. Jako że oprogramowanie nie sprawdza, czy ładowanie klucza zakończyło się powodzeniem, dane są przesyłane z kluczem o wartościach zerowych. Nie jest to wyłącznie zwykły przypadek niewłaściwego programowania: każda dobrze przygotowana aplikacja, dbająca o bezpieczeństwo naszych danych nie powinna nigdy ładować podejrzanego klucza, takie manipulacje powinny być automatycznie wykryte. Istnieje wiele sposobów ataków na klucze i związane z nimi kanały kontroli, atak na klucz w algorytmie DES jest po prostu najprostszy do naszych celów prezentacji.

## Inne problemy

Do tej pory omawialiśmy tylko podstawowe przetwarzanie danych sieci VPN i kanału kontroli, nie zajmowaliśmy się zaś procesem negocjacji (handshake) i procesem autoryzacji. Powodem tego jest wyjątkowo skomplikowana natura tego tematu (patrz ramka „Zalecane pozycje do przeczytania”), która zwykle pochłania ogromne ilości rozdziałów niektórych książek, a tutaj nie możemy sobie na to pozwolić. Nasze zadanie jest tym trudniejsze, że większość implementacji sieci VPN stosuje swoje własne mechanizmy negocjowania transmisji, których analizę należałoby rozpocząć od kodu źródłowego.

Ponadto wiele zastosowań obnaża nie tylko słabe punkty protokołu szyfrującego, ale także różne błędy podczas wdrażania, jak np. generowanie kluczy szyfrujących z identyfikatorem procesu i czasem lub przez polecenie `rand()`, a także niewystarczającą kontrolę lub całkowi-

ty brak kontroli dla wartości spoza zakresu i błędów zakresu (także w implementacji PPTP firmy Microsoft występował ten problem – serwer zawieszał się, bo kod nie wykonywał zbyt dokładnie kontroli danych wejściowych [9]). Spisanie wszystkich występujących problemów stworzyłoby pokaźną księgę życzeń i zażaleń. Ponadto nigdy do końca nie możemy być pewni, czy jest to problem spowodowany samym projektem protokołu (nieudokumentowanym błędem lub efektem ubocznym), czy też jest to po prostu błąd implementacji.

## Prędkość transmisji a bezpieczeństwo

Niektóre wirtualne sieci prywatne (VPN) oferują wiele trybów działania, z których jeden jest przygotowany pod kątem zapewnienia maksymalnej ochrony, inny zaś preferuje prędkość transmisji kosztem zmniejszenia bezpieczeństwa przesyłania danych (jest to całkowicie niepotrzebne, gdyż w większości sieci VPN rezygnuje się z maksymalnego poziomu zabezpieczeń). Od tej chwili będziemy określali tryb preferujący prędkość transmisji ponad bezpieczeństwem danych jako tryb „Zaatakuj mnie teraz!”, ponieważ nikogo nie zainteresuje tryb maksymalnego bezpieczeństwa, gdy pod ręką będzie miał tryb „Zaatakuj mnie teraz!”.

Oto sposób przeprowadzenia ataku: kiedy ofiara ustawi tryb maksymalnego bezpieczeństwa sieci VPN, atakujący wykorzystuje 1001 technik ataku typu odmowa usługi (DoS). Po przełączeniu w tryb „Zaatakuj mnie teraz!” (z powodu niemożności pracy) ataki DOS są wyłączone. Szczęśliwa jest zarówno ofiara jak i atakujący. Jest to standardowy mechanizm ataku wykorzystywany przeciwko najróżniejszym rodzajom uwierzytelniania w systemie

## Pozycje zalecane do przeczytania

Dobrym źródłem informacji na temat spraw związanych z ochroną i szyfrowaniem danych, które dodatkowo opisuje mechanizmy zabezpieczeń jest „Network Security: Private Communication in a Public World”, której autorem są Charlie Kaufman, Radia Perlman i Mike Speciner.

