平成20年度　　卒業研究論文

通信用公開鍵の配布機能を有した匿名通信路の設計と実装

指導教官
齋藤 彰一 准教授

名古屋工業大学　情報工学科
平成17年度入学　17115091番

田中 寛之

目 次

第1章

はじめに

近年，インターネットが普及するに伴い，電子メールやウェブサイト，掲示板システム (BBS) などインターネットを用いたコミュニケーション手段が多く使われるようになった．これに伴い，内部告発や医療相談などプライバシーの保護が重要な行為やサービスが，インターネットを用いて行われることも考えられる．

プライバシーの保護が必要な情報をやりとりする場合，実社会では匿名を前提とした仕組みがいくつもある．しかし，インターネットではIP アドレスと時間から個人の特定が可能であり，プライバシーの保護が十分とは言えない．インターネットでのプライバシーを保護するためには，通信における匿名性 [1] が必要である．

一般に通信における匿名性については次の 3 種類が挙げられる．

- 送信者匿名性
- 受信者匿名性
- 送信者と受信者のつながりの匿名性

上記の通信における匿名性を考慮した既存の匿名通信路にはTor(The onion router)[2]，Crowds[3] などがある．しかし既存の匿名通信路では，送信先による経路選択の柔軟性が充分ではない．また，鍵サーバが存在することによる問題点も抱えている．そこで本研究は，これらの問題を解決した匿名通信路を提案し，実装と評価を行った．実装はオーバレイ構築ツールキットである Overlay Weaver[4] を基盤としていくつかの機能を追加実装することで行った．

以下，2 章では提案手法の基盤となっている研究や比較対象となる研究など関連研究について述べ，3 章で提案方式について詳細に述べる．続く 4 章では提案手法の実装方法について述べる．5 章では提案手法の評価と他の手法との比較を行う．6 章で結論をまとめる．

第2章

関連研究

本章では提案方式での分散ハッシュテーブルに用いている Chord，多重暗号化により匿名通信を行う Onion Routing(Tor)[5],[6] と，既存の匿名通信方式である Crowds, ノード管理に DHT を用いた匿名通信方式の 4 つについて述べる．

2.1 Chord

提案方式ではノードの管理に Chord[7] を用いている．Chord は分散ハッシュテーブル (DHT) を実現するアルゴリズムの 1 つであり，環状のハッシュ空間を持つ．Chord に参加するノードには，環状のハッシュ空間に含まれる ID が割り当てられる．同時に各ノードには管理するべきデータの範囲が与えられ，この範囲に含まれるデータを所有することになる．管理する範囲はハッシュ空間の ID で与えられる．データの ID はデータをハッシュ関数にかけることで求められる．

Chord に参加する各ノードは Fingertable（ルーティングテーブル）を持っており，いくつかのノードの IP アドレスを知っている．目的のデータを管理するノードを高速に検索するため，Fingertable には，検索を高速に行うことができるようにスキップリストの方式を取り入れ，ID が 2^n $(0 \leq n <$ [ハッシュ空間の大きさ]) 離れたノードがいくつか登録されている。例えば，ハッシュ空間の大きさを 2^m とするとノードに割り当てられる ID は $0 \leq Node\_ID < 2^m$ となり，Fingertable には以下で示す $next(Z)$ のノードの IP アドレスが登録される．ここで next(Z) は ID が Z となるデータを管理するノードの ID を表す．

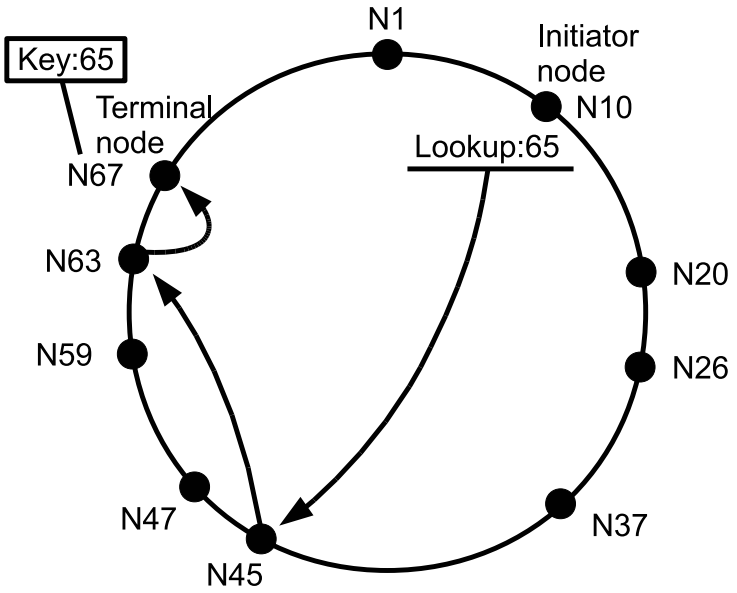

図 2.1: Chord による探索

$$Z = [\text{自ノードの } ID] + 2^i mod 2^m \ (0 \leq i < 2^m)$$

ある ID のデータを管理するノードを検索するときには，検索空間を半分以下に絞る動作を繰り返すことで高速にノードを検索することができる．図 2.1 に Chord の動作を示す．図 2.1 ではノード N10 が 65 というキーの検索を行っている．ノード N10 はまず自分の Fingertable を見て，キーである 65 以下で 65 に一番近いノードに対して問い合わせを行う．図 2.1 ではノード N45 である．N45 以降，同じように問い合わせを繰り返し最終的に Key:65 を管理するノード N67 にたどり着く．このように問い合わせを行うことにより，$O(logn)$ 回の問い合わせで目的のノードを見つけることが出来る．

2.2 Onion Routing(Tor)

Onion Routing は送信者と受信者，複数の中継ノードから成る．送信者は中継ノードとの共通鍵でメッセージを多重暗号化して送信する．各中継ノードはメッセージの復号を行い，次の宛先を得て，次のノードへとメッセージを中継する．この動作を受

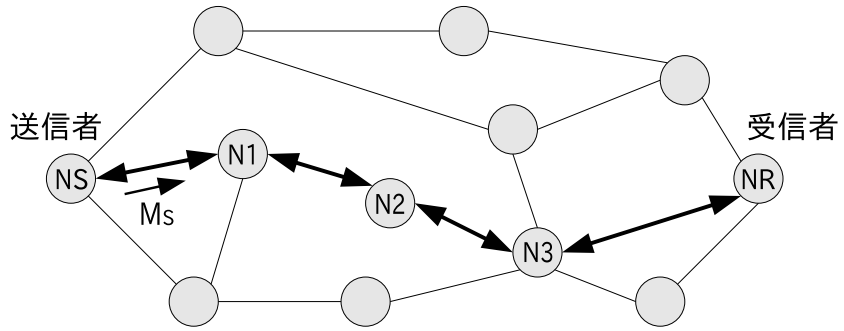

図 2.2: オニオンルーティングの動作

信者にたどり着くまで繰り返す.

このようにメッセージを多重暗号化して送ることで中継ノードは前後のノードしか知ることができず，1 つ前のノードが送信者なのか中継ノードなのか，1 つ後ろのノードが受信者なのか中継ノードなのか判断できない．図 2.2 にオニオンルーティングの動作を示す．送信者 NS はフラッディングにより得た経路のうち一つを用いてメッセージ M_S を送信する．図 2.2 では送信者 NS から始まり，中継ノード N1,N2,N3 を経由して受信者 NR へと至る経路である.

多重暗号化されたメッセージ M_S は以下に示す再帰的計算で得られる．ここで V は通信内容であり，K_1,K_2,K_3,K_{N_R} はそれぞれ N1,N2,N3,NR との共通鍵である．また，$A \parallel B$ は A と B との結合を表し，IP_n はノード n の IP アドレスを示す.

$$M_S = IP_1 \parallel K_1(IP_2 \parallel K_2(IP_3 \parallel K_3(IP_{NR} \parallel K_{NR}(V))))$$

各中継ノードは自分が持っている共通鍵でこのメッセージを復号し，中継することで段々と復号が進んで行く．最終的に受信者が復号を行うことで通信内容 V が得られる.

