Bezpieczniejsze DHT

Paweł Wiejacha

25 marca 2010

Plan prezentacji:

- Krótko o DHT
  - czym jest DHT, gdzie i czemu się jej używa
  - streszczenie protokołów Chord i Kademlia
- Ataki specyficzne dla DHT i ochrona przed nimi
  - Sybil attack
  - Eclipse attack
  - ataki typu storage i routing
- Teoria a praktyka
- Podsumowanie

Plan prezentacji:

- Krótko o DHT
  - czym jest DHT, gdzie i czemu się jej używa
  - streszczenie protokołów Chord ~~i Kademlia~~
- Ataki specyficzne dla DHT i ochrona przed nimi
  - Sybil attack
  - Eclipse attack
  - ataki typu storage i routing
- Teoria a praktyka
- Podsumowanie

Distributed Hash Table

- tablica mieszająca
- umożliwiająca przechowywanie par (klucz, wartość)
- rozproszona i zdecentralizowana
- węzły przechowują porcje danych
- zapewnia wydajne wyszukiwanie danych
- pozwala przechowywać duże ilości danych

Zalety DHT

- decentralizacja
- odporność na awarie węzłów,
- wydajne wyszukiwanie danych
- skalowalność
- niskie koszty administrowania

Zastosowania DHT

- sieci P2P
- rozproszone systemów plików,
- sieci anonimowe,
- rozproszone cacheowanie
- systemy CDN

Zastosowania DHT - sieci P2P

- BitTorent - rozproszone trakcery
- Gnutella
- Overnet
- KAD - eMule, MLDonkey

Zastosowania DHT - rozproszone systemy plików

- CFS (Cooperative File System)
- OceanStore - składowanie danych na PlanetLab
- Mnemosyne - steganograficzny system plików
- PAST

Zastosowania DHT - sieci anonimowe

- Freenet - *censorship-resistant distributed data store*
- GNUnet
- Perfect Dark - japoński anonimowy P2P

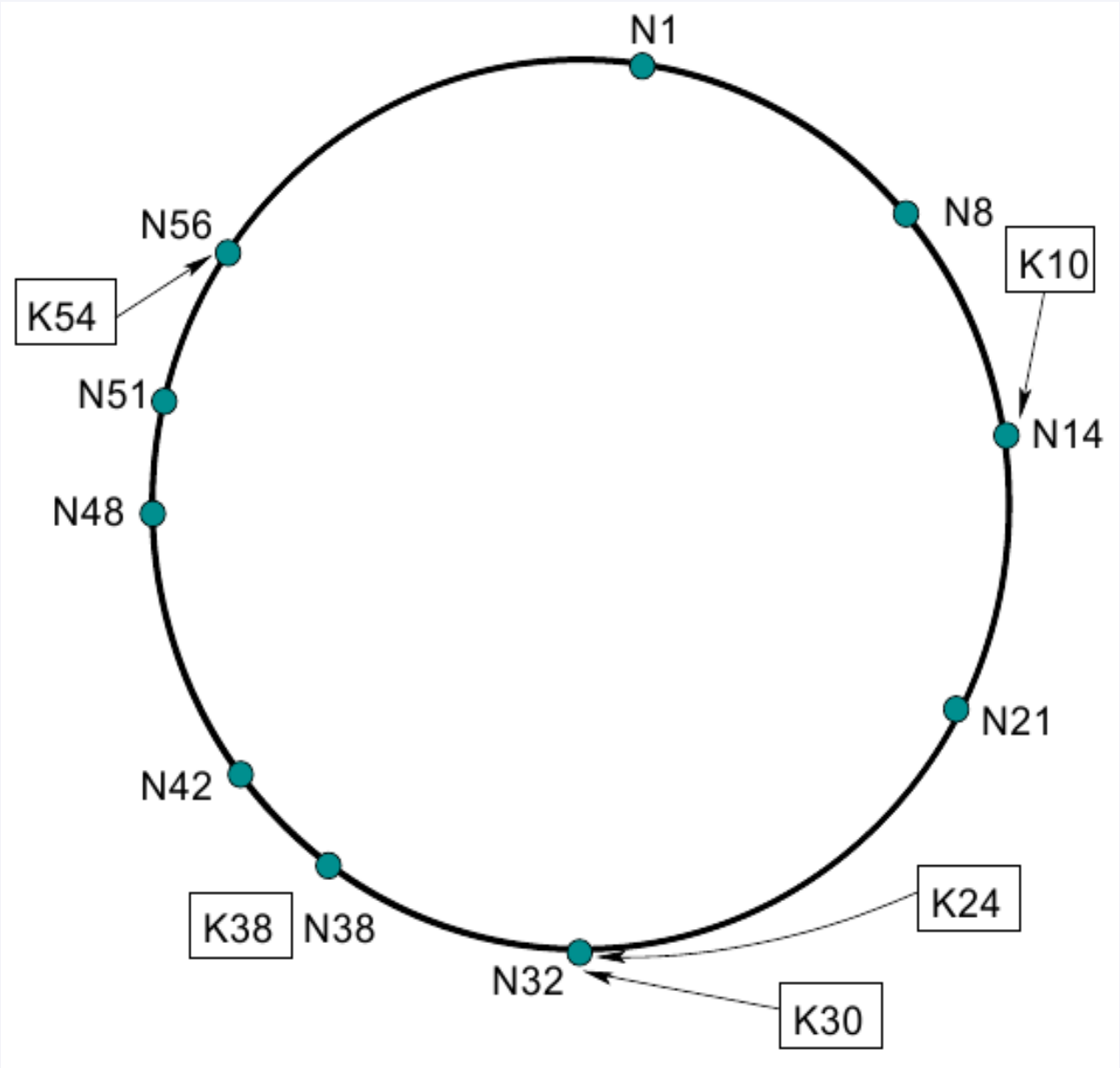
Zastosowania DHT - web cacheing i CDN

- CoDeeN - distributed webcache na PlanetLabie
- Coral Content Distribution Network - webcache/CDN

Protokół Chord - przykład DHT

- Ustalone m (np. 160). Maksymalnie 2^m węzłów
- *chordKey(key)* $=$ *mBitHashFun(key)*
- *myID* $=$ *chordKey(myIP)*
- węzły rozkładamy na okręgu $mod\ 2^m$
- za klucz K odpowiedzialny jest węzeł o $ID >= K\ mod\ 2^m$

