# Sieć Tor

Damian Matyjaszek

# Spis treści



[1] https://svn.torproject.org/svn/projects/design-paper/tor-design.html

# 1 Historia sieci Tor

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[2],[3].

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[4].

Obecnie rozwojem i utrzymaniem sieci Tor zajmuje się organizacja non-profit The Tor Project, założona w 2006 roku[5].

## 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 (ONR)[2].

Rok później pojawiła się już pierwsza formalna publikacja, oraz prezentacja Trasowania Cebulowego pod nazwą „Hiding Routing Information". Została ona opublikowana na First Hiding Workshop 31 maja. 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

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

[3]J. B. Fagoyinbo, The Armed Forces: Instrument of Peace, Strength, Development and Prosperity, Bloomington 2013, s. 262

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

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

pierwszego, inicjującego serwera proxy sprawiało, że wszystkie dane były ujawnione[6].

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 (NRL)[2]. Działająca wersja posiadała wsparcie dla protokołów HTTP, oraz Telnet, jednakże trwały prace nad serwisami mogącymi działać także z protokołami FTP i SMTP[6].

## 1.2 Generacja 1

Jeszcze w 1996 roku rozpoczęto prace nad Trasowaniem Cebulowym 1. generacji, zwanego również Systemem Następnej Generacji (ang. Next Generation System)[4]. 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 tego samego 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 (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.

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

Rok później organizacje NRL, NRaD (ang. Naval Research and Development) oraz Uniwersytet w Maryland 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][7].

Pod koniec tego samego 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[8]. Mimo to prace nad projektem zostały tymczasowo wstrzymane, aczkolwiek prace badawcze i analityczne nadal trwały.

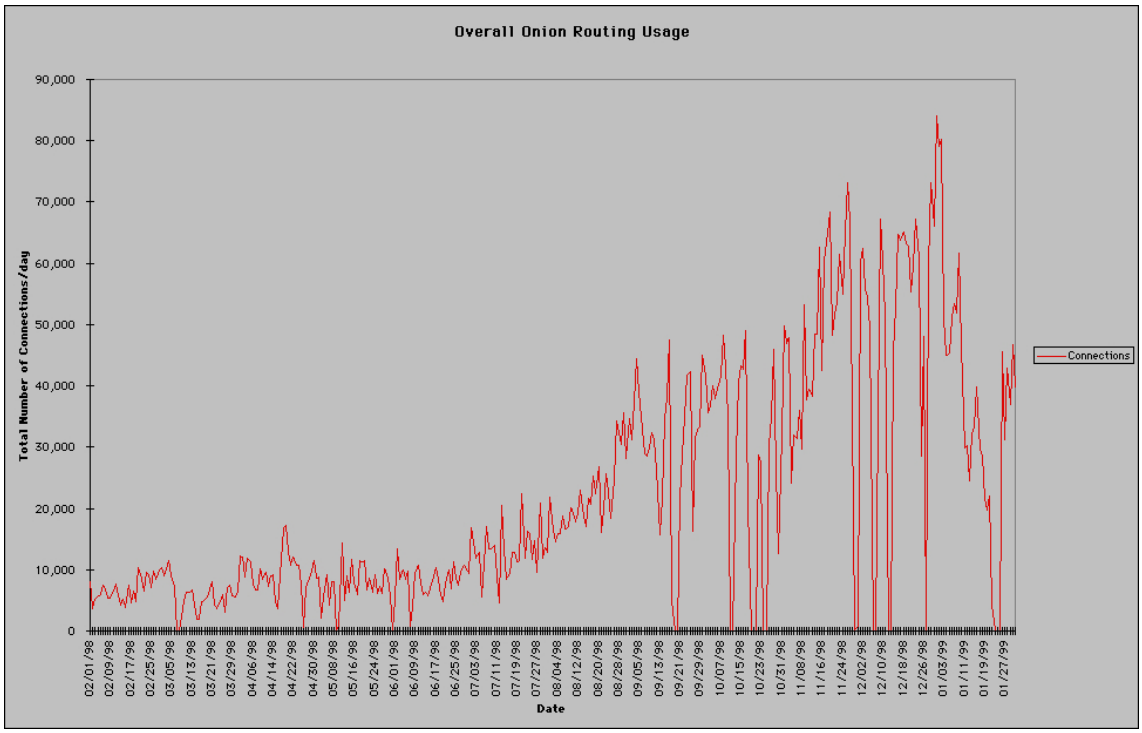
Kolejnego roku została zamknięta jedna z prototypowych sieci generacji 0. W trakcie swojego 2-letniego działania zanotowano ponad 20 milionów zapytań z ponad 60 krajów. Maksymalne obciążenie wyniosło 84022 odwiedzin i zostało odnotowane 12 grudnia 1998 roku. Wykres przedstaiający dzienne użycie sieci 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.

[7]W. Gragido, Cybercrime and Espionage: An Analysis of Subversive Multi-Vector Threats, Burlington 2011, s.188

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

[9]https://www.onion-router.net/Archives/Daily.gif




Rysunek 1.1: Dzienny przepływ ruchu w prototypowej sieci w NRL[9].

## 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 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.

Rok później 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 rozpoczął EFF[2].

## 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[10]. Początkowo sponsoringiem fiskalnym The Tor Project zajmowała się założona w 1990 roku pozarządowa organizacja Electronic Frontier Foundation (EFF), której głównym celem jest walka o wolność słowa oraz prywatności w Internecie[11],[12]. 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[13].

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, zawierająca 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[14]. Przeglądarka została publicznie udostępniona na oficjanlej stronie The Tor Project w 2008 roku[15]. Była ona dostępna pod nazwą Tor Browser Bundle, a od 2014 roku nosi po prostu nazwę Tor Browser[16]. 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

[10]https://www.torproject.org/about/findoc/2009-TorProject-Form990andPC.pdf
[11]https://www.eff.org/about
[12]https://www.eff.org/press/archives/2004/12/21-0
[13]https://www.torproject.org/about/sponsors.html.en
[14]https://www.torproject.org/projects/torbrowser/design/
[15]https://web.archive.org/web/20081029125231/http://www.torproject.org:80/easy-download.html.en
[16]https://web.archive.org/web/20140701221249/https://www.torproject.org/projects/torbrowser.html.en

przez sieć Tor przesyłanych przez Internet danych, wybranych przez użytkownika aplikacji znajdujących się na urządzeniu[17]. W 2009 roku została wydana pierwsza wersja programu działającego w trybie tekstowym o nazwie Nyx[18]. 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[19]. Poza powyżej wymienionymi jest jeszcze wiele innych aplikacji, pomagających w ochronie naszej prywatności, przy wykorzystaniu sieci Tor[20].