オニオンルーティングの様な多重暗号化を行う方式では、中継ノードのうち 1 つでも信頼できるものが存在すれば匿名性が保たれる．しかし，中継ノードのうち 1 つでも故障すると通信を継続することが出来ない．これは，各ノードは前後のノードしか知らず，通信路の修復を行うことが出来ないためである．また，経路の生成にフラッディングを用いているためスケーラビリティが高いとは言えず，経路選択の柔軟性も

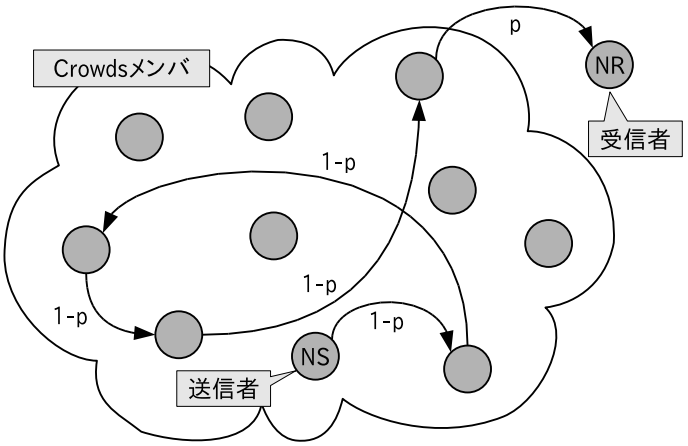

図 2.3: Crowds の動作

低い.

2.3 Crowds

Crowds は，協調し合う複数のノードのグループで構成される分散システムであり，参加するノードがそれぞれプロキシの役割をする．Crowds に参加するノードがメッセージを送信する際には，参加ノード間でランダムにメッセージを転送した後に受信者へメッセージを送信する．Crowds の動作概要を図 2.3 に示す.

図 2.3 では，メッセージは確率 p で受信者へ送信され，確率 $1-p$ で Crowds に参加するノードへ送信される．図 2.3 では送信者 NS から送信されたメッセージは 4 つのノードを中継した後，受信者 NR へ送信されている．このようにすることで最初にメッセージを送信したノードの特定を困難にする。返信時は，各中継ノードがメッセージを送ってきたノードに対して中継を行うことで送信者までメッセージを届ける.

Crowds では経路の選択に確率を用いており，経路選択の柔軟性は高い．しかし，メッセージの到達性を保証するため，どの中継ノードにも受信者が分かるようにメッセージに受信者の情報を記さねばならない．このため受信者の匿名性を保つことが出来ない．また，匿名通信路を構築するノードが離脱した際には再び通信路を構築しなおす

必要があり，離脱耐性が高くない．Crowds メンバを管理するためにサーバが必要であり，スケーラビリティも高いとは言えない．

2.4 ノード管理に DHT を用いた匿名通信方式

本研究室で研究されている既存方式について説明する．

2.4.1 全体

参考文献 [8] で提案されている方式はオニオンルーティングの多重暗号化と分散ハッシュテーブルである Chord を組み合わせることで，柔軟な経路生成が可能かつ可用性とスケーラビリティの高い匿名通信路を提案している．既存方式は，ノード管理層と匿名通信路層で構成されている．そのため，既存方式ではノード管理層と匿名通信路層の二つのオーバレイネットワークを持つことになる．

ノード管理層では Chord のアルゴリズムを用いてノードの参加・離脱時の処理や経路制御を行う．匿名通信路層では，経路の設定やメッセージの多重暗号化など匿名通信に必要な処理の全てを行う。

既存方式でのメッセージを中継する経路については送信者が指定する．経路の指定は受信エリアと呼ばれるノードの集合の始点と終点を設定することで行う．送信者は経路のどこかに受信者が含まれるように指定する．メッセージは全ての受信エリアを通り，途中のどこかで受信者に受信される．

また，既存方式には公開鍵サーバが存在し，Chord に参加するノードの公開鍵を管理している．送信者はヘッダ部とデータ部を暗号化するための公開鍵を公開鍵サーバから取得し，メッセージを生成する．

2.4.2 ノード管理層

ノード管理層は分散ハッシュテーブルである Chord を用いており，Chord に参加するノードの ID 管理，参加・離脱時の処理，経路制御（ルーティング）を行う．これらの処理はノードのアルゴリズムに従って行われる．ノードの参加・離脱時には Chord

に参加するノードが能動的に経路情報を再構成する．経路制御は，Fingertable を用いてスキップリストの要領で目的の ID を管理するノードを検索し，IP アドレスを得る．

ノード管理層では分散ハッシュテーブルを維持するためのメッセージがやりとりされるがこのメッセージには匿名性は必要ない．一方，匿名通信路層でやりとりされるメッセージには匿名性が必要となる．二つの層に分離することによって匿名性が必要ない．暗号化などの匿名性を保つために一手間かけるのは匿名通信路層のみでよくなるため，ノード管理層での処理を簡単に行うことが出来る．

2.4.3 匿名通信路層

匿名通信路層では経路を決定し，メッセージを暗号化し実際に通信を行う層である．ノード管理層のおかげで，この層では中継に使うノードが通信可能な状態であるか，や中継ノードの IP アドレスをわざわざ調べて指定する必要がない．経路の決定は送信者が中継させるノードを Chord の ID で指定することで行う．

既存方式でのメッセージはヘッダ部とボディ部から構成され，ヘッダ部には経路情報が，ボディ部には通信内容がそれぞれ多重暗号化されて入っている．匿名通信路層はノード管理層の情報を用いて経路を決定し，実際にヘッダ部・ボディ部を多重暗号化しメッセージの生成して通信を行う．

既存方式で用いられるメッセージには，構築メッセージ，データメッセージ，匿名通信路を制御するためのメッセージの 3 種類が存在する．構築メッセージは匿名通信の開始時に 1 度だけ送信され，各中継ノードと受信者との共通鍵と，返信用の構築メッセージが含まれる．共通鍵は各ノード，匿名通信路ごとに異なり，送信者が生成する．構築メッセージは公開鍵サーバから各受信エリアの終点ノードの公開鍵を取得して Onion Routing と同様に多重暗号化する．データメッセージは受信者とデータをやりとりするメッセージである．データメッセージは匿名通信路の構築メッセージに含まれて送られる共通鍵を用いて多重暗号化を行う．

受信者の指定方法は，受信者を匿名通信路の途中に配置し，返信用の構築メッセージを復号できたノードが受信者であるという間接的な方式を採っている．受信者以降もメッセージの中継は続き，この通信はダミー通信となる．

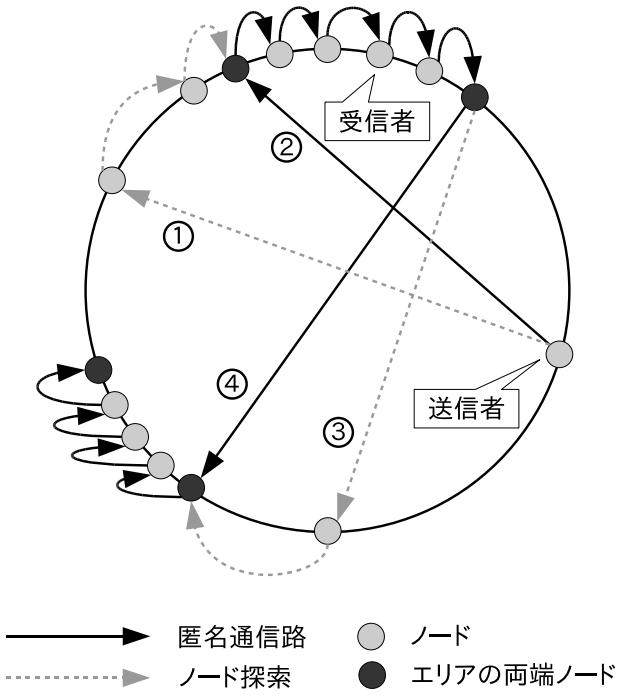

図 2.4: 既存方式の通信の流れ

2.4.4 通信手順の概要

実際の通信の流れを図 2.4 に示す．図 2.4 で受信エリアの両端ノードで挟まれている区間が一つの受信エリアであり，ここでは 2 つの受信エリアが存在する．

この方式では送信者が経路，すなわち受信エリアの始点と終点を決定する．その方法は受信エリアの終端ノードの公開鍵でヘッダ部を暗号化するという間接的なものである．送信者は受信エリアの終点ノードの公開鍵でメッセージを多重暗号化して送り，全てのノードはメッセージを中継する際に，自分の公開鍵でメッセージを復号しようと試みる．このとき，メッセージを復号出来たノードが受信エリアの終点であり，次の受信エリアが存在するならば，次の受信エリアの始点ノードの ID が得られる．

