Vol.2009-ARC-183 No.22
Vol.2009-OS-111 No.22
2009/4/23

分散ハッシュテーブルによるノード管理を行う匿名通信方式の設計と実装

近藤正基[†1]　田中寛之[†1]
齋藤彰一[†1]　松尾啓志[†1]

本稿では，我々が提案している匿名通信方式の詳細と基本性能評価について述べる．本方式は，分散ハッシュテーブルによるノード管理層と，多重暗号による匿名通信路層の二層構造が特徴である．実装は，オーバーレイネットワークの開発環境である OverlayWeaver を用い, LAN による評価を行った．結果, データサイズ 1MB の通信（往復各 16 中継ノード）で 3.4 秒であった．これは，Web サービスとして一般的に十分実用的な性能である．

Design and Implementation of an Anonimity Communication Method using DHT for Node Management

Masaki Kondo ,[†1] Hiroyuki Tanaka,[†1]
Syoichi Saito [†1] and Hiroshi Matsuo[†1]

Details and performance evaluations of the anonymity communication method proposed by us are described in this paper. One of its characteristics is two layer constructions, a node management layer based on DHT and an anonymity communication layer based on multiply-encoding. This paper evaluates communication performance over LAN of the method implemented using OverlayWeaver which is a tool kit for overlay-network system. A RTT is 3.4 seconds in case of the message size is 1MB and the number of relay-nodes is 16 on the one way. The result is practical speed for Web access.

†1 名古屋工業大学
Nagoya Institute of Technology

1. はじめに

近年インターネットの普及により，高度な機密性が求められる医療や行政と個人との通信にもインターネットが使われるようになっている．このような分野では，通信の秘密をより厳重に守らなければならない場合がある．しかし，通信の内容は暗号化によって守ることができるが,「だれ」が「どこ」と通信を行ったといった通信そのものを機密にすることはできない．つまり，適切な場面では強固な通信の匿名性が必要である．本論文ではオーバーレイネットワークを用いて匿名通信を行う新しい方式の実装と評価について述べる．

インターネットにおける通信 (TCP/IP) では，送受信者の IP アドレスは外部から観測可能である．つまり，ネットワーク管理者や通信記録システムは通信している事実や通信量，頻度などを知ることができる．このため，医療情報検索や内部告発などの慎重に取り扱うべき通信も，ネットワーク管理者や通信記録システムによって観測されている．最悪の場合，管理者の不注意やマルウェア感染などで，通信の事実が明るみに出る可能性がある．これらから自身を守るために，送受信者が外部から特定困難な通信方式が必要である．

送受信者が外部から分からない通信の成立要件として，通信中の各メッセージについて以下の 3 種が挙げられる[1)]．以下，これら 3 種を匿名のための性質 (**匿名性**) と言い，これら匿名性を備えた通信を**匿名通信**，匿名通信が使用する通信路を**匿名通信路**と言う．

- 送信者が特定できないこと (送信者匿名性)
- 受信者が特定できないこと (受信者匿名性)
- 送受信者間を追跡できないこと (追跡不可能性)

この 3 種の匿名性の中で，送信者匿名性が最も重要である．匿名通信の利用者は，まず第一に自らを他者から保護したいと考えるからである．受信者匿名性は，Web などのアプリケーションによっては，送信者匿名性が十分であれば不要な場合もある．しかし, 非公開のサーバーとの通信では，受信者の匿名性も同時に実現する必要がある．また，追跡不可能性は，送信者匿名性か受信者匿名性の一方が成立することで成り立つ性質である．これらの匿名性を実現するには，メッセージに当該送受信者の情報 (IP アドレス，経路情報) が含まれないこと，もしくは特定できないことが必要である．

我々はこれまでに，文献[2)] においてこれら 3 種の匿名性を実現する新たな通信方式を提案した．提案した方式 (以下，本方式という) ではノード管理と匿名通信路の管理をする二層構造で通信を行う．これにより，従来手法では対応することのできなかった中継するノードの突然の離脱に対応することができるようになる．さらに，匿名性の観点からも強固な匿