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[21]. 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[22].

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

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

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

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

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

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

# 2 Trasowanie Cebulowe

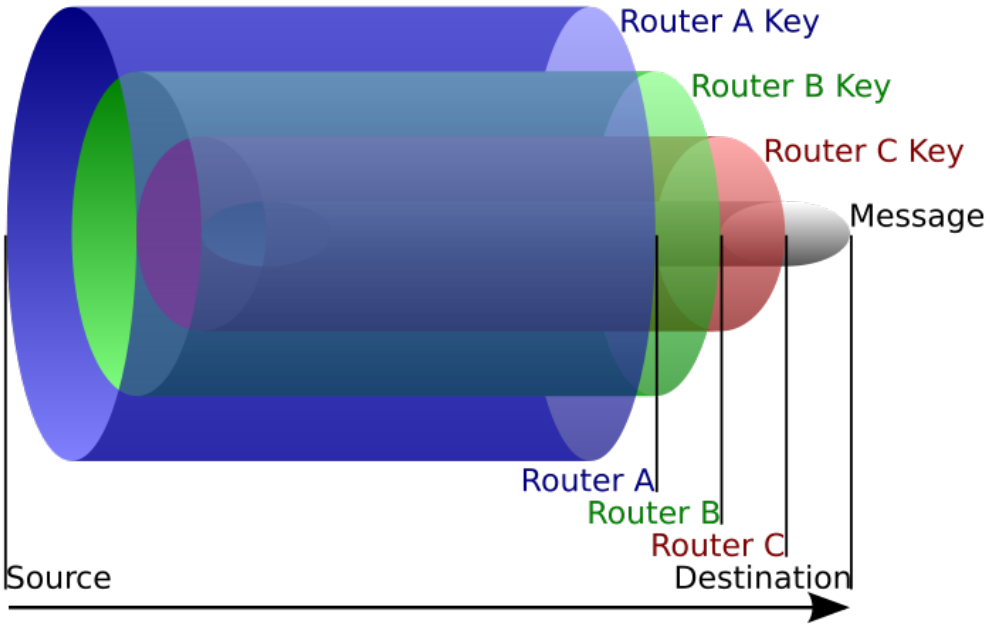
Trasowanie Cebulowe jest techniką pozwalającą na anonimową komunikację w Internecie. Do jej działania wykorzystystywana jest grupa węzłów przez które przekierowywana jest wiadomość, zanim trafi do docelowego odbiorcy. Zasada działania polega na tym, że nadawca wiadomości pobiera listę węzłów pośredniczących, a następnie wybiera kilka spośród nich (węzły te będą tworzyć obwód przez który będzie pośredniczona przesyłana wiadomość). Następnie szyfruje on wielokrotnie dane, które mają trafić do odbiorcy. Szyfrowanie odbywa się za pomocą kluczy kolejnych węzłów tworzących obwód w odwrotnej kolejności niż się w nim znajdują (najpierw do szyfrowania zostaje użyty klucz pierwszego węzła, a na końcu klucz węzła znajdującego się na końcu obwodu). Zaszyfrowana wiadomość zostaje wysłana do pierwszego węzła, który odszyfrowuje ją, co powoduje odkrycie następnego węzła do którego ma trafić wiadomość. Wiadomość zostaje przekazywana do kolejnych węzłów, które zdejmują kolejne warstwy szyfru, do momentu aż trafi do ostatniego węzła, który zdejmuje najgłębszą warstwę szyfru i przekazuje wiadomość w oryginalnej postaci do docelowego odbiorcy. Dzięki takiemu systemowi przesyłania pakietów niemożliwe jest ustalenie całej trasy przesyłanego pakietu. Każdy z węzłów zna tylko dwóch swoich sąsiadów, pierszy węzeł zna nadawcę, lecz nie wie jaka jest treść przesyłanej wiadomośći, a ostatni zna odbiorcę, ale nie zna nadawcy wiadomości.

Rysunek 2.1 przedstawia wygląd wiadomości, która zostaje przesłana przez obwód składający się z trzech węzłów. Kolor niebieski oznacza warstwę szyfru utworzoną za pomocą klucza węzła, znajdującego się na początku obwodu, a czerwony warstwę szyfru utworzoną za pomocą klucza ostatniego węzła. Najgłębsza (szara) warstwa wiadomości oznacza niezaszyfrowane dane, które mają trafić do docelowego hosta.

## 2.1 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 2.2 przedstawia

[24]https://upload.wikimedia.org/wikipedia/commons/e/e1/Onion_diagram.svg




Rysunek 2.1: Diagram przedstawiający przykładową wiadomość przesyłaną przez obwód.[24]

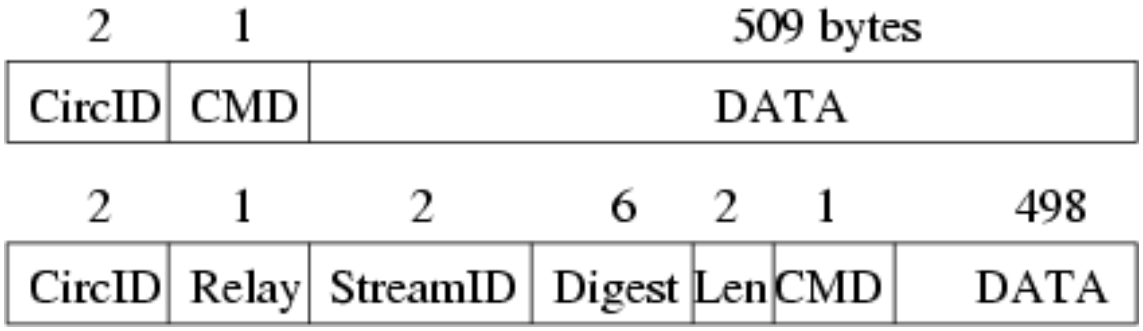
wygląd komórki.