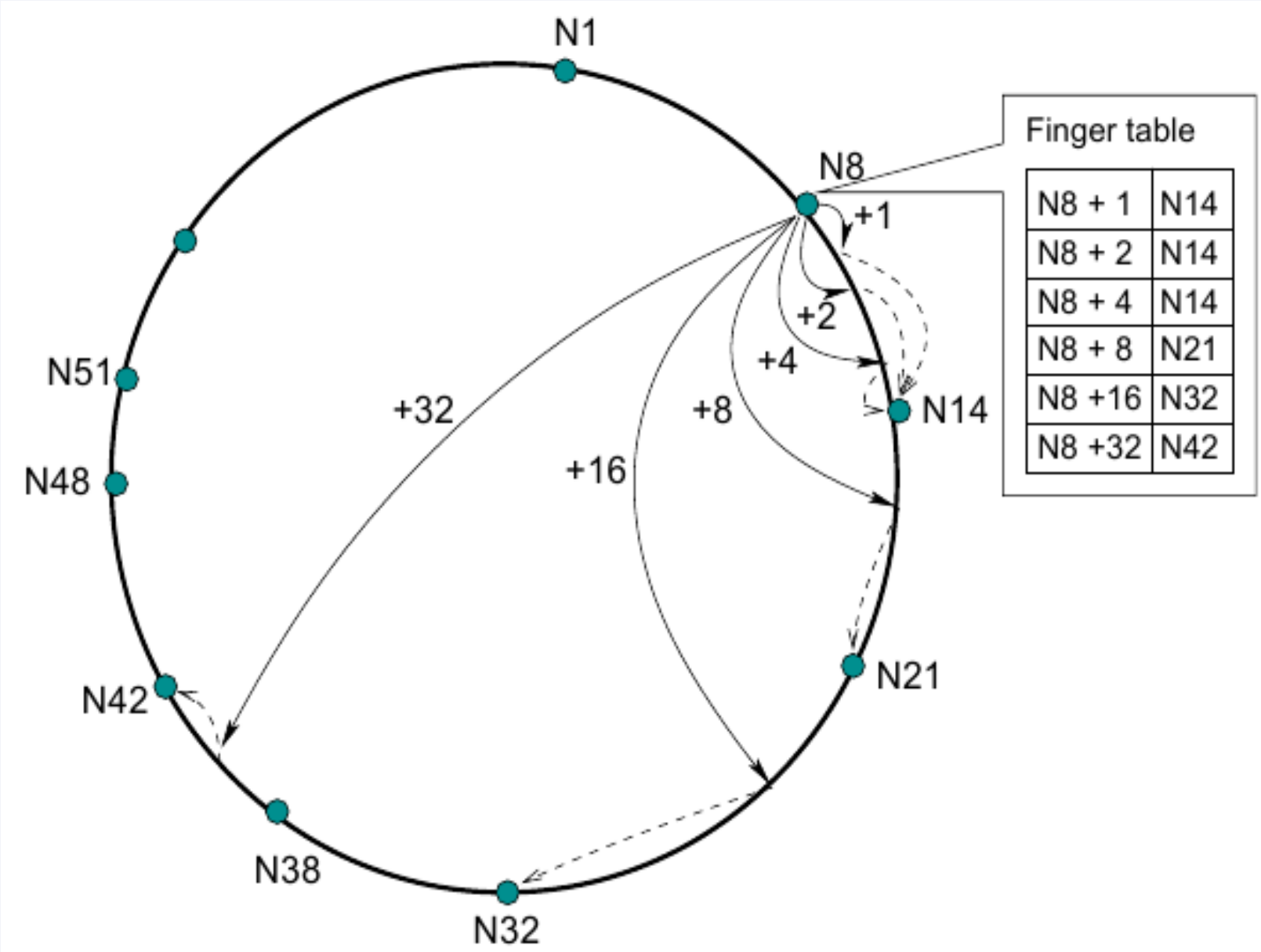
Krótko o DHT
protokół Chord (2)