Vol.2009-ARC-183 No.22
Vol.2009-OS-111 No.22
2009/4/23

名性を有することを示した．さらに，既存手法と比較した場合も十分な有用性が期待できることを示した．本論文では提案した方式[2] の実装報告を行う．

本論文では，2 章では本方式の基盤技術となるオニオンルーティング[3][4] と分散ハッシュテーブルの Chord[5] について述べる．3 章で本方式の詳細と動作について述べ， 4 章で実験結果と考察を行う．最後に 5 章でまとめる．

2. 基盤手法

本章では本方式の基盤となる 2 つの手法について述べる．多重暗号化を用いて送受信者ともに秘匿するオニオンルーティングと，分散ハッシュテーブルの一つである Chord について述べる．以下，匿名通信の始点となるノードを送信者，終点となるノードを受信者という．

2.1 オニオンルーティング

オニオンルーティングは, ネットワーク上に複数配置された中継ノードで構成される．通信方式は，まず送信者が受信者に至る各中継ノードと共有する鍵を送信と返信用に 1 組ずつ生成し, 各中継ノードと共有する．これによって送受信の経路を確立する．次に，これら共有鍵を用いてメッセージを多重暗号化する．図 1 では, 送信者 NS から受信者 NR までの間に中継ノード N_i(i=1, 2, 3) がある．それぞれとの共有鍵を K_i(i=1, 2, 3), 通信内容を V すると，多重暗号化メッセージ Ms は以下に示す再帰的計算で得られる．なお，A||B は A と B を結合したものを表し，IP_n はノード n の IP アドレスを示す．

$$Ms = IP_1||K_1(IP_2||K_2(IP_3||K_3(IP_{NR}||K_{NR}(V))))$$

上記の式により得られたメッセージを経路上の各ノードが共有鍵を用いて復号しながら受信者まで送る．受信者の NR は，メッセージを共有鍵を用いて復号することでデータ V を得る．返信時も同様に多重暗号化したメッセージを復号しながら送信者 NS まで送る．

2.2 Chord

本方式で用いる分散ハッシュテーブルは Chord である．Chord は，環状の ID 空間を用いることから，ID の始点と終点がないという特徴がある．Chord は，参加する各ノードに ID を割りふり，環状な ID 空間を形成する．そして，コンテンツをハッシュ関数を用いて一意な ID との組に変換し，各ノードはそれぞれの ID 範囲のコンテンツを管理するという手法である．また，各ノードがそれぞれ経路表 (FingerTable) を保持している．コンテンツの配置されたノードを発見するために，各ノードが経路表に基づいて繰り返し問い合わせを行う．

図 2 に Chord の動作を示す．ノード (N10) がコンテンツ (ID:65) を探索する場合，N10 は自分が保持する FingerTable の中から ID:65 に一番近いノード (N45) に対して問い合わせを送る．N45 が ID:65 を管理していない場合，同様に自分が保持する FingerTable の中から ID:65 に一番近いノード (N63) に対して問い合わせを送る．このようにして最終的にコンテンツを保持するノードにリクエストを送り，コンテンツを得る．また，N ノードの中から目的のノードの発見する問い合わせの回数は, 最悪でも O(logN) 回となる．

以後, 最初にリクエストを開始するノードを開始ノード (initiator)，あるノード X に対して大きい ID を持つノードの中で最も X に近いノードを X の Successor, X に対して小さい ID を持つノードの中でもっとも X に近いノードを X の Predecessor と呼ぶ．