図 2.4 でメッセージ送信の流れを追う．(1)1 つ目の受信エリアの始点ノードにメッセージを送るためそのノードの IP アドレスを Chord を用いて検索する．(2)IP アドレスが得られたら次の受信エリアの始点ノードへメッセージを送信する．始点ノードが

メッセージを受け取ったら，ヘッダ部の復号化を試みる．図 2.4 の場合は復号化に失敗し，直後のノード，すなわち Successor へメッセージが中継される．以後，ヘッダ部の復号に成功して次の受信エリアの始点ノードが分かるまでは Successor へのメッセージ送信を繰り返す．ヘッダ部の復号に成功した場合はそのノードが受信エリアの終点ノードであるということであり，次の受信エリアの始点ノードの ID を取得することができる．そして，(3) 取得したノードの ID を Chord を用いて検索し，(4) 先程と同様にメッセージを送信する．図の場合では 1 つ目のエリアの途中に受信者が存在する．受信者のみがメッセージのデータ部を復号することが可能であり，データ部を復号出来たノードは自分が受信者だと分かる．この様にしてメッセージが受信者に伝わる．

2.4.5 問題点

既存方式の匿名通信路では，システムに参加する全ノードの公開鍵を管理するために公開鍵サーバが必要である．攻撃者に公開鍵サーバと各ノードとの通信を監視されることで，送信者や受信者に関する情報を攻撃者に与えてしまう恐れがある。また、システムに参加する全ノードの公開鍵を登録し、さらに公開鍵を一定時間ごとに更新するため，公開鍵の管理コストが高くスケーラビリティが低い。

第3章

提案方式

本章では提案方式である鍵の配布機能を有した匿名通信路について述べる．

3.1 全体

2.4節で述べた既存方式の問題点を解決する，鍵の配布機能を有した匿名通信路を提案する．提案方式では公開鍵サーバが存在しないため，既存方式ではあった問題点が存在しない．本提案方式でも既存方式と同様にノード管理層と匿名通信路の二層構造となっている．

既存方式では全てのノードが公開鍵を持っており，その公開鍵は公開鍵サーバによって管理されていた．本提案方式では公開鍵を持つノードと持たないノードが存在し，公開鍵の総数を減らしている．また匿名通信路自体に公開鍵の配布機能を持たせているため公開鍵サーバが必要ない．そして，公開鍵の配布機能により提案方式に参加する全ノードは全てのスーパーノードの公開鍵を持つ．以降公開鍵を持つノードを**スーパーノード**，公開鍵を持たないノードを通常ノードと呼ぶ．

また．スーパーノードの存在を利用して，既存方式では出来なかったメッセージ送信の並列化を行う．これは，既存方式よりも速く後続ノードにメッセージを伝えるためである．

3.2 ノード管理層

ノード管理は既存方式と同様に，分散ハッシューテーブルである Chord により行っている．ノード管理層ではノードの ID 管理，参加と離脱時の処理，経路制御を行っている．この章では既存方式とは処理が異なるノードの参加時と離脱時の処理とエリア管理の処理について説明する．

3.2.1 ノード参加時

ノード参加時には Chord のアルゴリズムに加えて以下の処理を行う．

- スーパーノードと公開鍵リストの取得
- スーパーノードへの参加報告

ノードは提案方式の匿名通信路に参加するときに Chord のアルゴリズムに基づいて ID が割り当てられ，担当するべきデータの範囲が決定される．参加した時にはスーパーノードも公開鍵も分からないため，最初にアクセスするノードまたは Predecessor か Successor からスーパーノードと公開鍵のリストを取得する．取得したスーパーノードのリストと新規ノードに割り当てられた ID を見ることで新規ノードを担当するスーパーノードの ID がわかる．そして新規ノードがいるエリアを担当するスーパーノードへ新規ノードが参加したことを報告する．

3.2.2 ノード離脱時

ノード離脱時には Chord のアルゴリズムに加えてスーパーノードへの離脱報告を行う．特にスーパーノードが離脱する際には，更に以下の処理を行う．

- スーパーノード引き継ぎ処理
- スーパーノードと公開鍵リストの更新

離脱するノードがスーパーノードの場合，代わりのスーパーノードを立てるか隣接エリアと統合する．また，全スーパーノードにスーパーノードが離脱したことを伝え，スーパーノードと公開鍵のリストを更新する．これらの処理を行ったのちに離脱する．

ノードの故障等の理由で不意に離脱する場合は，Chord のアルゴリズムによりスーパーノードの Successor がスーパーノードの離脱を検知する．そして Successor がスーパーノードとなる．スーパーノードが交代したとき，前スーパーノードが持っていた共通鍵テーブルがないと既に確立した匿名通信路の通信を継続することができない．このため，スーパーノードは予め Predecessor と Successor に XOR 分割した共通鍵テーブルを渡しておく．処理を引き継ぐ Successor は前スーパーノードの Predecessor から共通鍵テーブルの片割れを取得して共通鍵テーブルを復元する [9].

3.2.3 管理エリア

スーパーノードはノードの管理責任を負い，管理を担当するエリアが割り当てられる．以降このエリアを**管理エリア**と呼ぶ．スーパーノードはエリアを管理するにあたり以下の処理を行う．

- 担当する管理エリアのノード数管理
- 担当する管理エリアの分割と統合処理

スーパーノードは，担当する管理エリアのノード数を定期的に数える．新規参加したノードからの参加報告，既に参加しているノードからの離脱報告によってもノード数を更新する．

また，スーパーノードは担当する管理エリアのノード数がある程度まで増加，減少した場合に管理エリアの分割・統合を行う．管理エリアを統合する場合は，自分の持っている鍵テーブルを統合先のスーパーノードに渡し，スーパーノードと公開鍵のリストを更新する．そして自身は通常ノードとなる．管理エリアを分割する場合には，管理エリアの中央付近のノードを新たに作成する管理エリアのスーパーノードに任命し，新たな管理エリアのスーパーノードは公開鍵を生成する．そして公開鍵のリストを更新する．

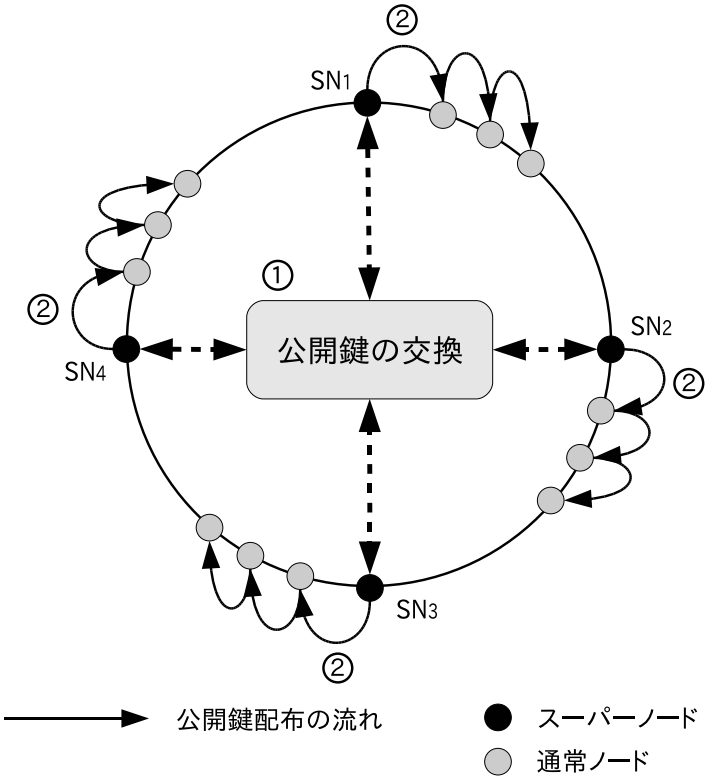

図 3.1: 公開鍵の配布動作

3.2.4 公開鍵の配布

提案方式の匿名通信路は公開鍵の配布機能を有している．公開鍵配布の流れを図 3.1 に示す．公開鍵の配布は，(1) 各スーパーノード同士で公開鍵を交換して全ての公開鍵を集め，(2) 各スーパーノードが管理エリアへの公開鍵を配布する，という順番で行う．これにより全てのノードに全てのスーパーノードの公開鍵が行き渡る．