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[25] i każdy węzeł[26] 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 identy-

[25]czym jest obwód
[26]czym jest węzeł




Rysunek 2.2: Struktura komórki Trasowania Cebulowego typu kontrolnego (u góry) oraz przekazującego.[28]

fikator strumienia, który jest wymagany ze względu na to, że pojedyńczy strumień może zostać zmultipleksowany na wiele obwodów i 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 określa odpowiednie polecenie komórki przekazującej. Do wykorzystywanych poleceń należą:

- *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

[28]https://svn.torproject.org/svn/projects/design-paper/tor-design.html

## 2.2 Proces tworzenia obwodu

Każde dwa węzły w obwodzie są ze sobą połączone przy użyciu protokołu TLS. Proces tworzenia obwodu przebiega w sposób iteracyjny. Załóżmy, że Alice jest nadawcą znajdującym się na początku obwodu, pierwszym węzłem jest Bob, a drugim w kolejności Carol:

1. Alice w celu utworzenia obwodu wysyła

## 2.3

# 3 Trasowanie Cebulowe

## 3.1 Ogólna zasada działania

- Sieć Tor składa się z grupy węzłów pośredniczących, zwanymi Routerami Cebulowymi. Każdy z tych węzłów utrzymywany jest przez ochotników.
- Sieć Tor znajduje się pomiędzy Cebulowym Proxy (klientem), a serwerem docelowym.
    1. OP pobiera listę wszystkich serwerów, wraz z ich kluczami publicznymi (kluczami cebulowymi)
    2. OP wybiera losowo kilka serwerów[29], przez które będzie pośredniczony ruch
    3. OP szyfruje przesyłaną wiadomość za pomocą kluczy cebulowych w kolejności odwrotnej niż ta w której znajdują się węzły (najpierw ostatni w obwodzie, a na końcu pierwszy z nich)
    4. OP wysyła wiadomość do pierwszego węzła w obwodzie
    5. pierwszy węzeł odszyfrowuje wiadomość za pomocą swojego klucza prywatnego, tym samym odsłaniając adres kolejnego węzła do którego ma trafić wiadomość
    6. pierwszy węzeł wysyła wiadomość do drugiego węzła, który odsłania adres kolejnego
    7. drugi węzeł wysyła wiadomość do kolejnego
    8. proces jest powtarzany do momentu, kiedy wiadomość trafi do ostatniego węzła
    9. ostatni węzeł odszyfrowuje wiadomość za pomocą swojego klucza prywatnego, i wysyła ją do serwera docelowego w oryginalnym stanie
- Podsumowanie
    -
- Sieć Tor/Trasowanie Cebulowe

[29] ile?

  - Ogólna zasada działania
  - Tworzenie połączeń pomiędzy Routerami
  - Struktura i opis komórki

- Ukryte serwisy
  - cdn...

# 4 Zasada działania

## 4.1 Tor: The Second-Generation Onion Router[30]

Streszczenie

- Nie wymaga modyfikacji jądra
- Nie wymaga specjalnych uprawnień
- Zapewnia kompromis pomiędzy zachowaniem anonimowości, użytecznością, oraz wydajnością
- Działa "w prawdziwym Internecie"

Przegląd (zmiany od generecji 1)

- Tor to nakładka na sieć, która powstała w celu anonimizacji aplikacji opartych na protokole TCP (przeglądanie sieci, ssh, wiadomości)
- Klient wybiera losowe przekaźniki w sieci, które przekierowują ruch.
- Każdy przekaźnik zna tylko swojego poprzednika, oraz następnika (nie można określić całej ścieżki pakietu, nadawcy i odbiorcy)
- Pakiet jest przesyłany w komórkach o stałej wielkości, które są rozpakowywane w kolejnych węzłach (podobnie jak cebula, stąd trasowanie cebulowe), i przesyłane dalej
- "Perfect forward secrecy"
  - Inicjator negocjuje klucz sesji z każdym sukcesywnym skokiem w obwodzie, zamiast przesyłać wielokrotnie zaszyfrowaną wiadomość
  - Nie potrzebna jest detekcja rejestracji ruchu
  - Bardziej niezawodne tworzenie obwodów (inicjator wie kiedy skok zawodzi i może rozszerzyć do nowego węzła
- Separacja "czyszczenia protokołu" od anonimowości
  - Tor używa protokołu SOCKS jako proxy na poziomie aplikacji, który wspiera większość aplikacji opartych na protokole TCP

[30] https://svn.torproject.org/svn/projects/design-paper/tor-design.html

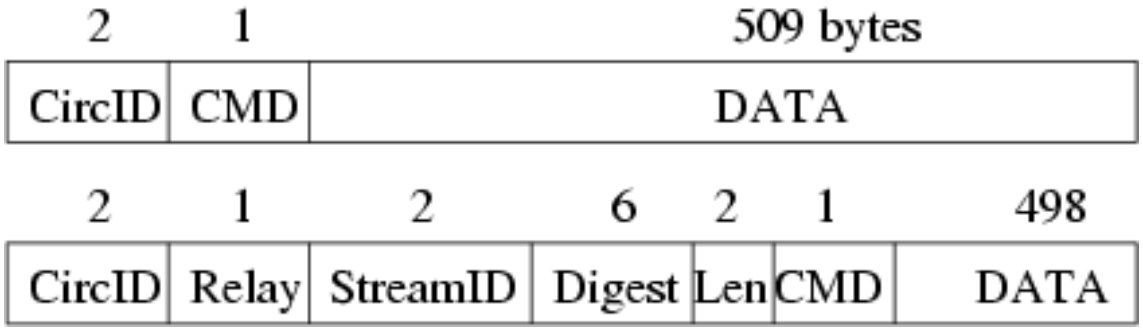
- Opiera się na funkcjach serwerów proxy działających na poziomie aplikacji, pozwalających zwiększyć prywatność, takich jak Privoxy[31]
- Możliwość udostępnienia wielu strumieni TCP, poprzez jeden zbudowany obwód (nie potrzeba negocjować wielu kluczy publicznych dla każdego połączenia, co daje większą efektywność i bezpieczeństwo)
- Sprawdzanie integralności przesyłanych danych
- Punkty spotkania i ukryte serwisy

Zasada działania

- Tor jest nakładką na sieć
- Każdy router cebulowy (ang. onion router (OR)) działa jako naromalny proces w przestrzeni użytkownika, bez żadnych specjalnych uprawnień
- Każdy router cebulowy utrzymuje połączenie TLS z każdym innym routerem cebulowym, z użyciem kluczy emeferycznych[32]
- Każdy użytkownik ma uruchomione lokalne oprogramowanie nazywane cebulowym proxy (ang. onion proxy (OP)), które pobiera katalogi, tworzy obwody, oraz utrzymuje połączenia od aplikacji użytkownika
- Każdy OR utrzymuje dwa klucze
  - Długoterminowy klucz tożsamościowy
    - Podpisywanie certyfikatów TLS
    - Podpisywanie opisu routera OR (podsumowanie o jego kluczach, adresach, przepustowości, polityce wyjścia, itd.)
    - Podpisywanie katalogów (przez serwery katalogowe
  - Krótkoterminowy klucz cebluowy
    - Używany do odszyfrowania zapytać użytkowników, aby skonfigurować obwód i wynegocjować klucze emfiryczne
  - Protokół TLS ustanawia krótkoterminowy klucz łącza podczas komunikacji pomiędzy OR.