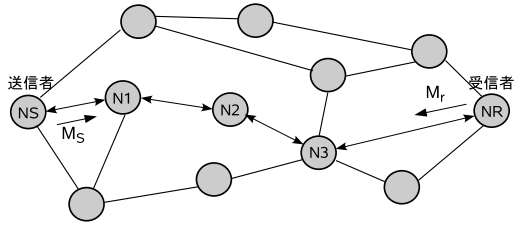

図 1 オニオンルーティングの動作

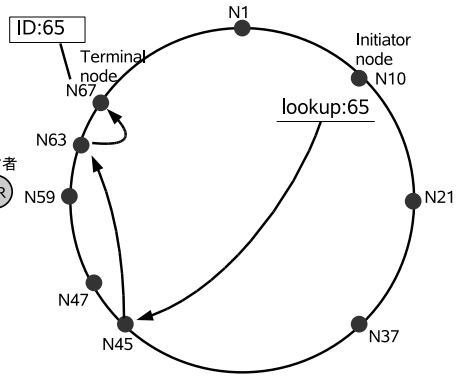

図 2 Chord よる探索

3. 匿名通信方式

本章では，提案する本方式について述べる．まず，本方式の全体構成を示し，続いて通信手順について述べる．

3.1 全体構成

本方式の構成は，オーバーレイネットワークを構成するノード群と，各ノードの公開鍵を管理する**公開鍵サーバ**(PKS) で構成される．全体像を図 3 に示す．各参加ノードは，Chord の参加手続きに基づいてオーバーレイネットワークに接続し，公開鍵サーバに自身の公開鍵を登録する．公開鍵サーバは，参加ノードの ID と公開鍵を関連付けて管理し，要求に応じて参加ノードに提供する．

Vol.2009-ARC-183 No.22
Vol.2009-OS-111 No.22
2009/4/23

本方式は，**ノード管理層**と**匿名通信路層**の二層構造で構成される．ノード管理層は，参加ノードの ID 管理を行う層で Chord による分散ハッシュテーブル機能を有する．この層が，ノードの参加と離脱，経路情報に関する保守を行う．匿名通信路層は，ノード管理層の上位に位置し，匿名通信路の決定，構築，暗号化など，実際の通信を行う層である．

3.2 ノード管理層

ノード管理層は，参加ノードの ID 管理と参加離脱管理と経路情報管理を行う．具体的には，分散ハッシュテーブル Chord を利用し，ノードの参加や離脱は Chord のアルゴリズムに従って処理する．また，匿名通信路層からのノード検索要求に対して，Chord の経路制御機構 (FingerTable) に従って経路制御を行う．

ノード管理層を独立させる利点は，ノード状態管理と匿名通信路の維持管理の依存関係をなくすことができる点である．オニオンルーティングなどに見られる多重暗号化を用いた匿名通信路は，匿名としての性質を保つため，必要最小限 (前後) のノードとしか接続を維持しない．これは，前後以外のノードが分かることで，経路全体の情報を得ることになり匿名性が低下するためである．そこで，本方式ではノード管理を匿名通信路の機能から分離する．これによって，匿名通信路の構成を考慮することなくノード管理が可能となる．ここで本方式では，分散でのノード管理と経路制御が可能な分散ハッシュテーブルに着目し，その中でも環状空間を構成する Chord をノード管理層の基盤とする．

3.3 匿名通信路層

匿名通信路層は，匿名通信路の経路決定と構築を行い，必要な暗号処理を実施して実際に通信を行う層である．匿名通信路層は，ノード ID の割り当て，稼働状況確認を行う必要がない．一方，匿名通信に関するすべての処理を行う．本節では匿名通信路層が行う各処理の詳細について述べる．

3.3.1 通信方式の概要

本方式の通信は，オニオンルーティングと同様の多重暗号を用いる．オニオンルーティングとの違いは，受信者を匿名通信路の途中に配置することで匿名性を向上させることである．さらに，受信者以降の通信がダミー通信となるため，通信を分析して匿名性を低下させる解析攻撃にも有効である．

匿名通信路の構築には，3 種のメッセージを用いる．1 種類目は，匿名通信の開始時に 1 度だけ送信される**構築メッセージ**，2 種類目は通信データを運ぶ**データメッセージ**，3 種類目は**制御メッセージ**である．構築メッセージには，各中継ノードとの共有鍵と受信者との共有鍵，返信用の構築メッセージを含む (共有鍵は，匿名通信路毎，中継ノード毎に異なり，返信用を含めてすべて送信者が作成する)．構築メッセージは，公開鍵暗号を用いた多重暗号化を行い，送信者が指定した経路 (中継ノード群) を辿る．構築メッセージを受信した各中継ノードは，自身の秘密鍵を用いて構築メッセージから送信者との共有鍵を取り出し，匿名通信路毎の**コネクション情報**を作成する．コネクション情報は，経路の識別 ID と，直後のノードの IP アドレスと Chord 空間における ID，当該構築メッセージに含まれていた共有鍵で構成する．コネクション情報は以降のデータメッセージの送受信の際に，使用するべき共有鍵を判断や次に送るノードを判断などに使用する．なお，共有鍵は定期的に更新し，漏洩に備える．