3.3 匿名通信路層

匿名通信路層では，匿名通信路の構築，メッセージの生成と暗号化，メッセージの送受信を行う．ここでは匿名通信路が行う処理について述べる．

3.3.1 通信方式の概要

提案方式は受信者を匿名通信路の途中に配置し，Onion Routing と同様に多重暗号化したメッセージを用いる．提案方式でもメッセージには構築メッセージ，データメッセージ，制御メッセージの 3 種類が存在する。

送信者は最初に匿名通信路を構築するため，構築メッセージを送信する．このとき各中継ノードに送信者との共通鍵を配布し，以降の通信では共通鍵を用いてヘッダの多重暗号化を行う．匿名通信路を構築したのち，伝えたいデータを含んだメッセージを送信する．

経路は，中継させるスーパーノードとスーパーノードがヘッダを復号したときに得られる次の宛先ノードを送信者が指定することで決定する．送信者はまず Successor にメッセージを送信し，自分以降の通常ノードに，次のスーパーノードにたどり着くまで Successor にメッセージを中継する処理を繰り替えす．スーパーノードまでたどり着くとヘッダ部の復号を試みる．ヘッダ部が復号出来なかった場合はまた次のスーパーノードまで Successor へ中継する処理を繰り返す．ヘッダ部が復号できた際には次にメッセージを送るノードの ID が得られるため，IP アドレスをノード管理層で検索して送信する．同時メッセージ送信を並列化し，そのスーパーノードが担当する管理エリアに存在する通常ノードにもメッセージを行き渡らせる．このときスーパーノードからの送信者やメッセージを受け取るノードを始点とし，次にヘッダ部の復号ができたスーパーノードを終点とする区間と管理エリアを合わせた領域を**受信エリア**と呼ぶ.

経路の途中に含まれるスーパーノード，通常ノードのどれかが受信者であり，データ部の復号に成功したノードが受信者であると分かる．返信時はデータ部から取り出した返信用ヘッダを使い，同様の処理を行う.

3.3.2 経路の決定

経路の指定は，中継させるスーパーノードと，スーパーノードがヘッダを復号したときに得られる次の宛先ノードを，ID で送信者が指定することで決定する．中継に用いるノードは ID で指定すれば良いため，実際の IP アドレスを調べるコストが必要な

い．Chord のアルゴリズムを用いるため，その ID を担当するノードを指定することになり，その ID のノードが実際に存在するかどうかも気にしなくてよい．

復路も送信者が予め指定しておき，受信者だけが復号できるデータ部に含めておく．受信者はこれを取り出して返信用のヘッダとする．送信者は経路の途中に受信者が存在する様に経路を指定する．

3.3.3 受信エリア

受信エリアとは，メッセージを中継する際のスーパーノードからの飛び先ノードと次のスーパーノードに挟まれる区間と管理エリアを合わせた，Chord の ID が連続する領域である．受信エリアは以下の性質を持つ．

- 受信エリアに属するノードはすべて当該匿名通信路の中継ノードである
- 受信エリア内では，ノード検索による経路制御を行わず，隣接する Successor にメッセージを中継する
- 受信エリアの終点は，メッセージのヘッダを復号できたスーパーノードである
- メッセージのヘッダには次の受信エリアの始点ノードの ID を記述する
- 受信エリアは 1 つの匿名通信路に複数設定する事が出来る
- 受信エリアは 1 つ以上のノードで構成される

既存方式と異なるのは受信エリアの終点が必ずスーパーノードとなる点である．このため既存方式より経路選択の柔軟性は低くなっているが，後述するメッセージ送信の並列化が可能であり，通信速度は向上している．

復号を行うノードを受信エリアの終点となるスーパーノードのみに絞り，匿名性を保ったままヘッダの暗号化・復号回数を減らすことが受信エリアを設定する目的である．また，受信エリア内では Successor にメッセージを中継するだけなので次の宛先を検索するコストがかからず，経路制御の回数を減らす事が出来る．このため匿名通信路を用いた通信にかかる時間を短縮する事が出来る．

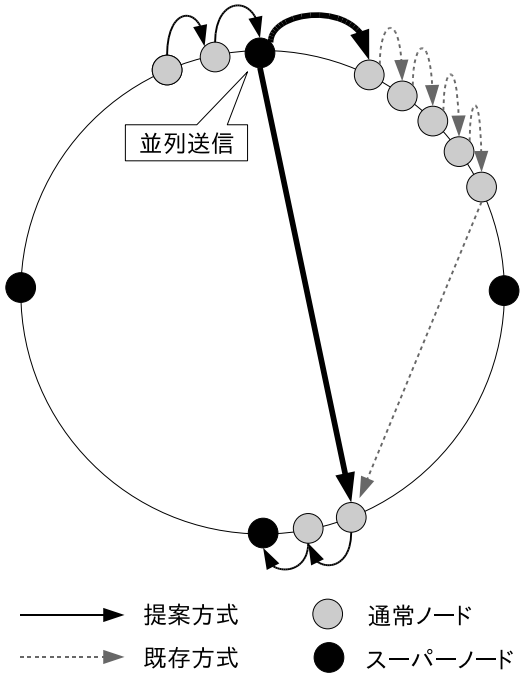

図 3.2: メッセージ送信の並列化

3.3.4 メッセージ送信の並列化

提案方式の匿名通信路ではスーパーノードからのメッセージ送信を並列化することで後続ノードに速くメッセージを中継する．図3.2に示すように受信エリアのうち，提案方式では太い実線で示すようにスーパーノードにおいてヘッダ部の復号に成功した時点で次の受信エリアの始点へのメッセージ送信と，残りの管理エリアのノードへの中継処理を並列に行う．既存方式では破線の様にエリアの全てのノードを経由してから次のエリアへメッセージが届くため，中継ノード数が同じ場合，既存方式よりも早い時点で次の受信エリアにメッセージを伝えることが可能である．

3.4 動作概要

ここでは提案方式について実際に通信を行う手順を説明する．全体の流れはメッセージの生成，通信路の構築，データメッセージの送受信という順番で通信を行う．

表 3.1: メッセージの構成要素

記号	説明
MB	構築メッセージ
HS	往路のヘッダ部
HR	復路のヘッダ部
BS	往路のボディ部
NS	送信者
NR	受信者
NN_i $(i = 1, 2, ...)$	ID が i の通常ノード
SN_i $(i = 1, 2, ...)$	ID が i のスーパーノード
PK_i	スーパーノード SN_i の公開鍵
$PK_i()$	公開鍵 PK_i で暗号化されたデータ
SK_R	受信者との共通鍵
SK_i	スーパーノード SN_i との共通鍵
SK_i	共通鍵 SK_i で暗号化されたデータ
$nextID_i$ $(i = 1, 2, ...)$	i 番目の飛び先ノード ID
$A \parallel B$	A と B の結合を表す
END	終端を示すデータ
D	通信内容

3.4.1 メッセージ生成

メッセージの生成は既存方式と同様に行う．ただし，公開鍵を公開鍵サーバから取ってくる処理はない．メッセージの構成要素を表 3.1 に示す．例えば構築メッセージ MB はヘッダ部 HB とボディ部 BS から構成され $MB = HB \parallel BS$ となる．ヘッダ部には匿名通信路の経路情報が含まれ，ボディ部には返信用ヘッダ HR と通信内容 D が含まれる．

ヘッダ部の生成

送信者 NS は公開鍵の配布機能により配られている公開鍵を用いてヘッダ部の多重暗号化を行う．多重暗号化するヘッダの内容は，中継させたいスーパーノードとそのスーパノードがヘッダを復号した時に得られる宛先ノード ID と以降の通信で用いる共

通鍵である．生成されるヘッダ部は以下の様になる．

$$MB = PK_1(nextID_1 \parallel SK_1 \parallel PK_2(nextID_2 \parallel SK_2 \parallel PK_3(END \parallel SK_3 \parallel END)))$$

生成手順はまず最初に最後の受信エリアに存在するスーパーノードとなる SN_3 の公開鍵 PK_3 で暗号化を行う．このとき，最後の受信エリアであるため次の飛び先ノードと残りのヘッダ部が入る位置には END を設定する．次の飛び先である $nextID_2$ と共通鍵 SK_2，PK_3 で暗号化された経路情報を結合し，PK_2 で暗号化する．同様に飛び先 $nextID_1$ と共通鍵 SK_1 を経由させたいスーパーノード SN_1 の公開鍵 PK_1 で暗号化する．この様な手順でヘッダ部を生成する．

