Sieć Tor

Damian Matyjaszek

Spis treści

1 Historia sieci Tor

Rozdział został opracowany na podstawie historycznej, nieutrzymywanej już strony opisującej prace nad trasowaniem cebulowym, www.onion-router.net, a także oficjalnej strony internetowej organizacji The Tor Project, www.torproject.org.

Sieć Tor powstawała przez wiele lat. Początkowo był to projekt rządowy, mający na celu ochronę komunikacji wywiadowczej Stanów Zjednoczonych, lecz na przestrzeni lat stał się wolnym, dostępnym publicznie oprogramowaniem [1, 4].

Rozwój sieci Tor można podzielić na kilka generacji. Od momentu rozpoczęcia prac w 1995 roku do wydania oprogramowania w połowie 1996 roku trwał pierwszy etap rozwoju projektu. Po niej, a przed pojawieniem się sieci Tor, trwał etap nazywany Trasowanie Cebulowe: Następna Generacja (ang. Onion Routing: The Next Generation). Ostatni etap rozwoju, trwający aż do teraz nosi nazwę „Tor: The Second-Generation Onion Route", chociaż przyjęło mówić się po prostu Tor (The Onion Router). Ze względu na to, że ostatni etap rozwoju sieci Tor wskazuje, że jest to druga generacja, więc numerację należy zacząć od 0. Tak więc pierwszy etap będzie generacją 0, drugi generacją 1, a ostatni oczywiście generacją 2 [3].

Obecnie rozwojem i utrzymaniem sieci Tor zajmuje się organizacja non-profit The Tor Project, założona w 2006 roku [6].

1.1 Generacja 0

Prowadzenie pierwszych rozmów na temat Trasowania Cebulowego rozpoczęto w 1995 roku. Początkowo dyskusje dotyczyły funkcji, które ma posiadać i na jakiej zasadzie ma ono działać. Projekt został sfinansowany przez Biuro ds. Badań i Rozwoju Marynarki Wojennej (ang. Office of Naval Research, ONR) [4].

W 1996 roku pojawiła się już pierwsza formalna publikacja, oraz prezentacja Trasowania Cebulowego pod nazwą „Hiding Routing Information". Została ona opublikowana na First Hiding Workshop 31 maja 1996 roku. Zostały w niej opisane m.in. cel powstania Trasowania Cebulowego, zasada działania, a także podatności na pewne rodzaje ataków. Trasowanie Cebulowe było odporne na analizę ruchu w czasie rzeczywistym, lecz jednak po zebraniu odpowiedniej liczby danych możliwe było odkrycie stron komunikacji. Także przejęcie pierwszego, inicjującego serwera proxy sprawiało, że wszystkie dane były

ujawnione [5].

W tym samym roku został uruchomiony pierwszy działający prototyp projektu, składający się z 5 węzłów działających na maszynie z systemem Solaris 2.5.1/2.6, znajdującej się w Laboratorium Badań Morskich (ang. Naval Research Laboratory, NRL) [4]. Działająca wersja posiadała wsparcie dla protokołów HTTP, oraz Telnet, jednakże trwały prace nad serwisami mogącymi działać także z protokołami FTP i SMTP [5].

1.2 Generacja 1

Jeszcze w 1996 roku rozpoczęto prace nad Trasowaniem Cebulowym 1. generacji, zwanego również Systemem Następnej Generacji (ang. Next Generation System) [3]. Prace obejmowały m.in. usunięcie z głównej części kodu fragmentu odpowiedzialnego za kryptografie, co miało zapewnić większą modułowość. Zdecydowano się również na zachowanie projektu w postaci otwartoźródłowej. Dzięki publicznie dostępnemu kodowi źródłowemu Trasowanie Cebulowe zapewniłoby większe bezpieczeństwo. Każda luka mogła być bardzo szybko zauważona przez społeczność i naprawiona. Sprawiało to również, że oprogramowanie było darzone większym zaufaniem. Użytkownik nie musiał bać się o swoją anonimowość, wierząc twórcom oprogramowania na słowo, że w kodzie nie znajduje się fragment, który ujawnia dane zawieracjące inforamcje o jego tożsamości. Miało to zachęcić większą liczbę osób chcących zapobiec analizie ruchu sieciowego przesyłanych przez siebie wiadomości do korzystania właśnie z Trasowania Cebulowego. Kolejnym powodem dla którego zdecydowano się tworzyć projekt o otwartym kodzie były pewne ograniczenia eksportowe, które uniemożliwiały rozpowszechnienie kodu Trasowania Cebulowego generacji 0. W lipcu 1996 roku uznano, że kod projektu może zostać udostępniony publicznie.

W 1997 roku projekt Trasowania Cebulowego, w ramach Programu High Confidence Network, dostał wsparcie finansowe od Agencji Zaawansowanych projektów Badawczych w Obszarze Obronności (ang. Defense Advanced Research Projects Agency, DARPA). Tego samego roku Trasowanie Cebulowe otrzymało wiele nowych funkcjonalności, m.in. od tego momentu ścieżka, po której były przesyłane pakiety, mogła posiadać zmienną długość, routery zostały oddzielone od serwerów proxy, a moduł kryptograficzny mógł zostać uruchomiony na oddzielnej, specjalnie do tego przeznaczonej maszynie.

W 1998 roku organizacje NRL, NRaD (ang. Naval Research and Development) oraz

Uniwersytet w Maryland (ang. The University of Maryland, UMD) zdecydowały się na uruchomienie, w swoich oddziałach, kilku sieci Trasowania Cebulowego. Były to implementacje zarówno generacji 0, jak i 1. Zbudowane sieci mogły obsłużyć protokoły HTTP, FTP, SMTP, oraz rlogin [2, 4].