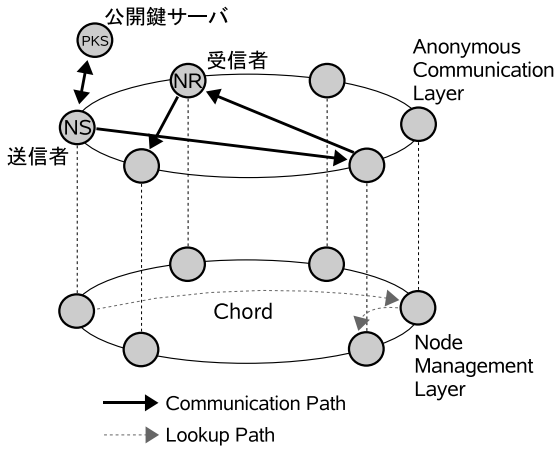

図 3 本方式の全体構成

データメッセージは，一般の通信データを送信者と中継ノードとの共有鍵で多重暗号化したものである．各中継ノードは，コネクション情報に基づいてデータメッセージの受信と復号を行い，次ノードに中継する．制御メッセージは，メッセージ内に制御命令を保持し，当該匿名通信路に対する各種制御に用いる．例えば，経路破棄命令や，鍵更新命令がある．制御命令は，共有鍵で暗号化されており，無関係な匿名通信路を制御することはできない．

匿名通信路が経由する中継ノード群と受信者，返信の経路の決定は，匿名通信路毎に送信者が行う．送信者は通信内容の性格や許容しうる通信遅延時間などを検討して，中継ノード数を決定する．その後，中継ノード数分の中継ノード ID を決定する．この中継ノード ID

Vol.2009-ARC-183 No.22
Vol.2009-OS-111 No.22
2009/4/23

はノード管理層が提供する ID の範囲であれば任意でよい．送信者は，中継ノード ID を担当する中継ノードと受信者の公開鍵を公開鍵サーバから取得し，多重暗号による構築メッセージを作成する．次に，最初の中継ノードを次ノードとするコネクション情報を作成し，指定された IP アドレスに構築メッセージを送信する．構築メッセージを受けとった各中継ノードは，最後の中継ノードに至るまで中継を繰り返す．

3.3.2 受信エリア

本方式は多段中継による通信を行うため，次の問題が考えられる．問題 1) 中継ノード間の経路制御をノード管理層の Chord に依存するため構築メッセージの通信時間が長い．問題 2) 多重暗号のための暗号化と復号処理にかかる時間が長い．そこで，本方式では，これらの問題に対して**受信エリア**を導入し，コスト軽減を図る．

受信エリアとは，Chord の ID の連続した部分空間であり，その ID 部分空間を担当するノード群のことである．受信エリアは以下の性質を有する．

- 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に，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，構築メッセージのヘッダを復号できたノードである
- 受信エリアは 1 つの匿名通信路に複数設定することができる
- 受信エリアはノード 1 つ以上で構成する

つまり受信エリアは，構築メッセージのヘッダに記述された ID から始まり，当該構築メッセージを復号できるノードまで続く．その間の経路制御は，静的に Successor に送るだけの固定経路となる．構築メッセージヘッダを復号できた受信エリアの終点ノードは，復号で得られたヘッダに記載された次の受信エリアの始点ノードに向けてメッセージを中継する．この際の経路制御は，ノード管理部の Chord によるノード検索である．