データメッセージのヘッダ部は，上記の構築メッセージのヘッダ部から共通鍵を取り除いたものになる．

ボディ部の生成

メッセージのボディ部 BS は返信用ヘッダ HR と通信内容 D からなり，以下の様に表す事ができる．

$$BS = SK_R(HR \parallel D)$$

これは受信者 NR とのボディ部暗号化用の共通鍵 SK_R で返信用ヘッダ部 HR と通信内容 D を暗号化した物である．返信用のヘッダ部は送信者 NS が予め生成する．受信者は返信用ヘッダ部 HR を取り出し，返信する通信内用と合わせて返信メッセージを生成する．

通信路の構築は，送信者が構築メッセージを送り，中継するスーパーノードが共通鍵を保存することで行われる．

3.4.2 メッセージの送信

メッセージ送信時の動作を図3.3に示す．この図では送信者 NS が SN_1 を経由し，SN_3 が終点となる匿名通信路を設定している．送信者 NS は通常ノードであるため Successor

に中継し，以後の通常ノードはスーパーノード SN_1 にたどり着くまで Successor への中継を繰り返す．送信者がスーパーノードの場合は直接次のエリアへ送信する．

スーパーノード SN_1 はメッセージをメッセージを受け取るとヘッダ部の復号を試みる．図 3.3 の場合では復号に成功し，次の受信エリアの始点ノードを得る．そして，次の受信エリアの始点ノードの IP アドレスを Chord を用いて検索する．スーパーノード SN_1 からは自分の管理エリアと次の受信エリアへのメッセージ送信が並列に行われる．管理エリアへの送信では，Successor への中継を繰り返すことにより管理エリア全てのノードにメッセージが行き渡る．次のエリアへ送信では，受信エリアの始点ノードが通常ノードのためスーパーノード SN_3 にたどり着くまで Successor への中継を繰り返す．

メッセージを受け取った SN_3 はヘッダの復号を試み，成功する．そして，ヘッダ部の次エリアの始点ノード指定に END が書かれていることから自分のいる受信エリアが最後であると判断し，SN_3 の管理エリアにのみメッセージを送信する．

図 3.3 では SN_1 のいる受信エリアに受信者が含まれているが，この様に送信者は匿名通信路のどこかに受信者が含まれるように経路を指定する．受信者 NR のみがボディ部の復号化に成功し，自分が受信者であると分かる．返信時も同様に，ボディ部に含まれる返信用ヘッダを用いて通信を行う．

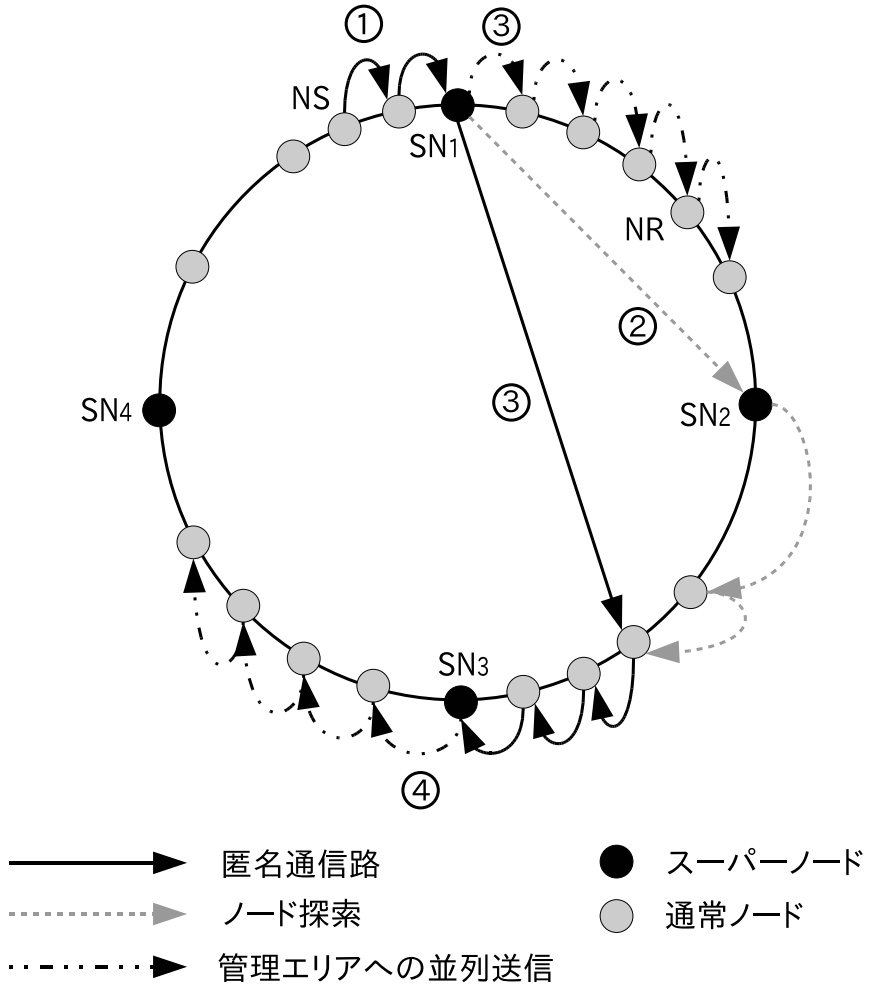

図 3.3: メッセージ送信の動作

第4章

提案方式の実装

提案方式の匿名通信路の実装を行った。本章では提案方式の実装概要と，実装の基盤としたオーバレイ構築ツールキットである Overlay Weaver について述べる．

4.1 実装概要

提案手法の実装はオーバレイ構築ツールキットである Overlay Weaver を基盤として，以下の機能を追加実装することにより行った．なお，3.2.1，3.2.2，3.2.3 節で述べた，ノードの参加と離脱時の処理を行う機能は現段階では未実装である．このためノードの動的な参加と離脱はないという条件で実験を行っている．

- スーパーノードを決定する機能
- スーパーノードが自分の担当する管理エリアに鍵を配布する機能
- 送信者が設定した経路情報を元にヘッダ部を生成し，多重暗号化する機能
- メッセージを送受信する機能
- 通信経路ごとにセッション ID を割り当て，2 回目以降の通信に共通鍵を使う機能

スーパーノードを決定する機能については，現在はスーパーノードの台数と ID を固定しているため，参加時に与える ID によって判断している．スーパーノードになる ID を予め指定している．

スーパーノードが自分の担当する管理エリアに公開鍵を配布する機能については，公開鍵配布メッセージであるというタグをメッセージに与え，管理エリアの通常ノードに，Successor へ中継させる処理を繰り返すことで行っている．そして次のスーパーノードにたどり着く．ノードがメッセージを受信したとき，メッセージにつけられたタグによってメッセージに対する処理を決定する．スーパーノードはメッセージのタグを見て，公開鍵配布メッセージであると判別する．そして公開鍵配布メッセージの中継を終了することによって一つの管理エリアへの公開鍵配布が完了する．

ヘッダ部の多重暗号化については，構築メッセージは RSA を，データメッセージについては AES を用いている．送信者は自身に配られた公開鍵リストを用いて構築メッセージを暗号化する．共通鍵はランダムに送信者が生成する．

ノードが Chord 参加時に生成するノード固有のランダムな値と経路情報を元にハッシュをとった値をセッション ID とする．また，スーパーノードがヘッダの復号を行うごとに，今までのセッション ID と中継するスーパーノード固有のランダムな値のハッシュをとり，新たなセッション ID として書き換える．これは，セッション ID を見られても匿名通信路の全容が分からない様にするためである．

4.2 Overlay Weaver

Overlay Weaver はオーバレイ構築ツールキットの１つであり，Chord など複数のルーティングアルゴリズムの実装を提供している．P2P などのインターネット上で分散して動作するようなアルゴリズムを設計し評価する場合には，可能であれば数百万ノードといった規模での動作を想定した実験が欠かせない．しかし，実際に大規模な実験環境を用意することは困難であるため，通常はシミュレーションでアルゴリズムの評価を行う．Overlay Weaver では Dabek らのルーティング処理抽象化の考えに基づき，あらゆる structured オーバレイに共通する処理を行うルーティング層と，DHT サービスやマルチキャストサービスを分離している (図 4.1 参照)．これにより，Overlay Weaver でシミュレーションや実権を行う場合，作成したプログラムに変更を加えることなく様々なルーティングアルゴリズムを差し替えて利用することが出来る．