Pod koniec 1998 roku organizacja Zero Knowledge Systems ogłosiła powstanie własnej sieci - Freedom Network, o podobnym działaniu co Trasowanie Cebulowe. Projekt ten składał się z komercyjnych węzłów pośredniczących, a nie tak jak w Trasowaniu Cebulowym z węzłów utrzymywanych przez ochotników. Użytkownicy, którzy chcieli korzystać z tego sposobu zachowania anonimowości, musieli wykupić subskrybcję. Jednakże projekt ten nie zdołał się zbyt długo utrzymać. Już pod koniec 2001 roku sieć została zamknięta. Rozwiąznanie to nie cieszyło na tyle dużą popularnością, aby organizacja była w stanie pokryć koszty utrzymania swoich węzłów pośredniczących.

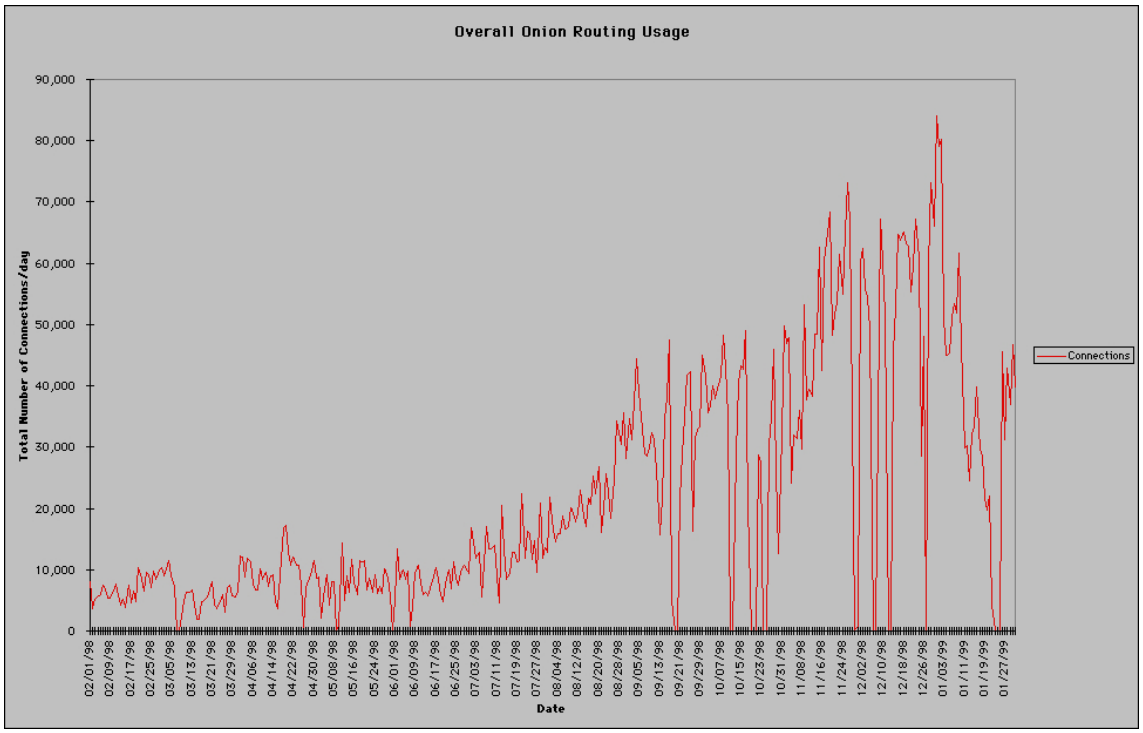
W 1999 roku publikacja dotycząca Trasowania Cebulowego o nazwie „Anonymous Connection and Onion Routing" została nagrodzona nagrodą Alan Berman Research Publication Award. Nagroda ta została ustanowiona przez pracownika NRL - Dr. Alana Bermana i przyznawana jest za najlepsze pisma techniczne w każdej z dziedzin naukowych [21]. Mimo to prace nad projektem zostały tymczasowo wstrzymane, aczkolwiek prace badawcze i analityczne nadal trwały.

W 2000 roku została zamknięta jedna z prototypowych sieci generacji 0. W trakcie swojego 2-letniego działania zanotowano ponad 20 milionów zapytań z ponad 60 krajów. Maksymalne obciążenie wyniosło 84022 odwiedzin i zostało odnotowane 12 grudnia 1998 roku. Wykres przedstaiający dzienne użycie sieci testowej w NRL został przedstawiony na rysunku 1.1.

Po dwuletniej przerwie w rozwoju ponowiono pracę nad rozwojem Trasowania Cebulowego. Projekt został sfinansowany przez DARPA w ramach programu Fault Tolerant Networks.

1.3 Generacja 2

Rok 2002 był przełomowy dla projektu. Cały dotychczasowy kod został porzucony ze względu na swoją przestarzałość. Projekt został napisany od nowa. Jako bazę dla nowej wersji Trasowania Cebulowego wykorzystano projekt jednego ze studentów uniwersytetu w Cambridge - Mateja Pfajfara. Od czasu rozpoczęcia prac nad Trasowaniem Cebulowym



Rysunek 1.1: Dzienny przepływ ruchu w prototypowej sieci w NRL.