この受信エリアは次の利点がある．1) 受信エリア内の中継ノード間のノード検索が不要となる．2) 多重暗号化の対象が受信エリア終点ノードに限定できるため，暗号化コストが軽減できる．1) と 2) 共に中継ノードの数を減らすことなく可能である．これは，先の問題 1) と 2) に対する解決策となる．一方，中継ノードが一つでも判明すると当該中継ノードの周りが受信エリアと推測できる．受信エリアがない場合は，1つの中継ノードが判明しても他の中継ノードには影響がない．受信者は中継ノードの一つであることを考慮すると，匿名性を低下させている．以上より，送信者は次の利用方針を考慮して，受信エリアの利用を決定する．1) 高い匿名性を要する場合は，受信エリア内ノード数を 1 として受信エリアを増やすことで匿名性を確保する．2) 高速な通信が必要な場合には，受信エリア内ノード数を大きくして受信エリアを少なくし，暗号化とノード検索コストを抑える．

表 1 構成要素

送信者	NS	匿名通信の始点ノード
受信者	NR	匿名通信の終点ノード
Chord 中間ノード	$R_{i,j}$ $(j = 1, 2, ...)$	受信エリア A_i へ Chord に従い中継するノード
受信エリア数	n	受信エリアの数
受信エリア	A_i $(i = 1, 2, ...)$	NS が指定した ID 空間. 1,2... の順で送信される
受信エリアの最小 ID	$ID_{min}(A_i)$	受信エリア A_i の始点 ID
受信エリアの最大 ID	$ID_{max}(A_i)$	受信エリア A_i の終点 ID
受信エリアの始点ノード	As_i	$ID_{min}(A_i)$ を管理するノード
受信エリアの終点ノード	At_i	$ID_{max}(A_i)$ を管理するノード
公開鍵サーバ	PKS	参加ノードの公開鍵を管理するサーバ
公開鍵	P_{node}	ノード $node$ の公開鍵
公開鍵暗号化データ	$P_{node}(Y)$	データ Y を P_{node} で暗号化したデータ
中継ノード共有鍵 (送信者用)	Cs_{At_i}	NS と中継ノードとの共有鍵
中継ノード共有鍵 (受信者用)	Cr_{At_i}	NR と中継ノードとの共有鍵
受信者との共有鍵	C_r	NS 受信者との共有鍵
初期構築メッセージヘッダ	HS	通信開始前の構築メッセージのヘッダ部
通信途中の構築メッセージヘッダ	HS_i	通信途中の構築メッセージのヘッダ部 i は通信中の受信エリア番号を示す
初期構築メッセージボディ	BS	通信開始前の構築メッセージボディ部
初期返信構築メッセージヘッダ	HR	返信用構築メッセージのヘッダ部

3.4 通 信 手 順

本方式は，メッセージ生成フェーズ，送信フェーズ，返信フェーズからなる．本節では動作の詳細を説明する．表 1 に本方式の構成要素を示す．

3.4.1 メッセージ生成フェーズ

メッセージ生成フェーズの疑似コードを図 4 に示す．また，図 4 中に使われる MakeHeader() と MakeBody() の疑似コードをそれぞれ図 5 と図 6 に示す．送信者は，図 4 のアルゴリズムで構築メッセージを作成する．まず，指定した受信エリアの範囲と配布する共有鍵を受信エリア終点のノードの公開鍵を用いて多重暗号化する (図 5 参照)．多重暗号化した送信先がヘッダ部であり，この構成を図 7 に示す．また，この時作成した共有鍵は保管し，GetKey(*nodename*) により *nodename* の共有鍵を取得することができるとする．次に，受信者が送信者に返信する際のヘッダ部を送信用と同様の方式で生成する．これは送信

Vol.2009-ARC-183 No.22
Vol.2009-OS-111 No.22
2009/4/23

```
GenerateConnectMessage(){
    HS = MakeHeader(AS_IDmin[1〜n] , AS_IDmax[1〜n]);//送信用ヘッダ生成
    HR = MakeHeader(AR_IDmin[1〜n] , AR_IDmax[1〜n]);//返信用ヘッダ生成
    BR = MakeBody(null , Psender , mr, AR_IDmax[1〜n]);//返信用ボディ生成
    受信者との共有鍵Crを生成;
    CAR = null;
    for(i = 1; i <= n; i++)
        CAR = CAR || GetKey(AR_IDmax[i]);//返信用各エリア終端ノードの共有鍵取得
    BS = MakeBody(HR||BR||CAR||Cr, Preceiver, ms,
                                        AS_IDmax[1〜n]);//送信用ボディ生成
    return HS||BS;
}
```