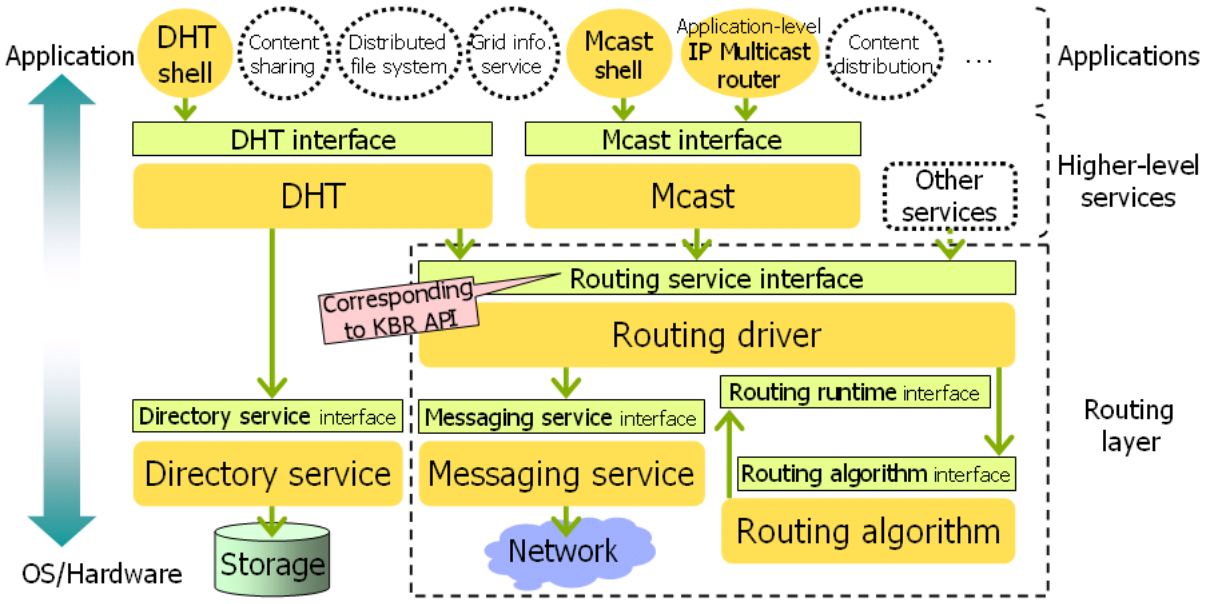

図 4.1: Overlay Weaver のコンポーネント群

第5章

評価と比較

本章では提案方式の匿名通信路の匿名性の検証と通信時間の評価を行う．また，2.4節で示したノード管理にDHTを用いた匿名通信方式との通信時間の比較も行う．

5.1 匿名性の検証

本節では提案方式の匿名性について検証する．まず攻撃者が単独の場合に，中継ノードのメッセージ送受信を監視することで，送信者と受信者の情報を得られるかどうかを検証する．次に複数の攻撃者が結託した場合について検証を行う．

5.1.1 通信解析攻撃

通信解析攻撃とはノードの通信を監視し，通信内容を含むメッセージ送受信を全てログに取り，送受信の関係性を解析することで匿名性を無くす攻撃である．様々な場所に存在するノードから構成されるオーバレイネットワークではネットワーク全体を監視することは困難である．そのため，ここでは特定のノードを監視した場合に得られる情報について考える．

中継ノードの送受信パターンの解析

ここでは中継ノードに対して通信解析攻撃を行った時に得られる情報を述べる．メッセージは暗号化されているため，メッセージ自体から送信者と受信者の情報を得るこ

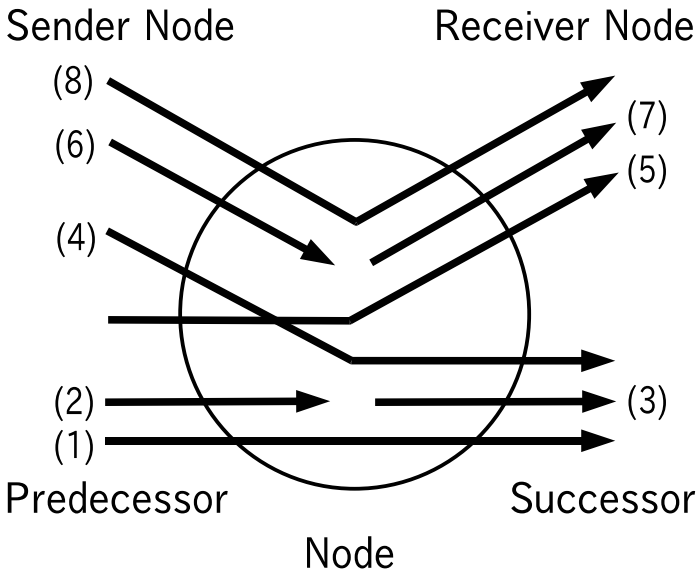

図 5.1: 中継ノードの通過パターン

とは出来ない．そのため多重暗号化されたメッセージがどこからどこへ中継されるかを監視することで送信者と受信者の情報が得られるかどうかを確認する．

送信パターンを分類したものを図 5.1 に示す．Predecessor は Chord における Predecessor からの受信を表し，Successor も同様に Chord の Successor からの受信を表す．Sender Node は Prdecessor 以外からの受信を表し，Receiver Node は Successor 以外への送信を表す．

通信解析攻撃では，同一のパケットの流れを追うことで送信者や受信者に関する情報を得ようとするが，メッセージが復号された場合は今までとパケットが変わるため復号前後のパケットが変化し，攻撃者は流れを追うことが出来ない．送信と受信をの両方を確認することができるのは，復号されずに中継されるメッセージのみである．

(1)Predecessor から Successor へ

Predecessor から受信し，Successor へ送信するパターンである．このパターンは受信エリア内でメッセージを中継する際に現れる．このパターンが現れたノードは受信

エリア内にいるということが分かるが，送信者と受信者の情報を得ることは出来ない．

(2)Predecessor から受信のみ

Predecessor からの受信は確認できたが，同じメッセージの送信が確認できないパターンである．このパターンはエリアの終点となるスーパーノードがメッセージを復号した場合に現れる．このパターンが現れたノードは，ある受信エリアの終点にいるということが分かるが，メッセージが次のエリアへ送信されているかどうかは分からない．また，送信者と受信者の情報を得ることも出来ない．

(3)Successor へ送信のみ

どこからの受信も確認されず，Successor へ送信のみ確認できたパターンである．このパターンは監視しているノードが送信者の場合に現れる．また，スーパーノードにおいて，次の受信エリアの始点がスーパーノードの Predecessor の場合にも現れるが，通常ノードで現れた場合は監視しているノードが送信者だと分かる．

しかし，この問題は Successor との間で DH を用いて共通鍵を交換し暗号化を行うか，より処理の簡単な XOR 暗号を用いてメッセージを変化させることで解決することが出来る．

(4)Predecessor 以外から Successor へ

Predecessor 以外からメッセージを受信し，Successor へ送信するパターンである．このパターンは受信エリアの始点ノードが前の受信エリアの終点ノードからのメッセージを受信エリア内のノードへ中継していることを意味する．攻撃者は，監視しているノードがある受信エリアの始点だということは分かるが送信者と受信者の情報を得ることは出来ない．

(5)Predecessor から Successor 以外へ

Predecessor からメッセージを受信し Successor 以外へ送信するパターンである．Successor 以外のノードへの送信はエリアの終点となるスーパーノードにおいて，メッセージを復号してから行うため，提案方式では発生しないパターンである．

(6)Predecessor 以外から受信のみ

Predecessor 以外からの受信を確認したが，送信は確認できないパターンである．このパターンは前の受信エリア空のメッセージをスーパーノードが受信し，復号に成功して次の受信エリアへと送信した場合に発生する．攻撃者は，スーパーノードが始点となる受信エリアがあるということは分かるが，次の受信エリアの有無を知ることや送信者と受信者の情報を得ることは出来ない．

(7)Successor 以外へ送信のみ

どこからの受信も確認されず，Successor 以外への送信のみが確認できるパターンである．このパターンはスーパーノードが送信者である場合か，受信エリアの終端でありメッセージが復号され，次の受信エリアへ送信された場合に現れる．攻撃者は監視しているノードが送信者であるのか受信エリアの終点であるかの判別をするだけの情報を得ることは出来ないため，送信者と受信者の情報を得ることは出来ない．