Źródło: https://www.onion-router.net/Archives/Daily.gif

minęło 6 lat. W tym czasie powstały niezależne filtrujące serwery proxy, będące w stanie sprostać wymaganiom projektu. Do tego celu wykorzystano Privoxy. Pośredniczeniem na poziomie warstwy aplikacji miał zajmować się protokół SOCKS. Był on wstanie obsługiwać większość protokołów, których obsługą miało zajmować się Trasowanie Cebulowe. Dzięki temu pracownicy nie musieli się już zajmować rozwojem oprogramowania dla serwerów proxy każdej z aplikacji.

W 2003 roku Tor otrzymał wsparcie finansowe od ONR, DARPA i NRL. Tego samego roku została uruchomiona pierwsza sieć Trasowania Cebulowego 2. generacji. Od samego początku działania sieci zaczęli pojawiać się nowi ochotnicy chcący rozwijać projekt przez udostępnienie swoich maszyn jako węzłów pośredniczących sieci Tor, początkowo tylko z USA, lecz później do projektu dołączyli ochotnicy z innych krajów.

W 2004 roku zostały uruchomione pierwsze ukryte serwisy, oraz ukryta wiki (Hidden Wiki). 13 sierpnia Tor został przedstawiony na USENIX Security jako „Tor: Second-Generation Onion Router”. Pod koniec roku ONR i DARPA zakończły wsparcie finansowe projektu, ale zamiast nich finansowanie rozwoju i wdrażania projektu rozpoczęła założona w 1990 roku pozarządowa organizacja Electronic Frontier Foundation (EFF),

której głównym celem jest walka o wolność słowa oraz prywatności w Internecie [4, 22].

1.4 Projekt Tor

W grudniu 2006 roku została założona ogranizacja non-profit The Tor Project. Miała ona na celu zapewnić dalszy rozwój oraz utrzymanie sieci Tor. Wśród jej twórców znajdują się m.in. Roger Dingledine i Nick Mathewson [7]. Początkowo sponsoringiem fiskalnym[1] The Tor Project zajmowała się organizacja EFF [23]. Wsparciem finansowym projektu zajmowały się takie organizacje jak: Broadcasting Board of Governors, National Science Foundation, Internews Europe, Human Rights Watch, Cyber-TA project, Bell Security Solutions, a także Omidyar Network Enzyme Grant [10].

The Tor Project, oprócz rozwoju Tor, zajmuje się również tworzeniem oprogramowania, które ma zapewnić anonimowość w Internecie przy wykorzystaniu sieci Tor. Flagowym projektem jest Tor Browser. Jest to przeglądarka internetowa, bazująca na Mozilli Firefox, zawieracjąca wbudowanego klienta sieci Tor. Używając jej do przeglądania stron internetowych, cały nasz ruch jest szyfrowany, a następnie przekierowywany przez sieć Tor. Dzięki niej mamy również umożliwiony dostęp do ukrytych stron internetowych, korzystających ze specjalnej, używanej tylko w sieci Tor, domeny najwyższego poziomu .onion [8]. Przeglądarka została publicznie udostępniona na oficjanlej stronie The Tor Project w 2008 roku [19]. Była ona dostępna pod nazwą Tor Browser Bundle, a od 2014 roku nosi po prostu nazwę Tor Browser [20]. Do innych ważniejszych projektów organizacji The Tor Project można zaliczyć Orbot, aplikację wydaną w 2008 roku, przeznaczoną na system operacyjny Android, której celem jest szyfrowanie, a następnie przekierowywanie przez sieć Tor przesyłanych przez Internet danych, wybranych przez użytkownika aplikacji znajdujących się na urządzeniu [26]. W 2009 roku została wydana pierwsza wersja programu działającego w trybie tekstowym o nazwie Nyx [15]. Została ona przeznaczona dla osób, które chciałyby monitorować stan administrowanego przez siebie przekaźnika znajdującego się w sieci Tor. Program ten został napisany w języku Python i pozwala m.in. na obserwację wykorzystania zasobów komputera, nawiązanych połączeniach i wielu innych informacji o naszym przekaźniku [13]. Poza powyżej wymienionymi jest jeszcze

[1] Sponsoring fiskalny jest praktyką, stosowaną przez organizacje nonprofit, zapewniającą innym organizajom lub projektom zwolnienie z podatku, nadzór powierniczy, zarządzanie finansami oraz inne usługi administracyjne [27]

wiele innych aplikacji, pomagających w ochronie naszej prywatności, przy wykorzystaniu sieci Tor [9].

Dzięki swoim osiągnięciom w walce o zachowanie anonimowości, prywatności i wolności słowa w Internecie The Tor Project otrzymał wiele nagród. Wśród nich można wyróżnić Free Software Foundation 2010, otrzymaną w marcu 2011 roku za Projects of Social Benefit. Nagroda ta przyznawana jest projektom, które przynoszą korzyści dla społeczeństwa [25]. Z kolei we wrześniu 2012 roku Projektowi Tor została przyznana EFF Pioneer Award, która zostaje przyznawana od 1992 roku liderom, którzy wpływają na rozwój wolności oraz innowacji w zakresie technologii informatycznych [24].