図 4 メッセージ生成の疑似コード

```
MakeHeader (IDmin[1〜n] , IDmax[1〜n]){
    HS = null;
    for(i = n; i >= 1; i–){
        P = PKSよりIDmax[i]の公開鍵を取得;
        C = 共通鍵生成;
        共通鍵(C)を保存;
        HS=P( IDmin[i-1] || C || HS );
    }

    return  HS ;
}
```

図 5 MakeHeader の疑似コード

```
MakeBody (Data, Pnode, m /*受信者のエリアの番号*/,
                                IDmax[1〜n] ){
    BS = Pnode(Data);
    for(i = m-1; i >= 1; i–){
        C = GetKey(IDmax[i]);
        BS = C(BS);
    }
    return  BS;
}
```

図 6 MakeBody の疑似コード

者が返信の際の経路を指定するためである．

次に受信者が送信者に返信する際のボディ部を生成する．これは null のデータを自身 (送信者) の公開鍵で暗号化した後，返信用に「送信者にメッセージが届く以前に現れる受信エリア終点ノード」に配布する共有鍵で多重暗号化したものである (図 6 参照)．例えば，受信エリア数が 5 として，受信者 (送信者に返信する場合は送信者) が 3 番目のエリアに位置したとすると，2 番目と 1 番目の受信エリア終点ノードで多重暗号化する．

次に，送信用のボディ部を生成する．まず，受信者との共有鍵と返信用の受信エリア終点ノードの共有鍵を取得する．その後，返信用のヘッダ (HR) とボディ (BR)，受信者との共有鍵 (Cr)，受信エリア終点ノードに配布する共有鍵 (CAR) を受信者の公開鍵で暗号化した後，受信者にメッセージが届く以前の受信エリア終点ノードに配布する共有鍵で多重暗号化する．以上により作成したヘッダ部とボディ部を結合することでメッセージを作成する．

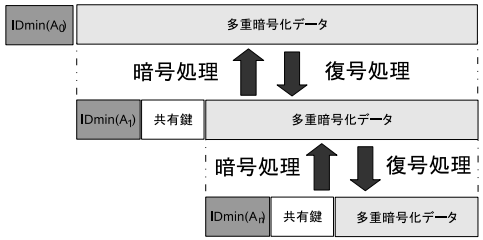

図 7 ヘッダ部の構成

3.4.2 送信フェーズと返信フェーズ

送信フェーズの疑似コードを図 8 に示す．この中の ChordSearch($nextID$) は Chord の検索手法を行って $nextID$ を管理するノードの ID アドレスを返す関数である．SendMessage($address$,msg) は指定した $address$ に msg を送る関数である．また，decrypt($data$) は $data$ を自身が持つ秘密鍵を用いて復号を試み，復号が成功した場合に true を返す関数である．

まず，自身が受信エリア内か外かを判断する．受信エリア外の場合，Chord の検索アルゴリズムにより得られたアドレスにメッセージを送信して，処理を終える．受信エリア内の場合，まず送信ヘッダにある ID を削除する．自身の秘密鍵を用いてヘッダの復号を試みる．復号できなかった場合，Chord 空間での次ノード (=Successor) にメッセージを送る．復号できた場合は，ノードは受信エリア終点ノードである．ヘッダから，送信者との共有鍵と次の ID(=nextID) を得る．この時 nextID が null だった場合，メッセージの送信を終了する．また受信エリア終点ノードは，得られた共有鍵を用いてボディ部 (BS) を復号し，nextID を管理するノードを Chord の検索処理を用いて探索してメッセージを送る．

最後に，受信エリア内のすべてのノードはボディ部の復号を試みる．ボディ部を復号することができた時，当該ノードが受信者である．受信者は，返信の為のメッセージと，送信者との共有鍵，返信の際の受信エリア終点との共有鍵を得る．

返信フェーズはほぼ送信フェーズと同様である．しかし，ヘッダ部，ボディ部は共に復号して得られたもの，つまり送信者が指定したものを用いる．そのため，受信者は送信者の推定が困難になり，受信者から見ても送信者が秘匿される．