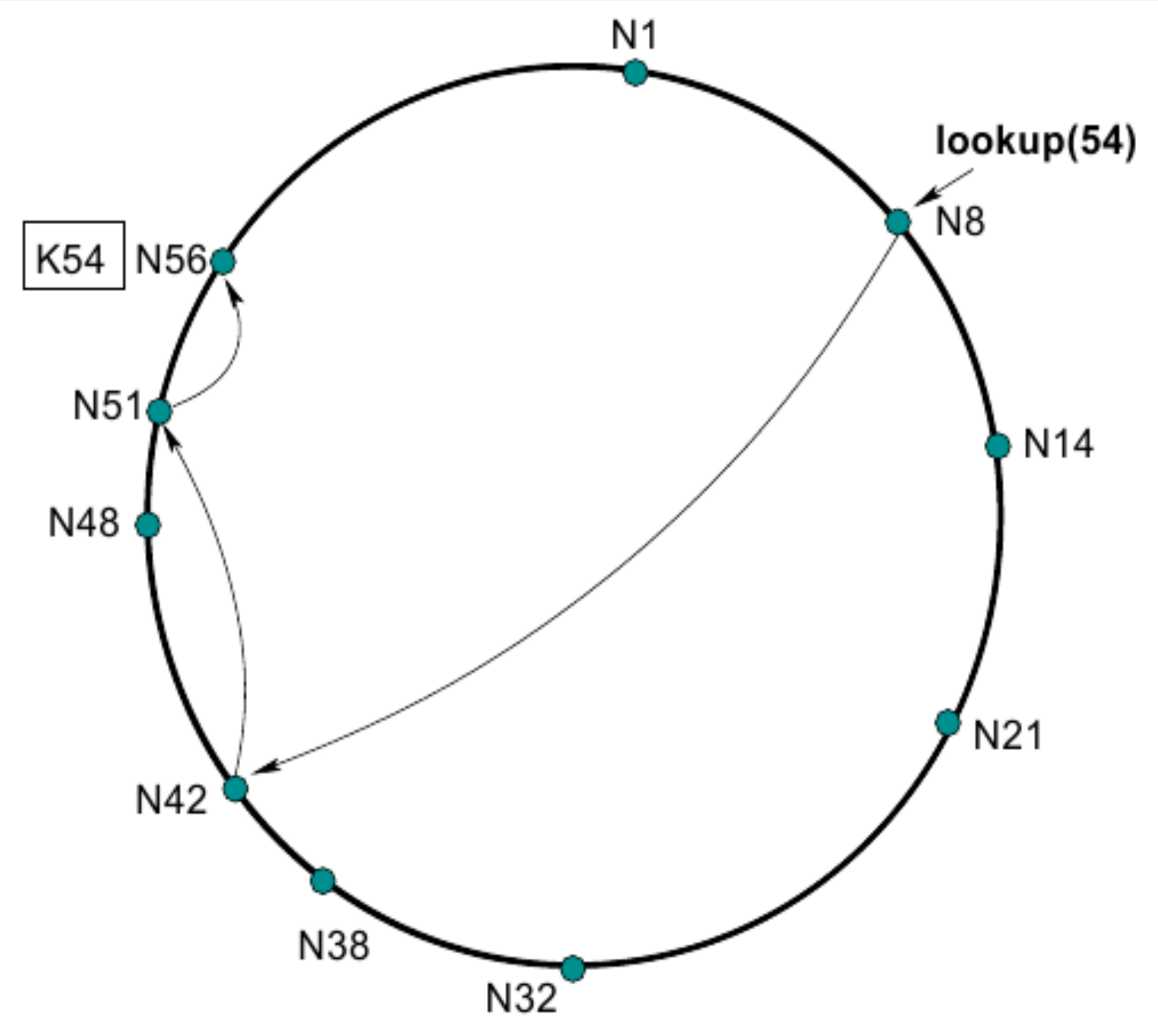
Protokół Chord

- każdy węzeł zna swojego poprzednika i następnika
- oraz ma tablicę *finger[k]* = *successor(myId+*2^k*)*
  - dla k = $1..log(m)$
- można więc wyszukiwać klucz w czasie $O(log(n))$
- a węzeł podłączyć do sieci w $O(log^2(n))$

Krótko o DHT
protokół Chord (4)



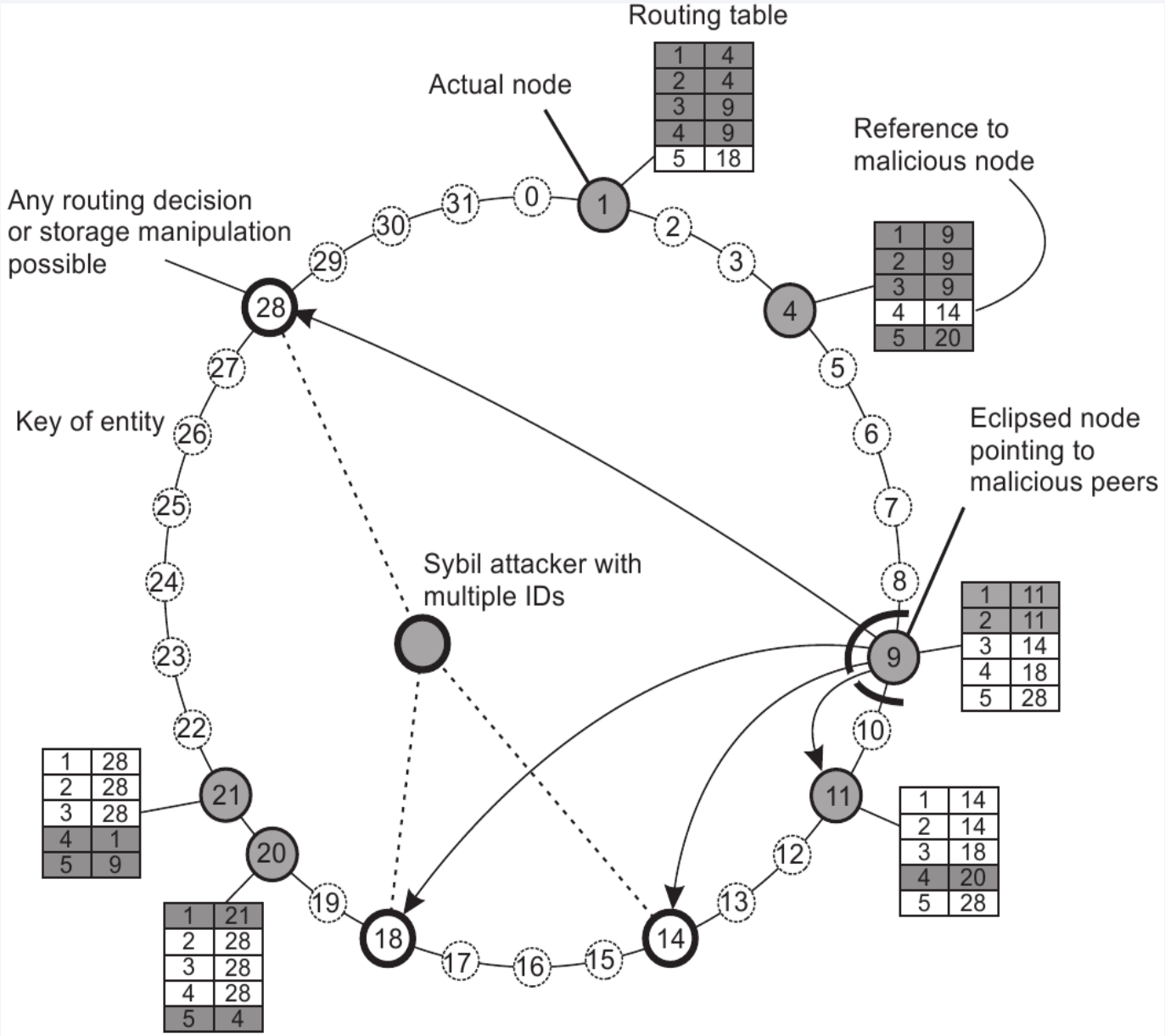
Krótko o DHT
protokół Chord (5)



Rodzaje ataków

- baza dla ataków
  - Sybil attack - jedna maszyna $\Rightarrow$ wiele węzłów
  - Eclipse attack - zatruwane tablic rutowania dobrych węzłów
- ataki właściwe
  - storage attack - zwracanie niepoprawnych danych
  - routing attack - złośliwe rutowanie

Ataki specyficzne dla DHT
ogólny zarys ataków (2)



Sybil attack - zasada działania

- jest więcej logicznych węzłów atakującego niż fizycznych maszyn
- wtedy atakujący może pokryć większą cześć sieci
- podsłuchiwać o odpowiadać na większą ilość zapytań
- to tylko podstawa dla innych ataków

Sybil attack - uwierzytelnianie węzłów

- centralna jednostka certyfikująca węzły
- wymagająca potwierdzenia fizycznej tożsamości
- lub pobierająca opłaty za potwierdzenie tożsamości
- tylko certyfikowane węzły mogą włączyć się do DHT

Uwierzytelnianie węzłów - charakterystyka

- niewątpliwie skuteczne
- administracyjny narzut
- scentralizowanie
- problemy z anonimowością
- można podpisać cyfrowo przypisanie (ID $\Rightarrow$ IP)
  - ale co z komputerami za NAT lub z dynamicznymi IP?