2 Szyfrowanie

2.1 Algorytm Diffiego-Hellmana

2.2 Szyfrowanie Symetryczne

2.3 Szyfrowanie Asymetryczne

2.4 Certyfikaty

3 Trasowanie Cebulowe

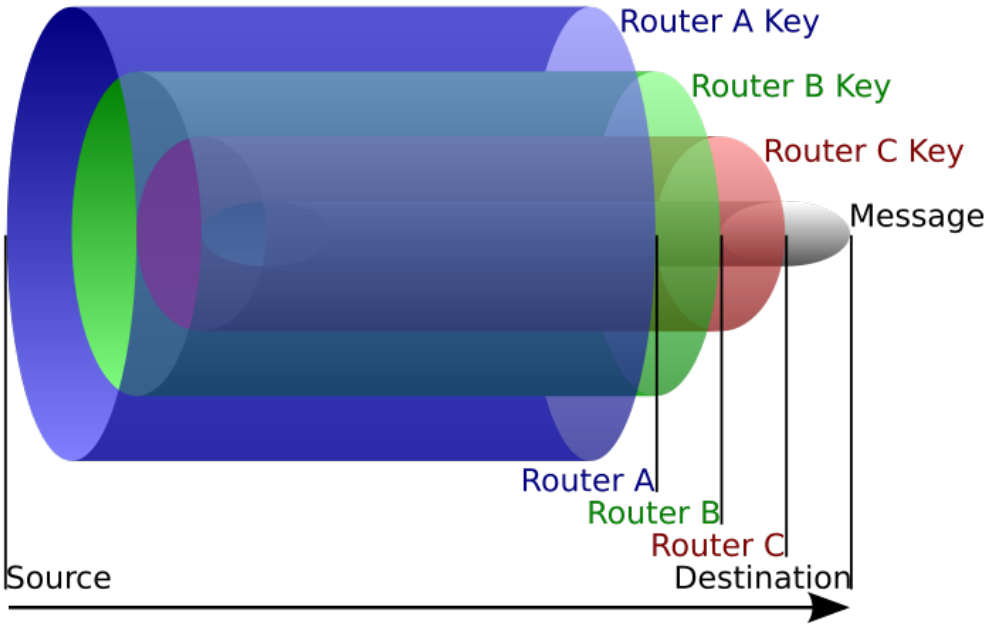
Rozdział ten został opracowany na podstawie dokumentacji projektowych Sieci Tor: „Tor: The Second-Generation Onion Router" oraz „Hiding Routing Information", a także oficjalnej strony internetowej organizacji zajmującej się rozwojem Sieci Tor, The Tor Project.

Trasowanie Cebulowe jest techniką mającą na celu zapobiec analizie ruchu sieciowego poprzez ukrywanie informacji dotyczących routingu przesyłanych pakietów. Do jej działania wykorzystystywana jest grupa serwerów pośredniczących, przez które przekierowywana jest wiadomość, zanim trafi do docelowego odbiorcy. Zasada działania polega na tym, że nadawca wiadomości pobiera listę węzłów i wybiera kilka spośród nich (węzły te będą tworzyć obwód, przez który będzie pośredniczona wiadomość), a następnie wielokrotnie szyfruje przesyłane dane. Szyfrowanie odbywa się za pomocą symetrycznych kluczy współdzielonych z kolejnymi serwerami tworzącymi obwód w odwrotnej kolejności niż się w nim znajdują (najpierw do szyfrowania zostaje użyty klucz współdzielony z pierwszym z nich, a na końcu ten, który jest wspólny z węzłem znajdującym się na końcu obwodu). Zaszyfrowana wiadomość zostaje wysłana do pierwszego serwera pośredniczącego, który odszyfrowuje ją, co powoduje odkrycie następnego węzła do którego ma zostać przekierowana wiadomość. Kolejno, wiadomość zostaje przekazywana do następnych węzłów, które zdejmują kolejne warstwy szyfru, do momentu aż trafi do ostatniego, który zdejmuje najgłębszą warstwę szyfru i przekazuje wiadomość w oryginalnej postaci do docelowego odbiorcy. Do pośredniczenia pakietów używany jest protokół SOCKS.

Zbyt długo istniejący obwód może doprowadzić do zebrania przez atakującego odpowiedniej ilości danych do wykrycia lokalizacji węzłów w sieci Tor. Aby temu zapobiec zdecydowano się na ograniczenie czasu istnienia obwodu. Każdy obwód w sieci ma ustalony czas wygasania. Domyślnie wynosi on 10 minut. Po tym czasie, jeśli strumień nie jest używany, tworzony jest nowy obwód [12].

Dzięki takiemu systemowi przesyłania wiadomości niemożliwe jest ustalenie całej trasy przesyłanego pakietu. Każdy z serwerów pośredniczących zna tylko dwóch swoich sąsiadów. Pierwszy z nich zna nadawcę, lecz nie wie jaka jest treść przesyłanej wiadomośći, a ostatni zna odbiorcę, ale nie zna nadawcy wiadomości [14].

Rysunek 3.1 przedstawia wygląd wiadomości, która zostaje przesłana przez obwód składający się z trzech węzłów. Kolor niebieski oznacza warstwę szyfru utworzoną za po-



Rysunek 3.1: Diagram przedstawiający przykładową wiadomość przesyłaną przez obwód.

Źródło: https://upload.wikimedia.org/wikipedia/commons/e/e1/Onion_diagram.svg

mocą symetrycznego klucza wspólnego dla nadawcy i węzła znajdującym się na początku obwodu, a czerwony warstwę szyfru utworzoną za pomocą klucza współdzielonego z ostatnim serwerem pośredniczącym. Najgłębsza (szara) warstwa wiadomości oznacza niezaszyfrowane dane, które mają trafić do docelowego hosta.