„Practical Cryptography”, której autorem są Bruce Schneier i Neils Ferguson zawiera szczegółowy opis projektowania protokołu zbliżonego do SSL/SSH. „Lessons Learned in Implementing and Deploying Crypto Software”, a także moja strona internetowa <http://www.cs.auckland.ac.nz/~pgut001/> skupia się nad pewnymi problemami, które pojawiają się

w momencie, gdy użytkownicy wykorzystują bezpieczne bloki szyfrowania w niezbyt bezpieczny sposób. Niestety dodatkowe informacje na temat protokołu IPsec w sieciach VPN to towar deficytowy – RFC nie zawiera prawie wcale informacji dla programistów, a większość książek dotyczących tej problematyki jest po prostu przedrukiem lub parafrazą RFC. Bogactwo wiedzy o sieciach VPN znajduje się w archiwum listy dyskusyjnej na stronie <http://www.vpnc.org/ietf-ipsec/>. Jest również wiele materiałów dodatkowych, ale głównym z nich jest „The SIGMA Family of Key-Exchange Protocols” <http://www.ee.technion.ac.il/~hugo/sigma.html>.

Windows, w którym nie trzeba przeprowadzać ataku na nowe mechanizmy zabezpieczające sieć, bo prościej przekonać ofiarę, aby użyła trybu z ery Windows 3.1, który jest całkowicie pozbawiony ochrony (protokoły takie jak protokół SSL zawierają wbudowaną ochronę przeciwko takim problemom).

Ataki tego typu były zgłaszane w sieci SWIFT (Ogólnoświatowe Towarzystwo Transferu Funduszy Międzybankowych, wykonujące operacje bankowe o wartości 6 bilionów dolarów dziennie) w latach osiemdziesiątych, gdy urzędnicy kodujące działały doskonale na złączach testowych, ale wysłanie jakichkolwiek danych przez rzeczywiste łącza kończyło się utratą pakietów i uszkodzeniem danych. Po kilku tygodniach prób bank zrezygnował z tego trybu przesyłu i przesłał transakcje finansowe w czystej, nieszyfrowanej formie i problemy z przesłaniem danych przestały istnieć.

## Kwestia czasu

Czasami problemy, z którymi musimy sobie poradzić, są wyjątkowo subtelne. Przykładowo, dwa oddzielne bezpieczne podsystemy mogą po połączeniu stworzyć jeden wspólny system nie spełniający wymogów bezpieczeństwa i nie posiadający zabezpieczeń swoich dwóch podsystemów (taka sytuacja jest koszmarem dla projektantów protokołów i obszarem, na który poświęca się najwięcej wysiłku). Odpowiednio zaprojektowane i prawidłowo użyte wirtualne sieci prywatne (VPN) zapewniają odpowiednią ochronę integralności i poufności danych. Jednakże po ich połączeniu, sieci takie mogą powodować spore problemy.

Typowym sposobem ochrony indywidualnych pakietów jest zabezpieczenie ich integralności kodem MAC i zaszyfrowanie całości, zapewniające ochronę poufności. Aby przeprowadzić udany atak, należy zmodyfikować wiadomość podczas przesyłania i obserwować, co się stanie po jej dotarciu do punktu docelowego. Jeżeli wiadomość zostanie odrzucona względnie szybko, to oznacza, że problem został wykryty podczas rozszyfrowywania. Jeżeli odrzucenie wiadomości zajmie więcej czasu – problem został wykryty w fazie deszyfracji kodu MAC. Umożliwia to atakującemu różnicowanie swoich ataków (ataki na poufność wiadomości i ataki na integralność MAC).

Obecnie jest to (w większości przypadków) słabość algorytmów, co jednak nie oznacza, że ktoś nie odkryje prawdziwych słabości tego rozwiązania w dziesięć minut po przeczytaniu tego artykułu (jeżeli tak się stanie, pamiętajcie, że czytaliście o tym właśnie w naszym cza-

sopiśmie). Co gorsza, ktoś mógłby opracować sposób wykorzystania tej wiedzy za kilka lat, kiedy odpowiednie oprogramowanie będzie szeroko stosowane.

Jeżeli zaistnieje taka sytuacja dla protokołów takich jak SSL czy SSH, wydane zostaną zalecenia CERT, dzięki którym programiści i sprzedawcy protokołów SSL/SSH szybko wprowadzą niezbędne poprawki do swoich produktów. Jednakże, dla niektórych protokołów [10] nie ma już ratunku i mimo, że ten szczególny problem nie dotyczy protokołu SSH, autorzy protokołu SSL już pracują nad dodatkowym zabezpieczeniem ich produktu.