Sybil attack - rozproszona rejestracja

- *myID = hashFun(myIP, myPort)*
- *registrationNode(j,IP) = hashFun(j, IP)*
- rejestrujemy się u *r* węzłów *registrationNode(1..r, myIP)*
- każdy *registrationNode(_,IP)* pamięta kogo zarejestrował
  - i odrzuca prośby o rejestracje, jeśli za dużo węzłów z tego samego IP
- potrzebujemy $\frac{r}{2} + 1$ akceptacji od węzłów rejestrujących
- okresowo weryfikujemy węzły z tablic rutowania
  - sprawdzając hasze i węzły rejestrujące

Rozproszona rejestracja - charakterystyka

- działa przy założeniu, że większość węzłów jest dobra
- eksperymenty pokazały, że działa przyzwoicie
- ponownie problemy z NATem
- co jeśli mamy do czynienia z botnetem?

Sybil attack - obliczeniowe zagadki

- jeżeli jedna maszyna udaje wiele węzłów
- to zadając każdemu węzłowi problem NP zupełny
- eliminujemy nieuczciwe maszyny
- trzeba w rozproszony sposób weryfikować rozwiązania
- część rozwiązań bazuje na haszach ID i losowych liczb
  - ale jest silnie powiązana z protokołem DHT

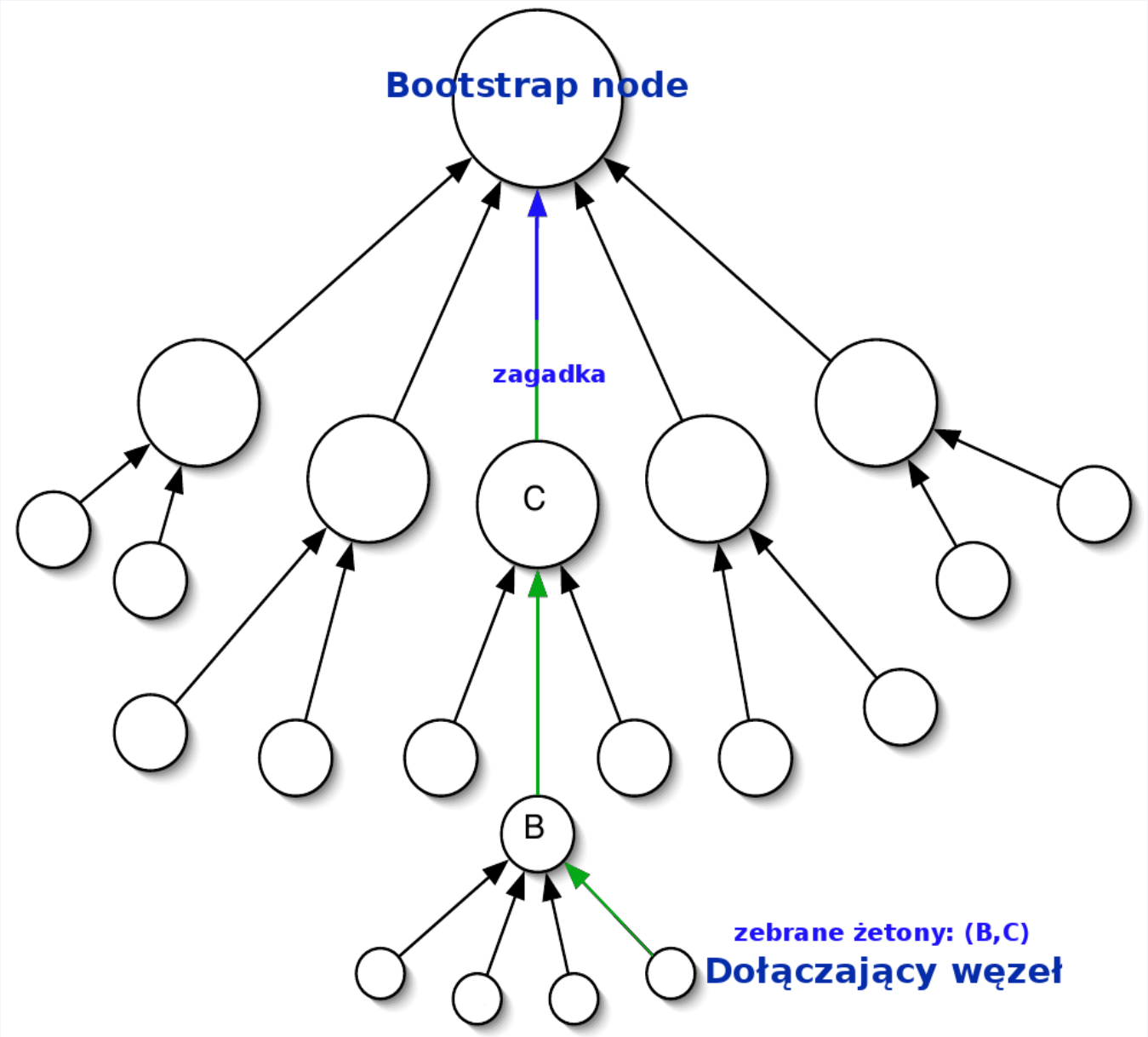
Sybil attack - hierarchiczne zagadki

- korzeń (Bootstrap node) musi być zaufany i stabilny
- X - dołączający się węzeł
- $ID(X) = hashFun(pubKey(X))$
- zagadka dla X:
  - znajdź K ($K = random(M)$)
  - znając H ($H = hashFun(K + currentTime + pubKey(X))$
- X rozwiązuje zagadki idąc od liścia do korzenia

Sybil attack - hierarchiczne zagadki

- gdy X rozwiąże zagadkę od Y,
- Y tworzy podpisany cyfrowo żeton
- X pokazuje żeton ojcu Y i rozwiązuje jego zagadkę
- X zostaje dołączy do drzewa zagadek po dłuższym czasie
- trzeba uwzględnić odłączanie się węzłów od sieci
- oraz uzależnić trudność zagadek od wielkości drzewa

Ataki specyficzne dla DHT
Sybil attack - hierarchiczne zagadki (3)



Sybil attack - hierarchiczne zagadki

- gdy atakujący ma węzeł w drzewie zagadek
  - może dawać żetony za darmo
  - da się to wykryć badając częstość przychodzących kandydatów od danego węzła
- gdy atakujący nie ma węzła w drzewie zagadek
  - jest (tylko) spowolniony
  - więc trzeba okresowo sprawdzać swoje dzieci zadając im zagadki