[31] napisac cos o Privoxy

[32] Co to są klucze emeferyczne?




Krótkoterminowe klucze są okresowo i niezależnie rotowane w celu ograniczenia wpływu na ujawnienie klucza

## 4.2 Komórki

Są jednostkami komunikacji w sieci Tor

Rozmiar komórki: 512 bajtów

Składa się z nagłówka, oraz treści

Nagłówek

CircID (2 bajty) określa do którego obwodu odnosi się komórka (wiele obwodów może być zmultipleksowanych w jednym połączeniu TLS)

Komenda(1 bajt) opisuje co należy zrobić z treścią komórki, na podstawie komendy można określić typ komórki:

Komórki kontrolne są interpretowane przez węzeł

padding - używane do utrzymywania połączeń

create - używane do ustanawiania nowych obwodów

destroy - używane do niszczenia obwodów

Komórki przekaźnikowe przenoszą dane z jednego do drugiego końca strumienia

Posiadają na początku treści komórki (CircID + CMD + 11 bajtów) dodatkowy nagłówek (nagłówek przekaźnikowy)

streamID (2 bajty) - identyfikator strumienia (wiele strumieni może być zmultipleksowanych w jednym obwodzie)

suma kontrolna (6 bajtów)

długość treści (2 bajty)

komenda "przekaźnikowa" (1 bajt)

relay data - dla danych przesyłanych w strumieniu

relay begin - aby otworzyć strumien

relay end - aby czysto zamknąć strumien

relay teardown - aby zamknąć uszkodzony strumien

relay connect(ed) - aby powiadomić OP, że powiodło się rozpoczęcie przekazania

relay extend(ed) - aby rozszerzyć obwód, oraz do potwierdzania tego

relay truncate - do zamknięcia części obwodu

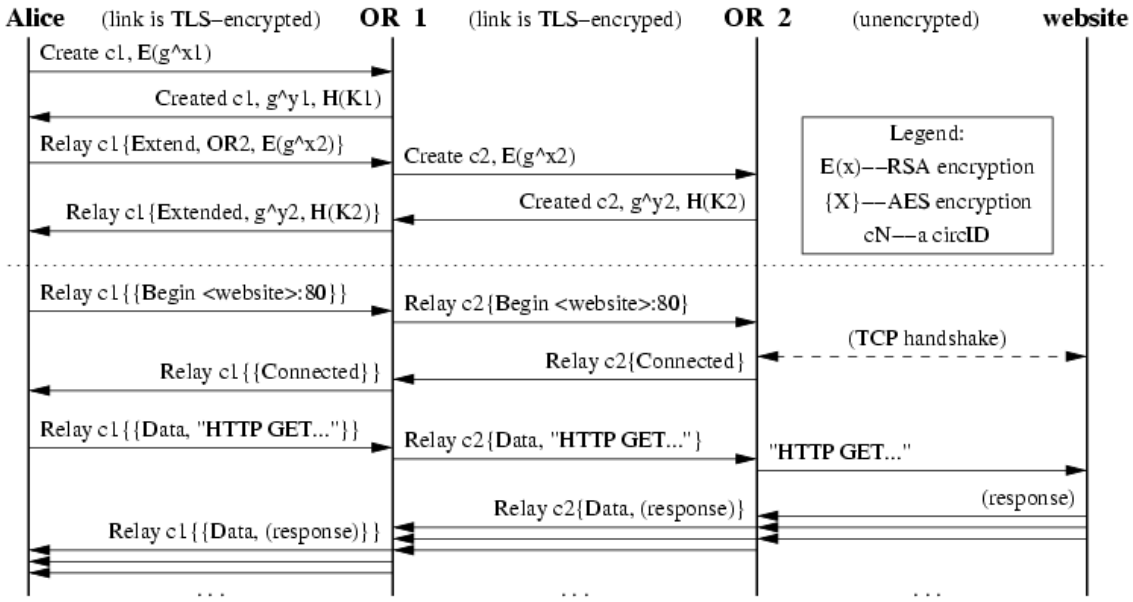
relay sendme - do kontroli przeciążenia

relay drop - atrapy dalekiego zasięgu

Zawartość nagłówka i treść komórki przekaźnikowej są szyfrowane i odszyfrowywane razem w trakcie przekazywania w obwodzie, przy użyciu 128-bitowego szyfru AES w trybie licznika, do wygenerowania "strumienia szyfru?" (ang. cipher stream).

## 4.3 Obwody i strumienie

- Jeden strumien może być współdzielony przez wiele strumieni TCP (podczas połączenia z witryną internetową tworzonych jest wiele połączeń TCP, co powoduje opóźnienia)
- OP okresowo tworzy nowe obwody, jeśli poprzednie zostały użyte, a te stare już wygasły i nie mają już żadnych otwartych strumieni.
- OP rozważają zbudowanie nowego obwodu co minutę.
- Budowanie obwodów odbywa się w tle (odbudowa uszkodzonego tworzenia obwodu odbywa się bez szkody dla użytkownika)
- Budowanie obwodu




OP buduje obwód w sposób przyrostowy (z każdym skokiem jest negocjowany klucz symetryczny z każdym OR)

Proces tworzenia obwodu (Alice (OP) –> Bob (OR1) –> Carol (OR2) –> ...)