Jeżeli jednak taka słabość zostanie wykryta w innych protokołach, prawdopodobnie nikt się nigdy o tych słabościach nie dowie – kryptografowie zajmują się przede wszystkim najchętniej stosowanymi i najbardziej bezpiecznymi protokołami. Tak więc na tym przykładzie widzimy, że aplikacje sieci VPN są podatne nawet na najbardziej proste ataki z zewnątrz, co pokazuje jak niebezpieczny może być brak szczegółowej analizy dotyczącej bezpieczeństwa naszych danych.

## Organizm biologiczny

Po pojawieniu się nowego rodzaju ataku, jest on zazwyczaj konfrontowany z wielką trójcą (protokoły SSL, SSH i IPsec) kryptografii. Jednak tak jak organizm biologiczny, tak samo wielka trójca adaptuje się i ewoluuje opierając się nowym rodzajom ataków, mimo że pewne nurty rozwoju tych protokołów (np. protokół SSHv1) w końcu zostaną porzucone.

Dawniej, istniał także protokół SSLv1, ale został złamany już przy wprowadzeniu do użytku. Z drugiej strony, mniej znane protokoły mogą przetrwać w odosobnieniu wiele lat do momentu wystawienia ich na światło dzienne, ale już pierwszy atak może dowieść ich całkowitej bezużyteczności. Nie jest to jedyny powód stosowania standaryzacji projektów.

Przykładowo kod jednej z aplikacji sieci VPN, którą testowałem, wyglądał tak jakby wysłał pierwszy klucz sesji kanałem danych w formie jawnej, przed ustaleniem klucza sesji, po 45 minutach poddałem się po przekonaniu się przez setki wierszy niekomentowanego kodu źródłowego. Takich żenujących błędów nie spotkamy w protokołach wielkiej trójcy, ponieważ każda implementacja, która wysłałaby klucz w formie jawnej, zostałaby szybko zdemaskowana.

Aby ułatwić wybór i proces wdrożenia sieci VPN przygotowaliśmy dwie ramki

„Gotowe rozwiązania VPN” i „Przygotowanie własnego oprogramowania VPN”, w których umieściliśmy ogólne wskazówki dotyczące wyboru lub implementacji własnego programowania wirtualnych sieci prywatnych (VPN). ■

## INFO

- [1] Wydajność FreeS/WAN:  
[http://www.freeswan.org/freeswan\\_trees/freeswan-2.02/doc/performance.html](http://www.freeswan.org/freeswan_trees/freeswan-2.02/doc/performance.html)
- [2] Bodo Möller, „TLS insecurity (attack on CBC)”, wysłane na listę dyskusyjną IETF-TLS, wrzesień 2001. Także tutaj: <http://www.openssl.org/~bodo/tls-cbc.txt>
- [3] Wei Dai, „an attack against SSH2 protocol”, wysłany na listę dyskusyjną sci.crypt, luty 2002
- [4] Phil Rogaway, „Problems with Proposed IP Cryptography”, kwiecień 1995: <http://www.cs.ucdavis.edu/~rogaway/papers/draft-rogaway-ipsec-comments-00.txt>
- [5] Ariel Futoransky, E. Kargieman i Ariel M. Pacetti, „An attack on CRC-32 integrity checks of encrypted channels using CBC and CFB modes”, CORE SDI S.A, 1998
- [6] Steven M. Bellovin, „Problem Areas for the IP Security Protocols”, Proceedings of the 1996 Usenix Security Symposium, sierpień 1996, strona 205
- [7] Stephen Kent i Randall Atkinson, „RFC 2406: IP Encapsulating Security Payload (ESP)”, listopad 1998
- [8] Nikita Borisov, Ian Goldberg i David Wagner, „Intercepting Mobile Communications: The Insecurity of 802.11”, Annual International Conference on Mobile Computing and Networking, 2001, strona 180
- [9] Bruce Schneier i Mudge, „Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)”, ACM Communications and Computer Security Conference, listopad 1998, strona 132: <http://www.schneier.com/paper-pptp.html>
- [10] Hugo Krawczyk, „The order of encryption and authentication for protecting communications (Or: how secure is SSL?)”, Crypto'01, Springer-Verlag Lecture Notes in Computer Science No.2139, August 2001, strona 310
- [11] Karl-Heinz Haag i Achim Leitner, artykuły o OpenSSH, Linux Magazine numer 24 – strona 50; numer 25 – strona 48; numer 49 – strona 52
- [12] Ralf Spenneberg, „Virtual Private Networks with Linux 2.4 and FreeS/WAN 2.01”, Linux Magazine, numer 36 – strona 52