3.1 Komórki

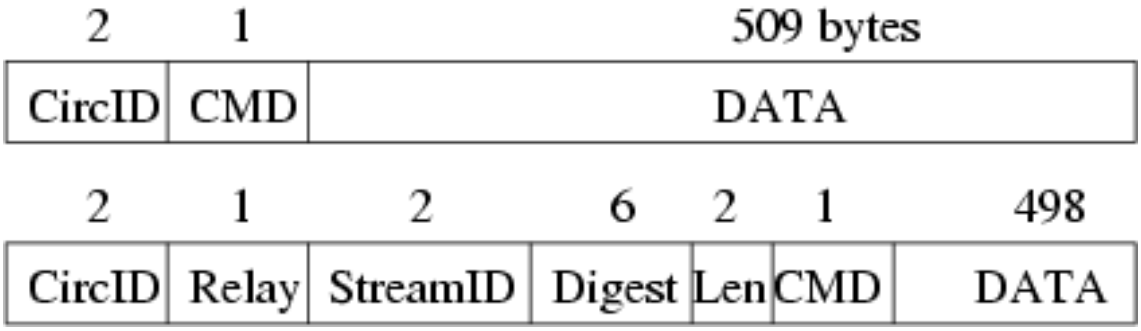
Przesyłaną jednostką danych w Trasowaniu Cebulowym jest tzw. komórka. Ma ona stałą długość 512 bajtów i składa się z nagłówka oraz przesyłanej treści. Długość nagłówka jest uzależniona od typu komórki. Może mieć on 3 lub 14 bajtów. Najważniejszymi składowymi nagłówka jest identyfikator obwodu oraz polecenie. Rysunek 3.2 przedstawia wygląd komórki.

Pierwsze 2 bajty są zajmowane przez wcześniej wspomniany identyfikator obwodu. Jest on wymagany, gdyż w pojedyńczym połączeniu pomiędzy poszczególnymi węzłami może zostać ustanowionych wiele obwodów i każdy węzeł musi wiedzieć do którego z nich należy komórka.

Kolejny zajmowany bajt przeznaczony jest na polecenie, dzięki któremu węzeł, który otrzyma komórkę, będzie w stanie zinterpretować ją w odpowiedni sposób. Wszystkie polecenia można podzielić na dwa typy, kontrolne oraz przekazujące. Do tych pierwszych należą *padding*, *create* oraz *destroy*. Służą one kolejno do utrzymywania połączeń, ustanawiania nowych obwodów oraz niszczenia ich, z koleji poleceniem mówiącym nam o tym, że mamy do czynienia z komórką przekazującą jest *relay*.

Struktura komórki posiadającej polecenie przekazujące różni się od komórek z poleceniem kontrolnym tym, że pierwsze 11 bajtów treści komórki traktowane jest jako rozszerzenie jej nagłówka. Pierwsze 2 bajty tego rozszerzenia przeznaczone są na identyfikator strumienia, który jest stosowany ze względu na to, że pojedyńczy strumień może zostać zmultipleksowany na wiele obwodów, więc potrzebny jest mechanizm pozwalający odróżnić do jakiego strumienia w obwodzie należy komórka. Kolejne 6 bajtów zajmuje suma kontrolna, stworzona za pomocą funkcji skrótu SHA-1, pozwalająca na sprawdzenie integralności przesyłanych danych. Kolejne pole nagłówka określa długość przesyłanej treści, a ostatni bajt definiuje odpowiednie polecenie komórki przekazującej, które może mieć następujące wartości:

- *relay data* - przesyłanie danych w strumieniu
- *relay begin* - otwarcie strumienia
- *relay end* - bezpieczne zamknięcie strumienia
- *relay teardown* - zamknięcie uszkodzonego strumienia
- *relay connected* - powiadomienie proxy nadawcy o rozpoczęciu przekazywania
- *relay extend* - rozszerzenie obwodu o nowy węzeł
- *relay extended* - potwierdzenie rozszerzenia obwodu
- *relay truncate* - zamknięcie części obwodu
- *relay truncated* - potwierdzenie zamknięcia części obwodu
- *relay sendme* - kontrola przeciążenia
- *relay drop* - implementacja atrap dalekiego zasięgu [16]



Rysunek 3.2: Struktura komórki Trasowania Cebulowego typu kontrolnego (u góry) oraz przekazującego.

Źródło: https://svn.torproject.org/svn/projects/design-paper/cell-struct.png

3.2 Proces tworzenia obwodu

Wybrane przez użytkownika serwery pośredniczące, przez które przekierowywana jest wiadomość tworzą tzw. obwód. Proces tworzenia takiego obwodu przebiega w sposób iteracyjny. Załóżmy, że Alice jest nadawcą, Bob pierwszym, a Carol drugim i jednocześnie ostatnim węzłem w obwodzie:

1. Alice w celu utworzenia połączenia z Bobem, wysyła do niego wiadomość z poleceniem *create*. W nagłówku wiadomości znajduje się również wybrany przez Alice, nieużywany pomiędzy nią, a Bobem identyfikator obwodu. Treścią wiadomości jest pierwsza część procesu Diffiego-Hellmana, służącego do uzgadniania klucza symetrycznego. Komórka jest zaszyfrowana za pomocą algorytmu RSA, przy użyciu klucza publicznego Boba.

2. Bob na podstawie otrzymanej od Alice wiadomości oraz własnych danych generuje klucz, a następnie w celu finalizacji procesu uzgadniania klucza wysyła do Alice wiadomość z poleceniem *created*, której treścią jest drugą część procesu, oraz skrót wygenerowanego klucza.

3. Alice generuje klucz, a następnie porównuje jego skrót z tym otrzymanym od Boba. Jeśli oba skróty się zgadzają oznacza to, że Alice i Bob posiadają ten sam klucz symetryczny, za pomocą którego będą szyfrowane oraz odszyfrowywane przesyłane między nimi wiadomości.

4. Alice w celu rozszerzenia obwodu o kolejny węzeł wysyła do Boba wiadomość

z poleceniem *relay extend*, posiadającą adres Carol oraz połowę procesu uzgadniania klucza symetrycznego.