1. (A -> B) Alice wysyła komórkę typu *create* do pierwszego węzła (Boba)
   - Alice wybrała nieużywany w połączeniu od niej do Boba circID $C_{AB}$
   - Treść (payload) komórki zawiera pierwszą połowę procesu uzgadniania klucza za pomocą protokołu Diffiego-Hellmana $g^x$, zaszyfrowaną do klucza cebulowego (publicznego) Boba
2. (B -> A) Bob odsyła odpowiedź Alice w komórce typu *created*, zawierającej drugą połowę procesu uzgadniania klucza $g^y$, wraz z hashem wynegocjowanego klucza $K = g^{xy}$
3. Obwód zostaje ustanowiony
4. Alice i Bob mogą wysyłać do siebie komórki typu *relay extended*, zaszyfrowane za pomocą wynegocjowanego klucza (obecnie klucz ten jest używany do dzielenia się dwoma kluczami symetrycznymi, dla każdego kierunku komunikacji)
5. (A -> B) Alice wysyła komórkę typu *realy* do Boba.
   - Określany jest adres następnego węzła (Carol)
   - Wysyłana jest pierwsza część zaszyfrowanego procesu uzgadniania klucza DH $g^{x_2}$

6. (B -> C) Bob kopiuje $g^{x_2}$ do komórki typu *create* i wysyła ją do Carol w celu rozszerzenia obwodu
   - Bob wybiera nowy, nieużywany do połączenia pomiędzy nim a Carol circID $C_{BC}$
   - Bob powiązuje $C_{AB}$ z $C_{BC}$
7. (C -> B) Carol opowiada komórką typu *created*
8. (B -> A) Bob pakuje treść (payload) komórki w komórkę typu *relay extended* i wysyła ją z powrotem do Alice
9. Obwód jest rozszerzony o węzeł Carol. Alice i Carol współdzielą wspólny klucz $K_2 = g^{x_2 y_2}$
10. W celu rozszerzenia obwodu, Alice powtarza powyższy proces zawsze wysyłając wiadomość do ostatniego węzła w obwodzie

Protokół realizuje jednostronne uwierzytelnianie (OR nie wie kto otwiera obwód, Alice nie używa klucza publiczego i pozostaje anonimowa)

## 4.4 Sprawdzanie integralności w struminiach

W poprzedniej wersji Trasowania Cebulowego nie była sprawdzana integralność plików, co sprawiało, że osoba atakująca mogła zmienić typ komórki na *destroy*, zmienić adres przeznaczenia na swój serwer, lub zmienić komende FTP na np. rm *

Integralność jest sprawdzana tylko na krawędziach strumienia (krawędzią strumienia może być każdy skok w obwodzie)

Kiedy Alice negocjuje klucz z każdym nowym skokiem, każdy z nich inicjalizuje skrót SHA-1 z pochodną tego klucza (ang. SHA-1 digest with a derivative of that key), rozpoczynając od losowych wartości, które są znane tylko przez dwóch z nich

Każdy z nich dodaje przyrostowo do skrótu SHA-1 zawartości wszystkich komórek typu *relay*, które stworzyli i dołączają z każdą komórką *relay* pierwsze 4 bajty obecnego skrótu (w celu zmniejszenia narzutu)

Poza tym każdy zachowuje skrót SHA-1 otrzymanych danych, aby zweryfikować, czy otrzymane hashe są prawidłowe

Jeśli OP lub OR otrzyma zły hash, to obwód zostaje zniszczony (ang. tear down)

## 4.5 Ograniczenia szybkości i uczciwości

Ochotnicy chętniej uruchamiają serwery, których możliwe jest ograniczenie przepustowości

Serwery Tor używają algorytmu Token Bucket w celu wymuszenia długoterminowych średnich szybkości przychodzących bajtów, podczas gdy stale zezwalają na krótkoterminowe serie przekraczające dozwoloną przepustowość

W protokole Tor liczba wychodzących bajtów jest mniej więcej taka sama jak liczba przychodzących bajtów, a więc w praktyce ogranicza się tylko liczbę przychodzących bajtów

## 4.6 Kontrola przeciążenia

Jeśli wystarczająca liczba użytkowników wybierze te same połączenie OR <–> OR dla swoich obwodów, wtedy to połączenie zostanie przeciążone

Rozwiązanie

- Throttling na poziomie obwodu
  - Każdy OR śledzi dwa okna
    - *packaging window* śledzi jak dużo komórek przekazywania danych OR może spakować (od przychodzących strumieni TCP) do przesłania z powrotem do OP
    - *delivery window* śledzi jak dużo jest chętnych komórek przekazywania danych (ang. relay data cells) do dostarczenia do strumieni TCP na zewnątrz sieci

    1. Zainicjalizowanie obu okien (np. do 1000 komórek danych)
    2. Kiedy komórka danych jest zapakowana lub dostarczona, zostaje zmniejszona wartość odpowiedniego okna
    3. Kiedy OR odbierze wystarczająco dużo komórek danych (obecnie 100), wysyła komórkę *relay sendme* w kierunku OP, z ustawionym streamID na zero.

4. Kiedy OR odbierze komórkę *relay sendme* z ustawionym streamID na zero, zwiększa wartość swojego okna pakowania, każda z tych komórek zwiększa wartość odpowiedniego okna o 100.
5. Jeśli okno pakowania osiągnie warość 0, OR przestaje odczytywać z połączeń TCP dla wszystkich strumieni odpowiedniego obwodu i nie wysyła już żadnych komórek przekazywania danych aż do momentu odebrania komórki *relay sendme*
6. OP zachowuje się idenycznie, jednak musi śledzić okna dla każdego OR w obwodzie. Jeśli okno pakowania osiągnie 0, to odczytywanie ze strumieni przeznaczonych dla tego OR zostaje zatrzymane

- Throttling na poziomie strumienia
  - Każdy strumień zaczyna się oknem pakowania (obecnie 500 komórek), i zwiększ jego wartość o wartość 50, aż do otrzymania komórki *relay sendme*
  - Kontrola zatorów na poziomie strumienia sprawdza także, czy dane zostały pomyślnie umieszczone w strumieniu TCP
  - Komórka *realy sendme* jest wysyłana tylko jeśli liczba bajtów oczekujących na *spłukanie* (do strumienia) jest poniżej pewnego progu (obecnie wartość 10 komórek)

Ogólna zasada działania + odnośniki do obrazków

Używane protokoły

Warstwy transportowej (TCP, czy UDP)