(8)Predecessor 以外から Successor 以外へ

Predecessor 以外からメッセージを受信し，Successor 以外へ送信するパターンである．このパターンは提案方式では発生しない．

5.1.2 攻撃者が結託した場合

複数の攻撃者が結託して攻撃を行った場合について述べる．複数の攻撃者が通信解析攻撃を行い，得られた情報を共有する事によって送信者と受信者を特定しようとする場合である．

(1) 攻撃者の中にスーパーノードが存在しない場合

まず攻撃者の中にスーパーノードが存在しない場合について考える．メッセージが復号される前と後でパケットの同一性を見出すことができないため，パケットの流れを追うことができるのは受信エリア内だけである．このため匿名通信路全体を知ることは出来ない．また，送信者と受信者の情報を得ることも出来ない．

(2) 攻撃者の中にスーパーノードが存在する場合

攻撃者の中にスーパーノードが存在する場合について考える．ある匿名通信路中の全てのスーパーノードが結託した場合，メッセージの復号を行うスーパーノードが攻撃者に含まれることで，メッセージの復号前後でパケットの対応をとることが可能である．このため匿名通信路の全体を知ることができる．しかし，送信者が結託する場合は考えなくてよいため，攻撃者は匿名通信路の全体を知ったとしても，それが全体であると確信することは出来ない．なぜならば，知り得た匿名通信路の前にノードがいるかどうかわからないためである．このため，スーパーノードが結託したとしても送信者と受信者の情報を得ることは出来ない．

5.2 メッセージをやりとりする時間の計測

本節では提案方式でメッセージをやりとりする場合の時間を計測した結果を述べる．

5.2.1 計測環境

計測に用いた計算機の構成を表 5.1 に示す．これは名古屋工業大学 CSE の教育用端末である．この計算機 32 台を提案方式のシステムに参加させて本節の計測を行った．

5.2.2 公開鍵の配布にかかる時間の計測

公開鍵の配布にかかる時間を図 5.2 に示す．縦軸はエリアの全てのノードに公開鍵が配布される時間であり，単位はミリ秒である．横軸は 1 エリアに含まれるノードの

表 5.1: 計測環境

CPU	Sempron 2800+ (2GHz)
メモリ	1GB
Java	version 1.6.0_02

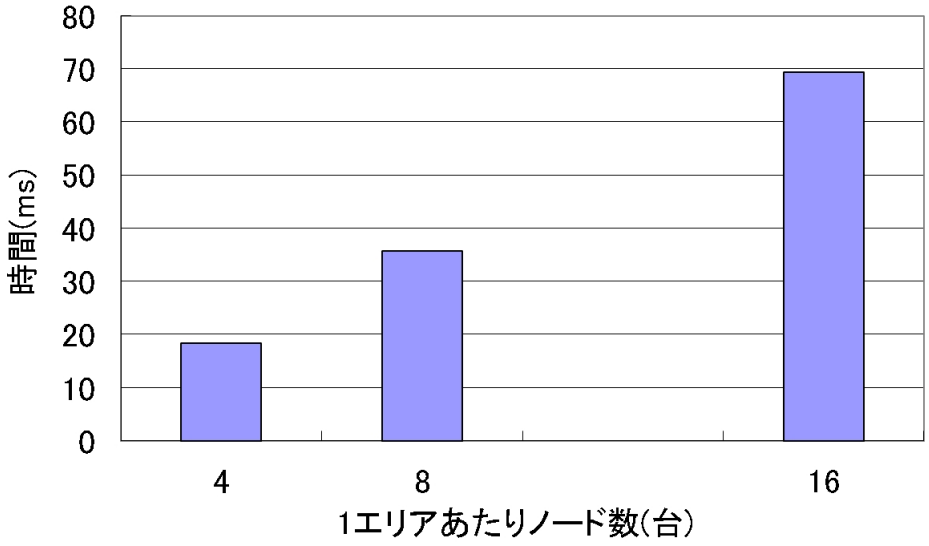

図 5.2: 公開鍵の配布にかかる時間とノード数の関係

数であり，この中には 1 台のスーパーノードを含む．1 ノードは表 5.1 に示される計算機 1 台が担っている．

5.2.3 データサイズを変化させた場合の通信時間

データサイズを変化させて通信にかかる時間を計測した結果を図 5.3 に示す．縦軸は時間であり，単位はミリ秒である．横軸はメッセージのボディ部に含まれる通信内容のデータサイズであり，単位はキロバイトである．データサイズ以外の経路情報等のサイズは 4KB 程度である．1 ノードは表 5.1 に示される計算機 1 台が担っている．中継回数は 24hop，ヘッダの復号回数は 5 回となる経路を指定し，計測を行った．

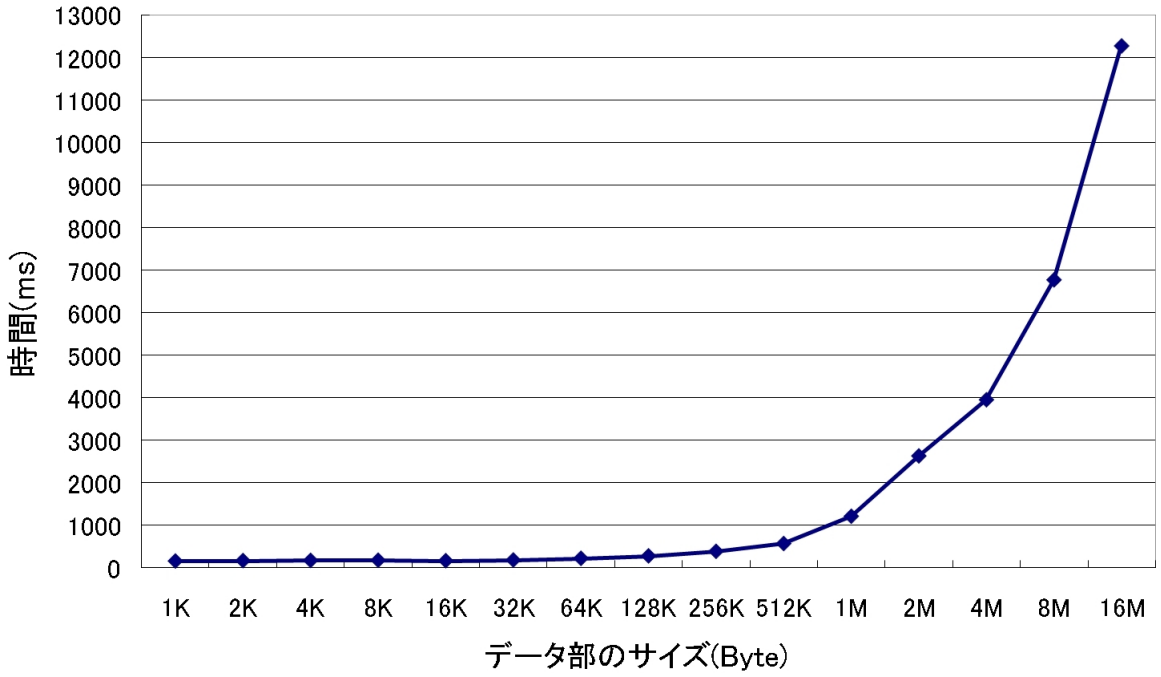

図 5.3: データサイズと通信にかかる時間の関係

5.2.4 考察

図 5.2 から，提案方式において公開鍵の配布にかかる時間は 16 ノードの場合で 70 ミリ秒程度である. ノード間の距離が離れて通信遅延が大きくなったとしても数秒から数十秒で配布が完了すると考えられる．しかし途中参加のノードは参加時に他のノードからスーパーノードと公開鍵の一覧を受け取るため，数十秒待たないと通信できないわけではない. つまり公開鍵の配布にかかる時間とは，スーパーノードと公開鍵の情報を更新するために要する時間である.

また，図 5.3 から 64KB の場合で約 2Mbps，2MB の場合で約 6Mbps の通信速度が得られていることが分かる．ノード間の距離が離れて通信遅延が大きくなった場合でもホップ数を調整すれば，WEB ページの閲覧やメールの送受信などそれほど帯域を必要としない物であれば使用に耐えると考えられる.