5. Bob tworzy nową wiadomość zawierającą polecenie *create* oraz nieużywany między nim a Carol identyfikator obwodu, a następnie pakuje do niej treść wiadomości otrzymanej od Alice i wysyła ją do Carol.

6. Po otrzymaniu wiadomości Carol wysyła do Boba wiadomość z poleceniem *created*, której treścią jest drugi etap procesu uzgadniania klucza symetrycznego między nią a Alice.

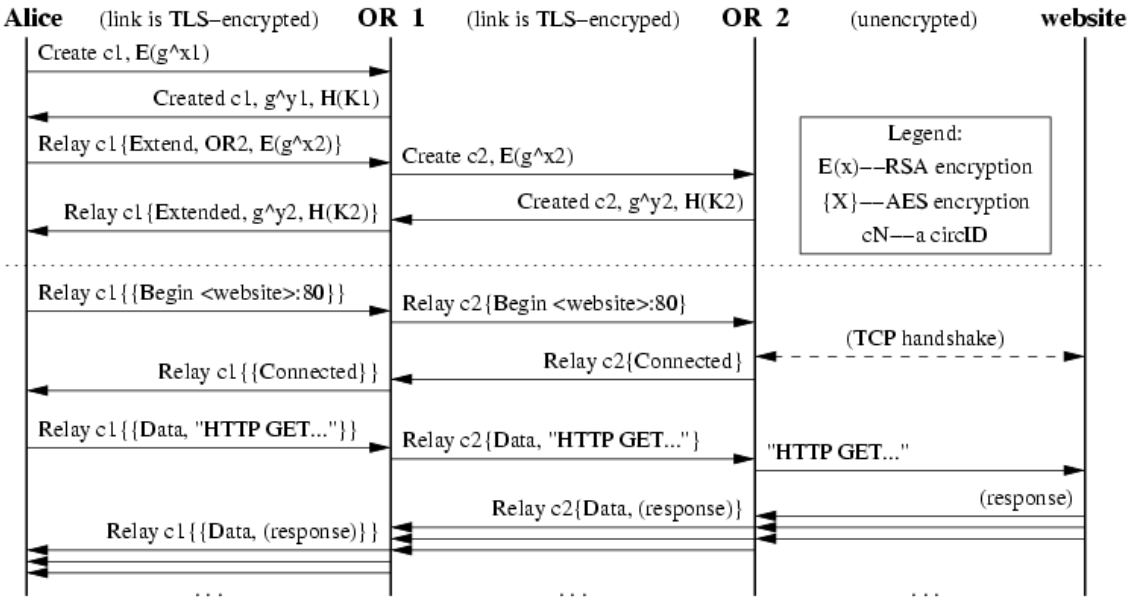
7. Bob oraz Carol mają zapisane w swoich tablicach informacje o utworzonych połączeniach. Dodatkowo Bob powiązuje ze sobą identyfikator połączenia Alice z Carol, dzięki czemu wszystkie wiadomości od Alice będą przekierowywane do Carol i na odwrót [5]. Po tym, w celu powiadomienia o pomyślnym rozszerzeniu obwodu oraz finalizacji procesu uzgadniania klucza, do Alice zostaje wysłana wiadomość z poleceniem *relay extended* z treścią poprzednio otrzymanej wiadomości od Carol.

8. Alice generuje klucz symetryczny, identyczny z tym posiadanym przez Carol. Od tego momentu obwód jest rozszerzony o nowy węzeł.

9. W celu dalszego rozszerzania obwodu powyższy proces może być powtarzany, z uwzględnieniem zwiększenia długości obwodu o nowy węzeł w każdym etapie rozszerzania.

Powyższy proces został zilustrowany na rysunku 3.3. H(Kx) oznacza skrót SHA-1 klucza symetrycznego współdzielonego z węzłem x, a g^{x_z} oraz g^{y_z} oznaczają pierwszą oraz drugą część procesu uzgadniania klucza z węzłem z. Dodatkowo poniżej przerywanej linii widoczny jest proces połączenia ze stroną internetową przy wykorzystaniu Trasowania Cebulowego.

W celu połączenia się z przykładową witryną internetową nadawca szyfruje zapytanie za pomocą wcześniej uzgodnionych kluczy symetrycznych serwerów pośredniczących, a następnie przesyła je przez obwód. Do nawiązania połączenia wykorzystywana jest komórka zawierająca polecenie *relay begin* wraz z nowym nieużywanym identyfikatorem strumienia. Ostatni węzeł, po odebraniu wiadomości, w imieniu inicjatora nawiązuje połączenie z witryną, a następnie odsyła do nadawcy komórkę *relay connected*. Po



Rysunek 3.3: Przykładowy przebieg tworzenia obwodu oraz komunikacji ze stroną internetową z wykorzystaniem Trasowania Cebulowego.
Źródło: https://svn.torproject.org/svn/projects/design-paper/interaction.png

nawiązaniu połączenia, każde zapytanie pakowane jest w komórkę zawierającą polecenie *relay data*, która jest szyfrowana i przesyłana przez obwód. Przy każdym przeskoku każdy kolejny węzeł zdejmuje warstwę przy użyciu współdzielonego między nim oraz nadawcą wiadomości klucza symetrycznego, aż niezaszyfrowana wiadomość trafi do serwera docelowego. Odpowiedź odbiorcy przebiega w odwrotnej kolejności. Przy każdym przeskoku każdy z węzłów dodaje jedną warstwę szyfru, co powoduje, że do nadawcy trafia wielokrotnie zaszyfrowana wiadomość. Oprogramowanie inicjatora komunikacji musi zdjąć wszystkie warstwy szyfru, zanim prawidłowa odpowiedź trafi do odpowiedniego programu w systemie [16].