Hierarchiczne zagadki - charakterystyka

- zabierają czas procesora uczciwym węzłom
- ciężko dobrać zagadki w zróżnicowanym środowisku
- centralizacja
- górne warstwy drzewa powinny być pewne

Sybil attack - fizyczna charakterystyka sieci

- grupujemy węzły ze względu na "podsieć" do której należą
- tę identyfikujemy poprzez mierzenie odległości od landmarków
- odległość mierzymy w sensie fizycznym (RTT)
- złośliwe węzły nie mogą udawać że są w innej podsieci
- a tej samej podsieci zadajemy zagadki

Fizyczna charakterystyka sieci - charakterystyka

- maszyny muszą rozwiązywać zagadki
- zmiana warunków w sieci spowoduje "fałszywe pozytywy"
- zmiana warunków sieci powoduje zmianę ID
- co z urządzeniami mobilnymi?
- i botnetami?

Sybil attack - Hop-count distance

- przystosowany głównie do dużych sieci WiFi
- każdy węzeł ma klucz prywatny-publiczny
- odległosć między węzłami mierzymy w Hop-countach
- składa się z dwóch protokołów
  - stwierdzania, czy węzeł jest "fizycznym sąsiadem"
  - podpisywanego cyfrowo mierzenia odległości

Hop-count distance - fizyczni sąsiedzi

- mają możliwość broadcastowania do siebie
- X wysyła do Y losowe bity B zakodowane kluczem publicznym
- X broadcastuje niezakodowana wiadomość M
- Y wysyła (M xor B) do X
- X wie, że jakiś fizyczny sąsiad ma dostęp do klucza prywatnego Y

Hop-count distance - mierzenie odległości

- chcemy zmierzyć odległość między X i D
- nie chcemy by złośliwe węzły raportowały odległość mniejszą od faktycznej
- budujemy minimalne drzewo rozpinające
- każdy węzeł wysyła informacje o znanych mu najkrótszych ścieżkach
- fragmenty ścieżek są podpisywane cyfrowo
- fizyczni sąsiedzi potwierdzają cyfro ścieżki

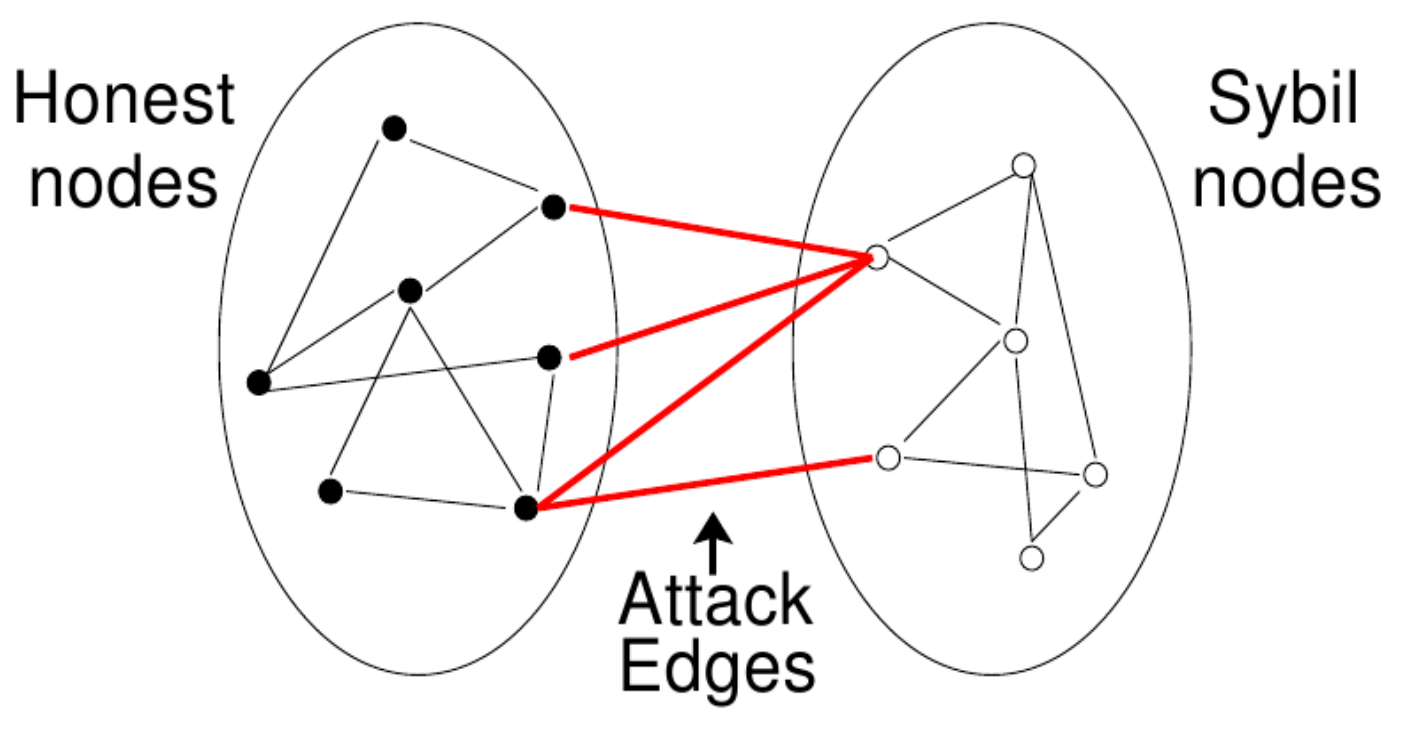
Hop-count distance - charakterystyka

- wydaje się być sensowne w sieciach WiFi
- ale jak zastosować to w zwykłych sieciach:
  - każdy węzeł jest fizycznym sąsiadem wszystkich
  - czy ograniczyć się do domeny broadcastowej?