Warstwy aplikacji (SOCKS)

serwery filtrujące (Privoxy)

serwery pośredniczące

algorytm szyfrujący

długość kluczy szyfrujących

obrazki ilustrujące zasadę działania

## 4.7 Ukryte serwisy

Nie potrzeba używać firewalla

Specjalna pseudodomena .onion

Losowo wygenerowane adresy serwisów

## 4.8 Użycie sieci Tor na systemie Linux i/lub Windows

Kolejne kroki + output z terminala

Dowód działania

## 4.9 Utworzenie ukrytego serwisu w systemie Linux

Kolejne kroki + output z terminala

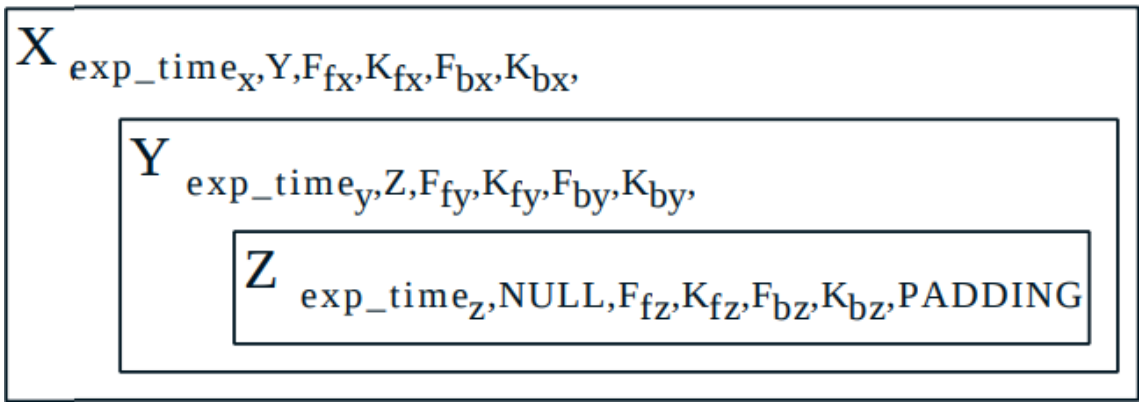
Dowód działania

## 4.10 Przyłączenie się do serwerów pośredniczących

Kolejne kroki + output z terminala

Dowód działania

## 4.11 Trasowanie Cebulowe




# 5 Hiding Routing Information

## 5.1 Wprowadzenie

- Ogranicza podatności sieci na analizę ruchu sieciowego
- Ukrywa informacje dotyczące routowania

## 5.2 Cebule

- W celu rozpoczęcia sesji pomiędzy inicjatorem i odbiorcą, proxy inicjatora identyfikuje serie węzłów routujących, które tworzą ścieżkę przez sieć i tworzą *cebulę*, która enkapsuluje tą sieżkę.
- Rysunek przedstawia cebulę skonstruowaną przez węzeł Proxy/Routujący W, do węzła Proxy/Routującego Z. Ścieżka przechodzi przez węzły X, oraz Z.
- Cały proces trasowania cebulowego (W –> X –> Y –> Z)
  - Struktura cebuli (kolejność warstw szyfru, kolejność węzłów, itd.)
  - Każdy węzeł wie kto przysłał mu wiadomość do kogo ma trafić, ale nie zna źródła i celu samej wiadomości
  - Rysunek przedstawia format wiadomości otrzymanej przez węzeł $P_x$. $\{exp\_time, next\_hop, F_f, K_f, F_b, K_b, payload\}_{PK_x}$, gdzie $PK_x$ jest kluczem publicznym węzła $P_x$, który posiada klucz deszyfrujący, exp_time to czas wygaśnięcia cebuli.
  - Wiadomość jaką otrzymuje węzeł $P_x$ wygląda następująco $\{exp\_time, next\_hop, F_f, K_f, F_b, K_b, payload\}_{PK_x}$

    * $PK_x$ szyfrujący klucz publiczny węzła $P_x$
    * *exp_time* - czas wygaśnięcia cebuli, używany w celu wykrycia
    * następny węzeł do którego mają trafić dane
    * dane
    * dwie pary funkcji/kluczy, określających operacje kryptograficzne i klucze, które mają być zastosowane do danych przesyłanych przez wirtualny obwód. Para $(F_f, K_f)$ jest zastosowana do danych przechodzących na przód, a para $(F_b, K_b)$ jest zasotosowana do przeciwnego kierunku.
  - Każdy przeskok pomniejsza cebulę poprzez zdjęcie z niej warstwy. Aby uniknąć wykrycia miejsca węzła w trasie, na końcu danych umieszcza się losowy ciąg bitów o wielkości zdjętej warstwy, przed wysłaniem wiadomości do następnego węzła. Dzięki temu tylko ostatni węzeł wie jak duże jest dopełnienie wiadomości, ponieważ znajduje się na końcu trasy i o tym wie.

## 5.3 Tworzenie obwodu

- Wiadomość zawiera id obwodu, komendę (create, destroy i data), oraz dane
- Węzeł, który otrzyma wiadomość zawierającą komendę *create* wybiera id wirtualnego obwodu i wysyła kolejną wiadomość *create* zawierającą ten id do następnego węzła. Węzeł przechowuje id wirtualnego obwodu, który otrzymał i id wirtualnego obwodu jaki wysłał, jako parę.
- funkcja kryptograficzna i klucz kryptograficzny są stosowane do danych przesyłanych w przód obwodu, oraz w przeciwnym kierunku. Dla każdego kierunku stosowane są oddzielne pary funkcja/klucz.
- rysunek 3. przedstawia wirtualny obwod dla cebuli widocznej na rysunku 2.
- Proxy inicjatora szyfruje wysyłaną wiadomość stosując odwrotną kolejność operacji kryptograficznych. Wiadomość znajduje się najgłębiej w cebuli.