3.3 Padding

Wraz ze zdjęciem każdej warstwy rozmiar wiadomości ulega zmniejszeniu. Osoba analizująca pakiety w sieci mogłaby na podstawie długości przesyłanych komórek ustalić położenie danego węzła w obwodzie. Aby temu zapobiec stosuje się tzw. padding, czyli wypełnienie pozostałego wolnego miejsca w komórce losowymi danymi. Przy każdym przeskoku w obwodzie każdy węzeł dodawaje odpowiednią ilość paddingu. Jeżeli komórka jest przesyłana w stronę inicjatora połączenia, padding musi być usuwany wraz

z każdym przeskokiem komórki w obwodzie. W przypadku niektórych komórek np. tych zawierających polecenie *destroy*, padding zajmuje całą treść wiadomości [5].

3.4 Niszczenie obwodów

Obwód może zostać zniszczony przez każdy z węzłów, który się w nim znajduje. W takiej sytuacji tworzona jest komórka zawierająca polecenie *destroy* oraz identyfikator obwodu, który ma zostać zniszczony. Treść takiej komórki jest pusta. Następnie taka komórka zostaje wysłana do obu sąsiednich węzłów. Każdy z nich ma obowiązek przekazać taką komórkę w odpowiednim kierunku obwodu, a następnie usunąć ze swojej tablicy wpisy dotyczące tego obwodu [5].

3.5 Serwery katalogowe

Klient sieci Tor musi posiadać aktualną listę serwerów pośredniczących, wraz z ich kluczami publicznymi, oraz aktualnym stanem. Pierwotnie te informacje miały być udostępniane na zasadzie peer-to-peer. Oznacza to, że węzły miały wymieniać się posiadanym o sobie danymi. Ten sposób okazał się jednak nieodpowiedni ze względów bezpieczeństwa. Niektóre węzły w sieci mogły posiadać różne informacje o tych samych serwerach pośredniczących, co mogło umożliwić potencjalnemu napastnikowi określenie położenia węzła w sieci. Poza tym wymiana tych danych między węzłami spowodowałaby niepotrzebne obciążenie sieci. Zdecydowano więc, że informacje te będą przechowywane w centralnych serwerach zwanych serwerami katalogowymi. Domyślnie oprogramowanie klienckie posiada listę wszystkich tych serwerów, od których raz na jakiś czas pobierane są informacje o wszystkich węzłach pośredniczących występujących w sieci Tor. Każdy taki serwer katalogowy jest serwerem HTTP. Pozwala to na proste udostępnianie oraz pobieranie informacji. Każdy serwer, który udostępnia swoje dane serwerowi katalogowemu musi wcześniej je podpisać za pomocą swojego klucza. Ta sama sytuacja dotyczy serwera katalogowego. Zanim zebrane przez serwer katalogowy infromacje zostaną udostępnione klientom, muszą zostać podpisane przez serwer katalogowy przy użyciu swojego klucza prywatnego. Ma to uniemożliwić potencjalnemu napastnikowi na podstawienie własnego serwera pośredniczącego lub katalogowego [16].

4 Ukryte serwisy

W 2004 roku twórcy sieci Tor opracowali funkcjonalność zwaną serwisami cebulowymi lub też ukrytymi serwisami. Funkcjonalność ta pozwala nam na udostępnienie w sieci Tor naszej usługi bez ujawniania publicznego adresu IP zarówno maszyny na której się ona znajduje jak i adresu osoby łączącej się z daną usługą [16].

Połączenie z takim serwisem znacznie różni się od tego co możemy zaobserwować w przypadku połączenia ze zwykłą usługą w internecie. Cała komunikacja przebiega przy użyciu sieci Tor, klient zamiast łączyć się bezpośrednio z serwisem, komunikuje się z nim za pomocą jednego z węzłów, pełniącego rolę tzw. punktu spotkania, a nazwa domeny za pomocą której się łączy nie odnosi się do lokalizacji danego serwisu, tylko do specjalnego deskryptora zawierającego listę węzłów pełniących rolę punktów wejściowych dla danego serwisu oraz jego klucza publicznego [11]. Dokładne wyjaśnienie czym są punkt spotkania oraz punkty wejściowe zostało omówione w podrozdziałach 4.2 oraz 4.3.

Dostęp do takiej usługi jest możliwy tylko przy wykorzystaniu sieci Tor oraz specjalnego adresu w postaci *nazwa.onion*, gdzie *nazwa* jest nazwą serwisu powstałą na podstawie jego klucza publicznego. Taka nazwa może składać się z 16 lub 56 znaków (długość nazwy jest zależna od wersji protokołu, której zdecydujemy się użyć) [17]. Jednym z powodów dla którego nazwa serwisu jest tworzona na podstawie klucza publicznego, zamiast użycia mnemonicznej nazwy wymyślonej przez użytkownika jest możliwość sprawdzenia, czy otrzymany przez klienta, chcącego połączyć się z danym serwisem, klucz publiczny należy do serwisu o podanym adresie [18]. Taki adres nie jest przechowywany w żadnym centralnym punkcie, a uzyskanie go jest możliwe tylko z niezależnych źródeł na których został umieszczony przez właściciela serwisu lub inne osoby mające go w posiadaniu [11].