Sybil attack - SybilGuard

- sieć społeczna: krawędzie - silna przyjaźń/więzi rodzinne
  - przyjaciele wymieniają się offline kluczami symetrycznymi
- atakujący może być połączony z dowolnie dużą ilością *złych* węzłów
- ale nie może mieć zbyt wielu krawędzi do *dobrych* węzłów
  - tzw. *attack edges*

Ataki specyficzne dla DHT
SybilGuard - sieci społeczne



Sybil attack - SybilGuard

- SybilGuard dzieli węzły na grupy, tak
- że grup które zawierają choć jeden zły węzeł (*Sybil group*)
- jest nie więcej niż g - ilość *attack edges*
  - niezależnie od ilości wirtualnych węzłów atakującego
- przy ustalonym globalnie w
  - *uczciwy* węzeł zaakceptuje co najwyżej w węzłów z jednej Sybil grupy

SybilGuard - losowe ścieżki

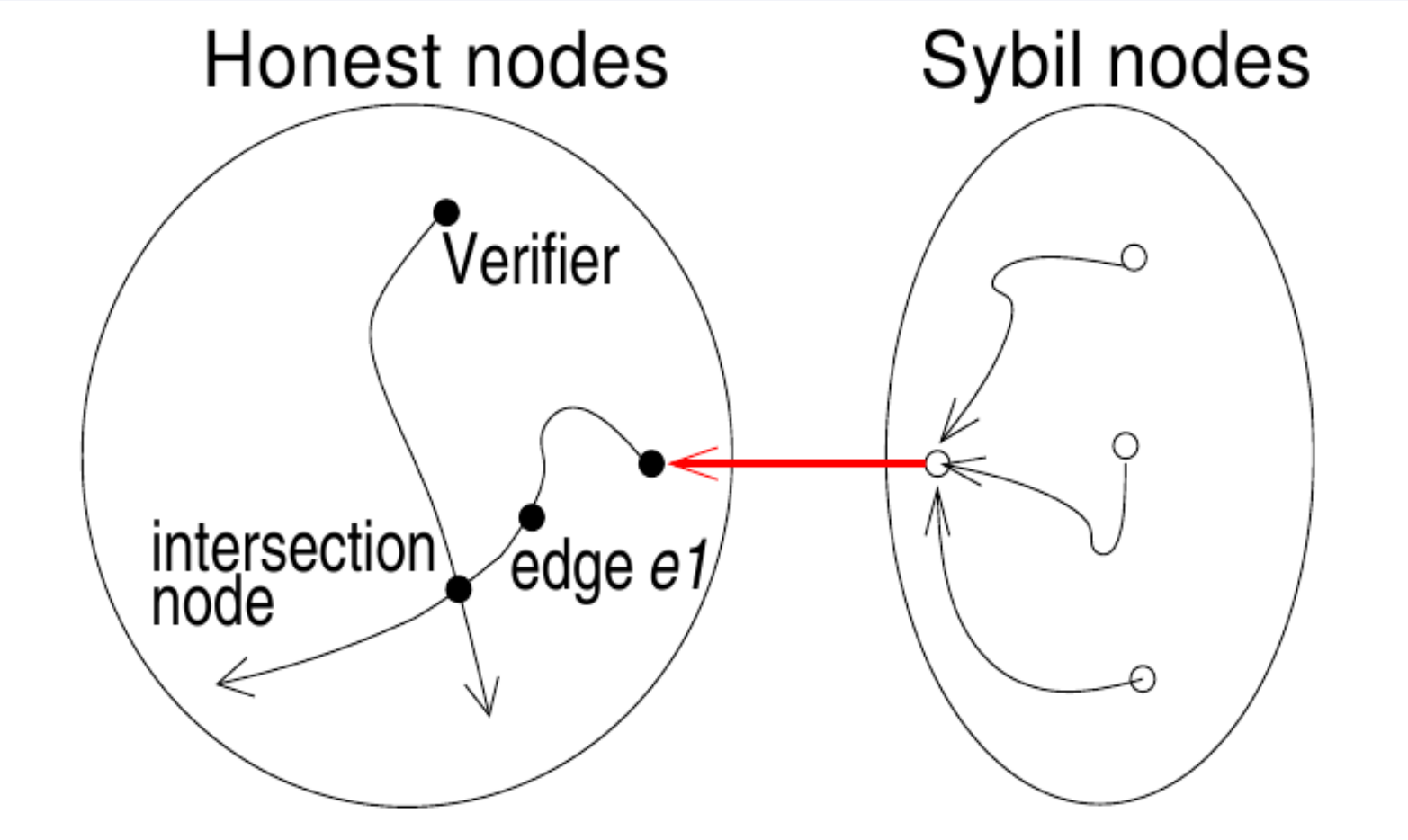
- każdy węzeł ma ustaloną losową permutację **krawędzi**
- bijekcję między krawędziami wejściowymi a wyjściowymi
- $(e2, e3, e1)$ - wszystko co wejdzie przez $e1$ wyjdzie przez $e2$
  - jeśli dwie losowe ścieżki się przecinają w *uczciwym* węźle, to dalej idą "razem"
  - mając daną krawędź wychodzącą, wiemy jaka jest krawędź wchodząca
- każdy węzeł X ma tyle losowych ścieżek co sąsiadów
  - każda ma długość w
  - każda wychodzi przez inną krawędź
  - każda zaczyna się w X

SybilGuard - działanie

- ponieważ krawędzi atakujących jest mało
- to losowe ścieżki wychodzące z *uczciwych* grup
- raczej nie będą przechodziły przez węzły atakującego

SybilGuard - weryfikacja

- Weryfikujący V sprawdza podejrzanego S czy jest uczciwy
- sprawdza czy przynajmniej połowa ścieżek V
- przecina się z jakąś ścieżką S
- jeśli tak jest to oboje są w tym samym rejonie
  - więc jeśli V jest uczciwy, to S jest uczciwy


Honest nodes
Sybil nodes
Verifier
intersection node
edge e1

SybilGuard - Weryfikacja

- wszystkie złe węzły z jednej grupy muszą przejść przez *attack edge*
- by przeciąć się ze ścieżką V
- V może je utożsamić z jedną grupą (bo e1)
- jeśli jest g atakujących krawędzi, to V zaakceptuje g złych grup

SybilGuard - Weryfikacja

- może być conajwyżej w różnych losowych ścieżek
  - które przecinają się w węźle n **oraz**
  - wchodzą do n wzdłuż krawędzi e
- więc V zaakceptuje nie więcej niż w złych węzłów z jednej grupy

Każdy węzeł trzyma:

- kluczy prywatny-publiczny
- klucz symetryczny z każdym przyjacielem
- tablicę rutowania (permutacja krawędzi)
- tablicę rejestracji
- tablicę świadków

Tablica rejestracji dla krawędzi e dla X

- i-ty wpis w tabeli to
  - klucz publiczny węzła y,
  - dla którego e jest i-tą krawędzią na losowej ścieżce
- węzły rejestrują swoje ścieżki u węzłów ścieżki
  - przekazując tablice krawędzi sąsiadom