表 5.2: 通信路構築時間の比較

提案方式	既存方式
2964 (ms)	21144 (ms)

表 5.3: メッセージ伝播速度の比較

提案方式	既存方式
672 (ms)	1112 (ms)

5.3 既存方式との比較

本章では 2.4 節で述べた既存方式と提案方式の，メッセージが受信者に伝わるまでの時間の比較と，匿名通信路の構築に要する時間を比較を行った結果を述べる．

5.3.1 通信路の構築にかかる時間

両方式で通信路の構築にかかる時間を計測した．全ノード数は 32 台に設定し，表 5.1 の計算機 32 台を用いて評価を行った．1 つの管理エリアには 8 台を割り当てた．中継ノード数は行き 24 台、帰り 24 台ののべ 48 台とした．そしてヘッダ部の多重暗号化回数は 6 回に設定した．結果を表 5.2 に示す．

既存手法では，構築メッセージが中継される際には全てのノードで復号が試みられる．これは，エリアの終端がどのノードであるか分からないためである．一方，提案方式では構築メッセージが中継される際に復号を試みるのはスーパーノードのみである．このため復号にかかるコストが既存方式よりも少ない．よって提案方式の方が，通信路の構築にかかる時間が短い．この差は匿名通信路の中継ノード数が増えれば増えるほど大きくなる．メッセージの並列送信と合わせて，既存手法よりも通信速度の向上を見込むことができる。

表 5.4: 既存方式との比較

手法	Onion Routing	Crowds	2.4 節の方式	提案方式
送信者匿名性	○	○	○	○
受信者匿名性	○	×	○	○
つながりの匿名性	○	△	○	○
可用性	×	△	○	○
スケーラビリティ	△	×	△	△
応答時間	△	○	△	○

5.3.2 メッセージの並列送信による効果

5.3.1 節と同じ通信路でメッセージをやりとりする場合の時間を計測した．データ部のサイズは 512KB とした．結果を表 5.3 に示す．既存方式では，メッセージを送信してから返信を受けとるまでに少なくとも中継を 48hop とヘッダ部の復号 6 回が必要である．提案方式ではスーパーノードを 4 台とし，1 つの管理エリアに 8 台ずつ割り当てている。この時，最後のノードにメッセージが伝わるまでの最短経路は，中継を 24hop とヘッダ部の復号 6 回となる．

表 5.3 の結果より，既存方式よりも提案方式の方が後続の受信エリアに，高速にメッセージを伝えることが可能である事が確認された．これは、メッセージ送信の並列化により最長 hop 数が削減された効果が洗われているためである．このため表 5.3 に示す結果が得られた．また，全ノード数や 1 管理エリアあたりのノード数，ノード間の距離による通信遅延が増えるほど並列送信の効果が得られると考えられる．

5.3.3 比較

既存方式と提案方式を匿名性，可用性，スケーラビリティ，応答時間についての比較を表 5.4 にまとめた．

Onion Routing

Onion Routing は 2.2 節で述べた通り，送信者の匿名性，受信者の匿名性，つながりの匿名性の 3 つ全てが保たれる．しかし，中継ノードが 1 台でも離脱した場合には再度フラッディングを用いて経路を生成する必要があるため，可用性が低い．また，経路生成にフラッディングを用いるためスケーラビリティも低い．応答時間については，全てのノードを必ず中継して復号処理を行うため時間がかかる．

Crowds

Crowds は 2.3 節で述べた通り，受信者か中継ノードのどちらにメッセージを中継するかを確率で決定する方法である．このためメッセージがどのノードから発生したかを特定することは困難であり，送信者の匿名性は保たれる．しかし，受信者へのメッセージ到達を保証するために受信者の匿名性を保つことが出来ない．また，通信路の中継ノードが 1 つでも離脱すると受信者からの返信を受け取ることが出来なくなり，通信をやり直すしかないためそれほど可用性は高くない．スケーラビリティについては Crowds のグループのメンバシップを管理するためにサーバが必要であり，低いと言える．しかし，多重暗号化を行わないため Onion Routing よりも応答時間は短い．

ノード管理に DHT を用いた匿名通信方式 (2.4 節の方式)

ノード管理に DHT を用いた匿名通信方式は 2.4 節で述べた通り，Chord を用いたノード管理と Onion Routing の多重暗号化を組み合わせた方式である．多重暗号化により Onion Routing と同様に全ての匿名性が保たれる．また，ノード管理に Chord を用いることで中継ノードが離脱したとしても匿名通信路の修復を行うことが出来る．このため可用性が高いといえる．しかし，全てのノードの公開鍵をサーバで管理し，加えて一定時間ごとに公開鍵の更新を行うためスケーラビリティが高いとは言えない．

応答時間について述べる．2.4 節の方式では Onion Routing と同様に多重暗号化を行うが Onion Routing よりも回数が少なくて済む，しかし全中継ノードを中継後に受信者へメッセージが届くため時間がかかる．

提案方式

提案方式は 3 章で述べた通り，2.4 節の方式と同等の匿名性と可用性を備えている．なおかつメッセージの並列送信の効果で 2.4 節の方式よりも早く受信者にメッセージが届く．このため Onion Routing よりも復号にかかる時間が短いという利点が生きる．よって応答時間は短いといえる．しかし，公開鍵の配布にかかる時間がノード数の増加とともに増えるためスケーラビリティが高いとは言えない．

第6章

まとめ

本論文では，鍵の配布機能を有し柔軟な経路生成が可能な匿名通信路を提案した．提案方式ではオニオンルーティングで用いられる多重暗号化と Chord によるノード検索を組み合わせて用いていることで離脱耐性の高い匿名通信路を実現している。さらに、スーパーノードを導入することで，公開鍵の配布機能とメッセージの並列送信を実現した．これにより公開鍵サーバが必要なくなり，公開鍵サーバを監視されることで送信者と受信者の情報が漏れる心配がなくなる．また，メッセージを並列送信することで後続ノードに従来手法よりも高速にメッセージを伝えることが可能となった．

また，提案方式をオーバーレイ構築ツールキットである Overlay Weaver を基盤に提案方式の実装を行い，評価を行った．しかし，今回の評価結果はノード数の動的変化が無い前提で行われている．現実の環境に近い状況で動作させ，評価するためにはノードの動的な参加や離脱に対応する必要がある．また，通信解析攻撃に対する防御策として隣接ノード間でやりとりするパケットを，処理時間のかからない簡単な暗号化で違うパケットであるかのように擬装する必要がある．これらの部分の実装と評価に加え，各ノード間の物理的距離が離れている場合の評価，スーパーノードが管理するノード数とスケーラビリティの評価が今後の課題である。

謝辞

本研究のために多大な御尽力を頂き，日頃から熱心な御指導を賜った名古屋工業大学の齋藤彰一准教授に深く感謝致します.

また, 本研究の際に多くの助言，協力をして頂いた松尾啓志教授，津邑公暁准教授，松井俊浩助手，及び齋藤研究室ならびに松尾・津邑研究室の皆様に深く感謝致します。

参考文献

[1] Pfitzmann, A. and Waidner, M. : Networks without user obserbility, Eurocrypt ’85, LNCS 219, pp. 245–253 (1986).

[2] Dingledine, R., Mathewson, N. and Syverson, P. : Tor: The Second-Generation Onion Router, Proceedings of the 13th conference on USENIX Security Symposium, Vol. 13, pp. 303–320 (2004).

[3] Reiter, M. K. and Rubin, A. D. : Crowds: Anonymity for web transactions, ACM Trans. Information and System Security, pp. 66–92 (1998).

[4] 首藤一幸, 田中良夫, 関口智嗣. オーバーレイ構築ツールキット Overlay Weaver. 情報処理学会論文誌:コンピューティングシステム, Vol. 47, No. ACS15, (2006)

[5] Goldschang, D. , Reed, M. and Syverson, P. : Onion routing for anonymous and private internet connections, Comm. ACM, Vol. 42, No. 2, pp. 39–41 (1999).

[6] Reed, M. G., Syverson, P. F. and Goldschlag, D. M: Anonymous connections and Onion routing, IEEE Journal on Specific Areas in Communications, Vol. 16, No. 4, pp. 482–494 (1998).

[7] Stoica, I., Morris, R., Karger, D., Kaashoek, F. and Bal akrishnan, H. : Chord: A Scalable Peer–To–Peer Lookup Ser vice for Internet Applications, Proc. 2001 ACM SIGCOMM Conference, pp. 149–160 (2001).

[8] 近藤 正基, 齋藤 彰一, 松尾 啓志：DHT を用いた双方向匿名通信路の提案, 電子情報通信学会技術研究報告,Vol. 2008 No. 71 pp. 195–202 (2008).

[9] 石黒 聖久: 匿名通信路のノード離脱に対する通信路継続方式の提案, 名古屋工業大学卒業研究論文 (2009).