4.1 Rozproszona tablica mieszająca

Rozproszona tablica mieszająca (ang. Distributed Hash Table, DHT) jest zdecentralizowanym, rozproszonym systemem wyszukiwania, złożonym z węzłów (chyba) katalogowych, który pozwala, w przypadku sieci Tor, na wyszukanie deskryptora należącego do serwisu cebulowego o podanym adresie. Informacje w DHT są przechowywane na zasadzie pary (klucz, wartość). W tym przypadku kluczem jest nazwa ukrytego serwisu,

a wartością jego deskryptor. Każdy z węzłów w DHT przechowuje deskryptory o adresie należącym do grupy za którą jest on odpowiedzialny. Podczas tworzenia serwisu cebulowego, deskryptor wysyłany jest do sieci, a następnie przesyłany jest przez kolejne węzły do momentu aż trafi do węzła odpowiedzialnego za przechowywanie deskryptora o podanym adresie, który to zapisuje w swojej tablicy [11, 28].

4.2 Proces tworzenia serwisu cebulowego

1. Na początku serwis cebulowy wybiera losowo kilka węzłów w sieci, aby pełniły rolę tzw. punktów wprowadzających. Serwis tworzy obwód do każdego z tych węzłów, a następnie odpytuje każdego z nich o możliwość pełnienia funkcji punktu wprowadzającego, podając przy tym własny klucz publiczny.

2. Po otrzymaniu odpowiedzi od wybranych węzłów, serwer cebulowy tworzy deskryptor, składający się z klucza publicznego serwisu oraz informacji na temat wszystkich punktów wejściowych. Następnie taki deskryptor zostaje podpisany za pomocą klucza prywatnego serwisu.

3. W kolejnym kroku serwer cebulowy wysyła wcześniej utworzony deskryptor do rozproszonej tablicy hashującej.

4. Od tego momentu możliwe jest połączenie się z wybranym serwerem cebulowym poprzez adres serwisu [11].

4.3 Proces połączenia z serwisem cebulowym

1. W celu połączenia się z ukrytym serwisem, klient pobiera z rozproszonej tablicy mieszającej *deskryptor serwisu cebulowego*, tym samym otrzymując lokalizacje jego punktów wprowadzających oraz jego klucz publiczny.

2. Klient wybiera jeden z węzłów sieci Tor, aby był on jego *punktem spotkania* a następnie wysyła do niego jednorazowy sekret (ciasteczko).

3. Następnie tworzy on *wiadomość wprowadzającą*, zawierającą adres punktu spotkania oraz jednorazowy sekret.

4. W kolejnym kroku klient wysyła do jednego z punktów wejściowych wcześniej utworzoną wiadomość, aby ten przekazał ją do serwisu cebulowego (cała komunikacja przebiega przy użyciu wcześniej otworzonego obwodu pomiędzy punktem wprowadzającym, a ukrytym serwisem).

5. Serwer cebulowy odszyfrowuje otrzymaną wiadomość wprowadzającą z której otrzymuje adres punktu spotkania oraz jednorazowy sekret.

6. Następnie serwer cebulowy tworzy obwód do punktu spotkania.

7. Serwer cebulowy wysyła do punktu spotkania wiadomość zawierającą otrzymany jednorazowy sekret.

8. Serwer cebulowy informuje klienta o nawiązaniu połączenia.

9. Klient oraz serwer mogą się komunikować poprzez pośredniczący punkt spotkania. (połączenie pomiędzy klientem, a punktem spotkania oraz między punktem spotkania i serwisem cebulowym przebiega przez sieć Tor, zresztą jak cała opisana powyżej komunikacja) [11].

Literatura

[1] J. B. Fagoyinbo, The Armed Forces: Instrument of Peace, Strength, Development and Prosperity, Bloomington 2013

[2] W. Gragido, Cybercrime and Espionage: An Analysis of Subversive Multi-Vector Threats, Burlington 2011

[3] https://www.onion-router.net/

[4] https://www.onion-router.net/History.html

[5] https://www.onion-router.net/Publications/IH-1996.pdf

[6] https://www.torproject.org/press/2008-12-19-roadmap-press-release

[7] https://www.torproject.org/about/findoc/2009-TorProject-Form990andPC.pdf

[8] https://www.torproject.org/projects/torbrowser/design/

[9] https://www.torproject.org/projects/projects.html.en

[10] https://www.torproject.org/about/sponsors.html.en

[11] https://www.torproject.org/docs/onion-services.html.en

[12] https://www.torproject.org/docs/tor-manual-dev.html.en

[13] https://www.torproject.org/projects/nyx.html.en

[14] https://www.torproject.org/about/overview.html.en

[15] https://nyx.torproject.org/changelog/legacy.html

[16] https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf

[17] https://trac.torproject.org/projects/tor/wiki/doc/HiddenServiceNames#Howare.onionnamescreated

[18] https://trac.torproject.org/projects/tor/wiki/doc/HiddenServiceNames#Whyare.onionnamescreatedthatway

[19] https : / / web . archive . org / web / 20081029125231 / http : / / www . torproject . org : 80/easy-download.html.en

[20] https://web.archive.org/web/20140701221249/https://www.torproject.org/projects/torbrowser.html.en

[21] https://www.gl.ciw.edu/news/ahart-receives-berman-award

[22] https://www.eff.org/about

[23] https://www.eff.org/press/archives/2004/12/21-0

[24] https://www.eff.org/awards/pioneer/2012

[25] https://www.fsf.org/news/2010-free-software-awards-announced

[26] https://guardianproject.info/apps/orbot/

[27] http : / / www . trustforconservationinnovation . org / about / pdf / TCI-FS-whitepaper-201403.pdf

[28] http://students.mimuw.edu.pl/SR-MSUI/06-dht/dht.pdf