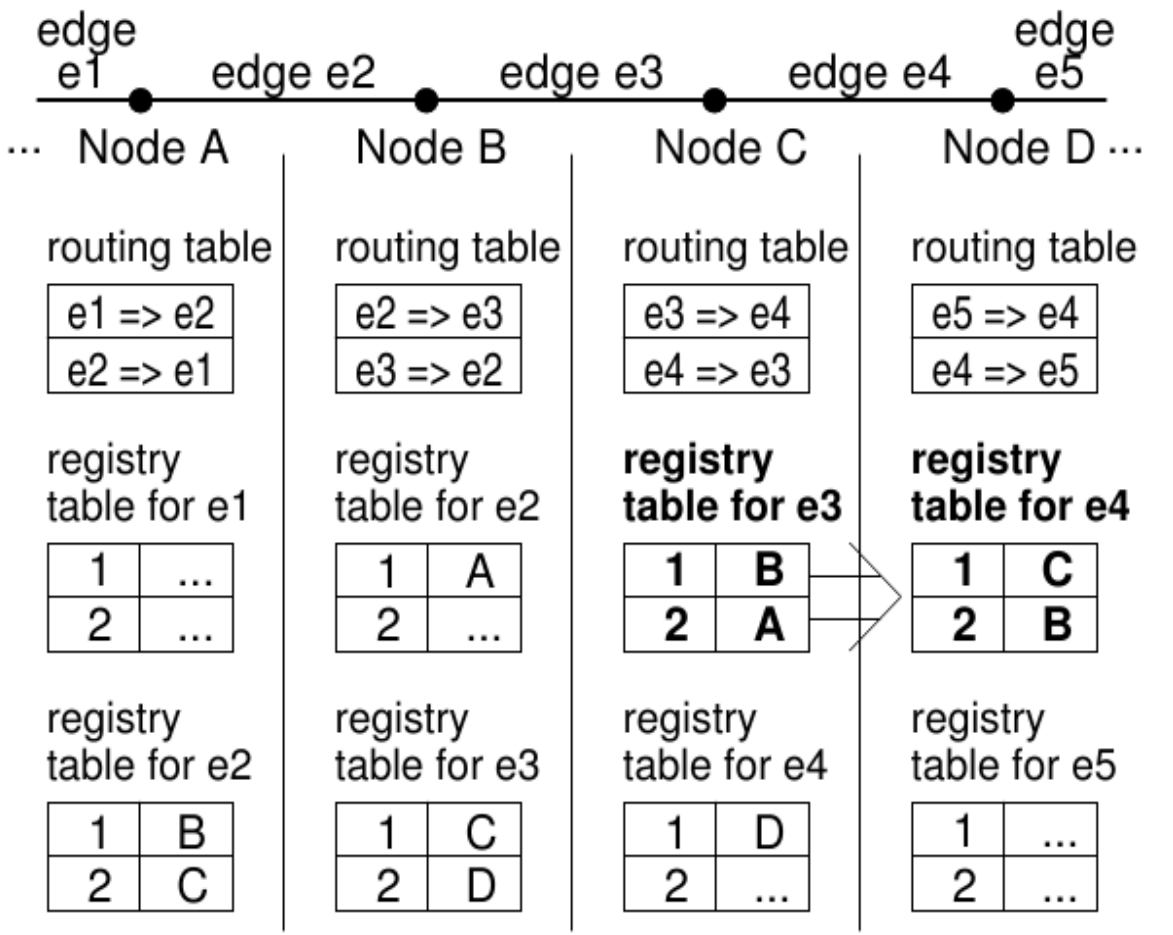


Figure 6: Maintaining the registry tables. In order to simplify this example, $w = 2$, each node has exactly two edges, and the routing tables are carefully chosen. The node names in the registry tables stand for the nodes' public keys.

SybilGuard - tablice świadków

- analogicznie do tablicy rejestracji
- węzeł zapisuje, kto jest na jego losowych ścieżkach
- zapisujemy klucze publiczne lub ich hasze
- oraz IP by przyśpieszyć weryfikację

SybilGuard - protokół weryfikacji

- S wysyła V tablicę świadków
- V szuka pierwszego przecięcia w X ścieżek S i V
- V pyta się X czy S jest w jego tablicach rejestrach
- X odpowiada V podpisując komunikaty swoim kluczem prywatnym
- połowa ścieżek ma się przeciąć

SybilGuard - zatruwanie tablic

- nawet jeśli złe węzły będą wysyłały oszukane tablice
- to jedna atakująca krawędź zepsuje
  - $w + (w-1) + ...1 \ = w^2/2$ tablic dobrych węzłów
- będzie $g\frac{w^2}{2}$ z $n * d * w$ popsutych tablic
- co daje mniej niż połowę

Twierdzenie 1

W każdej spójnej nie dwudzielnej sieci społecznej, losowa ścieżka o długości w startująca z uczciwego regionu przetnie jakąkolwiek z g krawędzi atakujących z prawdopodobieństwem mniejszym niż $g * w/n$.

W szczególności dla $w = \Theta(\sqrt{n}log(n))$ i $g = \Theta(\sqrt{n}log(n))$ prawdopodobieństwo jest $o(1)$

Twierdzenie 2
Niech $w = \sqrt{n}log(n)$.
Dla dowolnych X i Y w każdej spójnej nie dwudzielnej sieci społecznej, losowa ścieżka o długości w startująca z X przetnie się z jakąś losową ścieżką Y o długości w z prawdopodobieństwem większym niż $\frac{1}{2}$.

Jak dobrać w nie znając rozmiaru sieci?