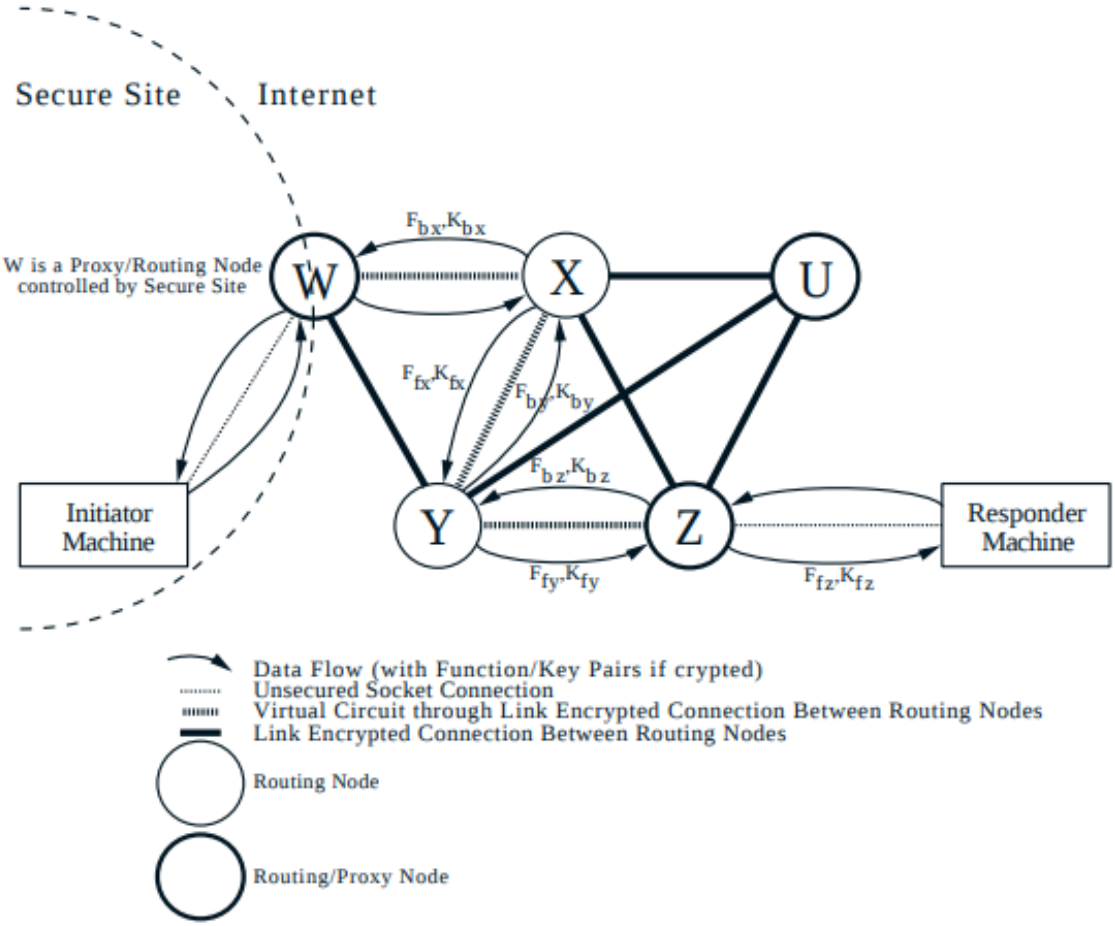



**Fig. 3.** A Virtual Circuit.

## 5.4 Luźny routing

- Proxy inicjatora nie musi okereślać całej ścieżki. Zamiast tego nakazać konkretnemu węzłowi, aby ten sam wybrał trasę do następnego węzła. Dzięki temu w obwodzie znajduje się więcej przeskoków, co ma wpływ na wzrost bezpieczeństwa. Jeśli proxy inicjatora nie zna kompletnej ścieżki do punktu docelowego, wtedy węzeł może zbudować ścieżkę do następnego węzła. Luźny routing może być wykorzystany obsługi zmian połączenia, krórego proxy inicjatora nie jest świadomy.

- Możliwe jest również zezwolenie węzłom w dodanej ścieżce na użycie tego procesu, ale potrzebny jest mechanizm, który ochroniłby przed tworzeniem ścieżki o nieskończonej długości. Cebula, która pozwala na użycie luźnego routingu wygląda następująco:

  $\{exp\_time, next\_hop, max\_loosecount, F_f, K_f, F_b, K_b, payload\}_{PK_x}$

- Jeśli węzeł, który odbierze cebulę, zadecyduje o użyciu luźnego routingu, przygotowuje nową cebulę z maksymalnie max_loosecount ilością warstw. Danymi w tej cebuli jest po prostu odebrana cebula ze zmienionym $PK_x$ dla ostatniego (najgłębszego) węzła, którego dodał do łańcucha. Innymi słowy zachowuje się on jak proxy inicjatora, z tym wyjątkiem, że przesyłane w cebuli dane są również cebulą. W celu utrzymania stałej długości cebuli, węzeł musi obciąć ilość danych współmierną do ilości dodanych warstw. Inicjujące proxy musi przewidzieć ilość dopełnienia (zarówno obecnego początkowo, jak i każdego dodanego i/lub obciętego podczas przechodzenia przez ścieżkę), które będzie występować w centralnych danych podczas trwania luźnego routingu, które będzie zezwalać na to obcięcie. Całkowita wartość wszystkich max_loosesount występujących w dodanych warstwach plus liczba dodanych warstw musi być mniejsza lub równa wartości max_loosecount odebranej przez dodany węzeł.

## 5.5 Cebule zwrotne

- W niektórych aplikacjach przydatne jest odsyłanie wiadomości zwrotnych (odpowiedzi, ang. replay) przez serwer docelowy, podczas uszkodzenia oryginalnego obwodu. To pozwoliłoby na wysyłanie odpowiedzi (np. email) na zapytania, które nie

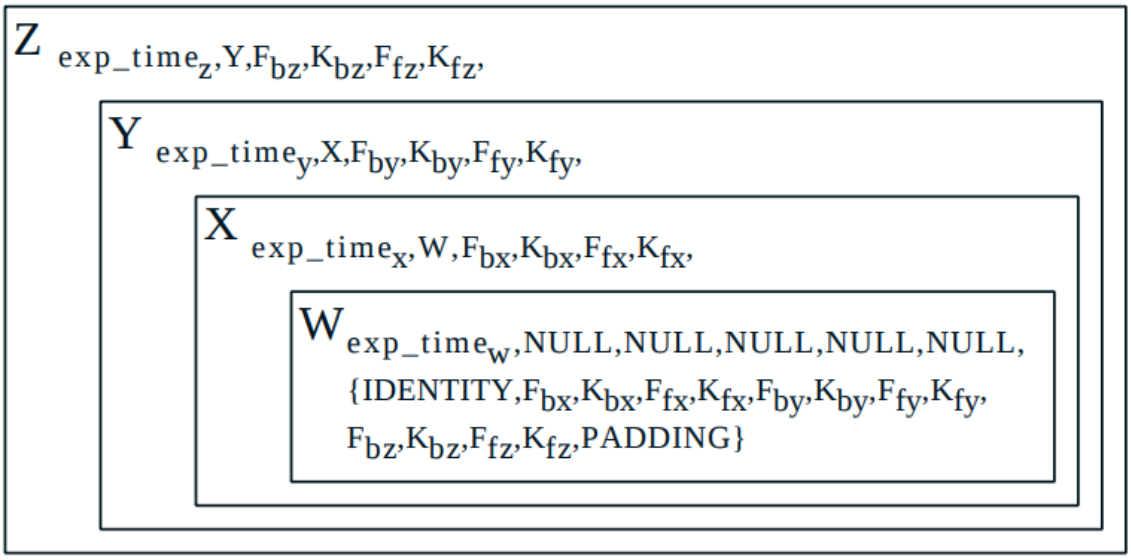



**Fig. 4.** A Reply Onion.

były dostępne podczas oryginalnego połączenia. Jak teraz zobaczymy, to również pozwala odbiorcy, tak samo jak nadawcy, na pozostanie ukrytym.