- używając losowego błądzenia po sieci
- szacowań i rachunku prawdopodobieństwa

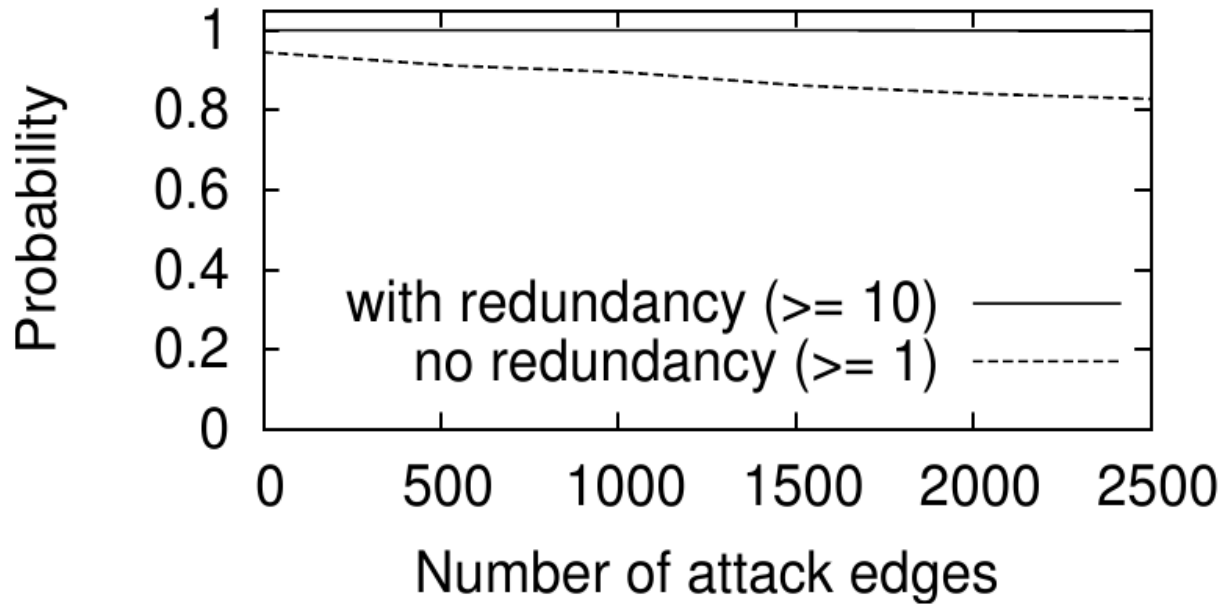


Figure 12: Probability of an honest node accepting another honest node (i.e., having at least a target number of intersections). The legends are the same as in Figure 9, and SybilGuard corresponds to "with redundancy $(>= 10)$".

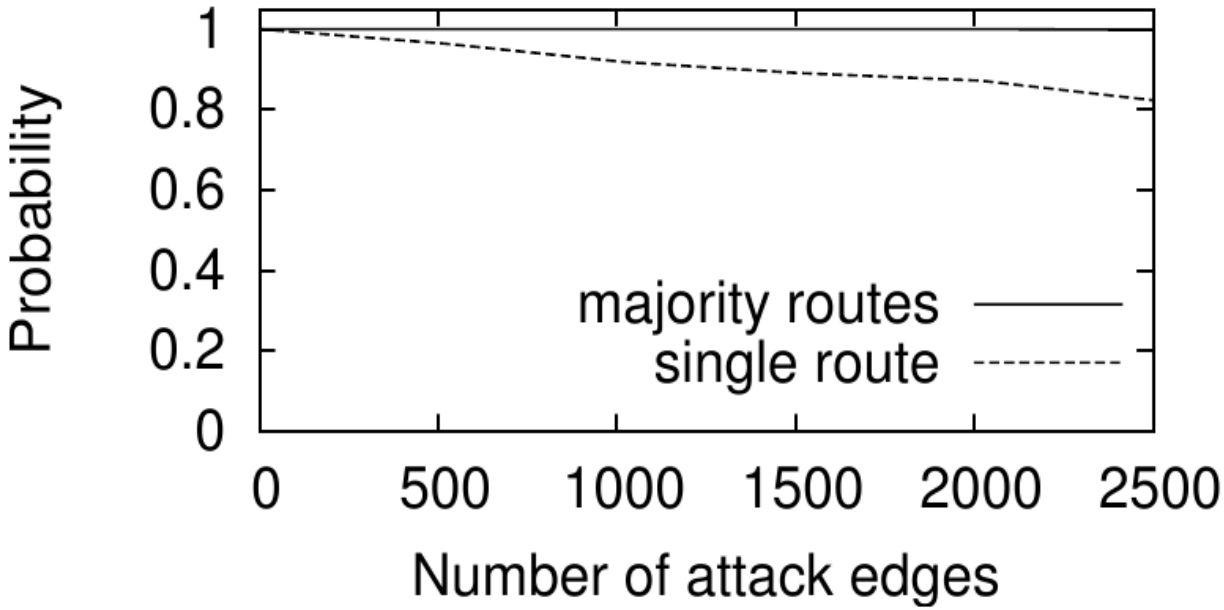


Figure 11: Probability of routes remaining entirely within the honest region.

Jak dobrać w nie znając rozmiaru sieci?

- używając losowego błądzenia po sieci
- szacowań i rachunku prawdopodobieństwa

SybilGuard - charakterystyka

- bardzo skuteczne rozwiązanie
- wymaga istnienia silnych przyjaźni
- wymaga wymiany kluczy symetrycznych offline

Eclipse attack - zasada działania

- jeżeli *uczciwe* węzły mają wystarczająco dużo *złych* sąsiadów
- to mogą zostać "przyćmione" przez *złe* węzły
- co może zepsuć niektóre protokoły bezpieczeństwa
- również może powstać w przypadku zatruwania tablic rutowania

Eclipse attack - przeciwdziałanie

- ograniczenie od kogo przyjmujemy informacje o rutowaniu
- nadmiarowe tablice rutowania
- głosowanie przy ustalaniu ID
- wymuszany *churn*
- audyt stopni węzłów
- komunikowanie się tylko z sieciowo-bliskimi węzłami

Ataki typu routing i storage

- rzeczywiste, szkodliwe ataki
- złośliwe węzły nie odpowiadają na zapytania
- wysyłają niewłaściwe dane
- nie rutują komunikatów
- rutują do złych węzłów (DDoS)

Ataki typu routing i storage - przeciwdziałanie

- dwie tablice rutowania (zoptymalizowana i weryfikująca)
- równoległe zapytania po różnych klasach abstrakcji sieci
- certyfikowanie, że dane prefiksy ID istnieją
- dużo replik i zapasowych ścieżek
- specjalnie konstruowane ścieżki, grupowanie węzłów

Teoria a praktyka

- najbardziej popularne DHT oparte są na Kademlii
  - Kademlia niespecjalnie przejmuje się bezpieczeństwem
  - dba o wydajność
  - pozwala węzłom na wybór ID
  - trzyma repliki ze względu na wydajność
  - wykonuje "szerokie" rutowanie
- reszta raczej nie przejmuje się bezpieczeństwem

Literatura

- A Survey of DHT Security Techniques, G. Urdaneta, G. Pierre and M. Van Steen
- Limiting Sybil Attacks in Structured Peer-to-Peer Networks
- SybilGuard: Defending Against Sybil Attacks via Social Networks
- Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications
- Defending the Sybil Attack in P2P Networks: Taxonomy, Challenges, and a Proposal for Self-Registration

Pytania?