- Cebule zwrotne mają taką samą strukturę jak zwykłe cebule, a węzły traktują je w ten sam sposób.

- Główną różnicą między cebulą zwykłą (przekazującą, ang. forward), a odpowiadającą (ang. reply) jest najgłębsza treść. W cebuli przekazującej może być ona pusta (zawierająca jedynie padding (dopełnienie)). Treść cebuli zwrotnej zawiera wystarczającą ilość informacji, aby umożliwić proxy inicjatora dotarcie do inicjatora i wszystkich par funkcji kryptograficznych i kluczy z cebuli.

- Rysunek 4. przedstawia cebulę zwrotną skonstruowaną przez Węzeł Proxy/Router W inicjatora.

- Zwrotna trasa rozpoczyna się w Węźle Proxy/Routerze Z odbiorcy przez pośrednie węzły routujące Y, oraz X.

- Przekierowywujące funkcje kryptograficzne są przypisane do danych poruszających się w kierunku, w którym obwód został ustanowiony.

- Kryptograficzne funkcje zwrotne są przypisane do danych poruszających się w przeciwnym kierunku

- Lokalizacja końcowych Węzłów Proxy/Routujących jest odwrócona. (inicjator na końcu, odbiorca na początku)

- Zarówno cebula przekierowująca jak i zwrotna może być użyta tylko raz. Kiedy węzeł odbierze cebulę, utrzymuje ją aż do momentu wygaśnięcia. Każda odebrana cebula jest z nią porównywana w celu wykrycia powtórek, które są traktowane jako błędne i są ignorowane.

## 5.6 Implementacja

- Wprowadzenie
  - Tradycyjny serwer proxy działa na maszynie zapory sieciowej.
  - Jest on odpowiedzialny za przekierowywanie zapytań z chronionej domeny do Internetu i utrzymywanie ścieżki powrotnej dla odpowiedzi na zapytania.
  - Serwer proxy może być podzielony na dwie części. Front-end, który otrzymue i parsuje zapytania i back-end, który przetwarza zapytania i zwraca wynik do pytającego. Klasycznie back-end i front-end są tym samym procesem.
  - W systemie Trasowania Cebulowego back-end i front-end są oddzielnymi procesami na oddzielnych maszynach, które są połączone tunelem.

- Połączenia między węzłami
  - Wszystkie węzły są ze sobą połączone szyfrowanym połączeniem
  - Wszystkie wiadomości przechodzące przez te połączenia mają stałą wielkość
  - Każda wiadomość składa się z dwóch części, nagłówka i treści. Nagłówek zawiera id virtulanego obwodu, komendę i są szyfrowane za pomocą połączenia.
  - Wszystkie pola treści są szyfrowane za pomocą kluczy publiczych lub kluczy cebulowych, więc nie muszą być szyfrowane przez połączenie.

- Komendy
  - *create* - tworzy wirtualny obwód.
    - Każdy węzeł utrzymje dwa połączenia
    - Każde połączenie definiowane jest przez etykietę

    - Proces tworzenia obwodu polega na zdefiniowaniu tych etykiet
    - Cebula jest dzielona na części, które są przesyłane do następnego węzła i zawierają pole kontrolne, w którym znajdują się zarówno etykieta połączenia jak i komenda *create*
    - Każdy kolejny węzeł składa i obiera cebulę tym samym odsłaniając adres kolejnego węzła oraz dwie pary (funkcja/klucz).
    - Przed wykonaniem komendy *create* węzeł sprawdza, czy cebula nie jest wygasła lub czy nie jest powtórką.
    - Jeśli cebula jest prawidłowa, węzeł wstawia ją do tablicy i przypisuje nowe połączenie do następnego węzła a następnie wysyła obraną i dopełnioną cebulę do następnego węzła.
    - Węzeł również aktualizuje tablicę zawierającą etykiety i pary kryptograficzne funkcja/klucz dotyczące nowego wirtualnego obwodu.
- *data* - wysyłanie danych przez wirtulany obwód
    - Strumień danych jest podzielony na kawałki
    - Każdy kawałek jest szyfrowany za pomocą operacji kryptograficznych określonych w cebuli, w odwrotnej kolejności (najgłębsza jako pierwsza)
    - Para klucz/funkcja oraz id wirtualnego obwodu są otrzymywane z tablicy.
    - W nagłówku znajdują się etykieta połączenia i komenda *data*
    - Każdy kolejny węzeł wyciąga ze swojej tablicy parę funkcja/klucz powiązaną z obwodem (dla odpowiedniego kierunku) i id wirtualnego obwodu dotyczący połączenia z następnym węzłem.
    - Węzeł obiera jedną warstwę szyfru i przekierowywuje treśc do następnego węzła.
    - Ostatni węzeł zdejmuje ostatnią warstwę szyfru i wysyła ją do odbiorcy wiadomości.
- *data* w drugą stronę
    - odbiorca pobiera parę oraz id virtulanego obwodu do następnego węzła ze swoich tablic i szyfruje strumien
    - odbiorca dzieli zaszyfrowany strumień na kawałki wielkośći treści

    - odbiorca przekierowywuje do następnego węzła z odpowiednim polem kontrolnym
    - każdy kolejny węzeł szyfruje każdą treść przy użyciu odpowiedniej pary (funkcja/klucz) powiązanej z wirtualnym obwodem
    - kiedy wiadomość dotrze do węzła proxy/Routującego inicjatora, stosuje on na niej odpowiednie operacje powrotne określone w cebuli (najgłębsza na końcu), w celu otrzymania wiadomości w postaci czystego tekstu, który jest przesyłany do inicjatora

- *destroy* - zniszczenie obwodu (kiedy nie jest już dłużej potrzebny lub wystąpił pewien błąd)
    - może być użyta przez każdy z węzłów
    - węzeł który inicjuje taką komendę wysyła ją do obu swoich połączeń
    - każdy węzeł ma obowiązek przesłać taką wiadomość w odpowiednim kierunku
    - treść wiadomośći jest pusta i zawiera jedynie padding (mimo to jest dalej szyfrowana za pomocą odpowiedniej pary funkcja/klucz)
    - nagłówek oprócz komendy zawiera ponadto pole kontrolne zawierające id wirtualnego obwodu odbiorcy komendy
    - Odbiorca, który otrzymał wiadomość zawierającą komendę *destroy*, kasuje wpisy w swojej tablicy dotyczące odpowiedniego wirtualnego obwodu