3.4.3 動 作 概 要

図 9 に受信エリア数 2 の場合の動作を示す．なお図中には中継ノードと Chord 中間ノー

Vol.2009-ARC-183 No.22
Vol.2009-OS-111 No.22
2009/4/23

```
Relay(){
    msg = 受信メッセージ;

    if(受信エリア外){
        ヘッダより次ノードID(nextID)を取得;
        nextAddress = ChordSearch(nextID);
        SendMessage(nextAddress, msg);
    }
    else{ // 受信エリア内
        If (nextID == 自ノードID)
            msg中のHSからnextIDを削除;

        if(decrypt(HS) == false){ //HSの復号が失敗
            nextAddress = ChordSuccessor();
            SendMessage(nextAddress, msg);
        }
        else{  //HSの復号が成功(受信エリア終端ノード)
            復号したヘッダより次ノードID(nextID)と
                送信者との共有鍵CとBSを取得, 保管;
            if(nextID == null)
                return; // 通信路終端の場合は処理終了
            nextMsg = Cを用いて復号(BS);
            nextAddress = ChordSearch(nextID);
            SendMessage(nextAddress, nextMsg);
        }

        if(decrypt(BS) == true){ // 受信者確認の復号
            HR,BR,共有鍵を入手, 保管;
        }
        セッション情報を登録;
    }
}
```

図 8 送信フェーズの疑似コード

ドのみを記した．まず，送信者 NS は最初の受信エリアの始点ノード $ID(A_1s)$ にメッセージ Ms を Chord のアルゴリズムに基づいて送信する (図 9:Lookup Path)．A_1s からは Successor にメッセージを中継する (図 9:Communication Path)．受信エリア内のノードは，自身の通信用秘密鍵を用いてヘッダ部の復号を試みる．復号できなければ Successor に送る．もし復号できた場合は, 次のエリアの始点ノードの $ID_{min}(A_2)$ を知ることができるので, initiator となって Chord のアルゴリズムに従いメッセージを送る．このときメッセージの対応関係を知られないために，ボディ部を得られた共有鍵を用いて復号する．これを繰り返し行うことで，ヘッダ内に受信エリアがなくなるまでメッセージを送る．

また, 受信エリア内のノード (A_iN_k) はメッセージ中継後に, データ部をデータ用秘密鍵

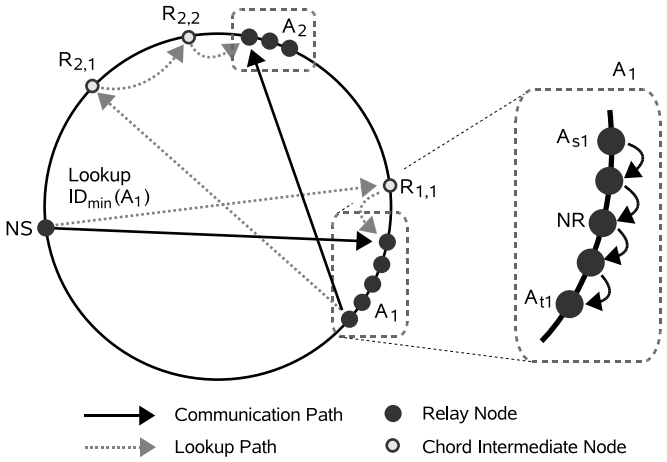

図 9 本方式の経路制御

を用いて復号を試みる．復号できた場合, 当該ノードが受信ノード NR であることが判明する．返信は, 送信メッセージのデータ部に含まれる返信用ヘッダを用いて同様に送る．

データ部とヘッダ部を独立に暗号化することと, 返信時のヘッダを送信者が用意すること, さらに受信後もすべての受信エリアについて送信が続くことで, 送受信者の匿名性と送受信者のつながりの匿名性を確保する．

4. 実装と評価

本章では本方式の実装方法と評価結果を述べる．

4.1 実　　装

実装には，オーバーレイ開発環境である OverlayWeaver[6] を利用した．OverlayWeaver には Chord のオーバーレイネットワーク構築機能，ルーティング機能，メッセージ送受信機能などが提供されている．このメッセージ送受信機能に 3.4 で示した多重暗号化処理，復号処理，経路制御処理を追加実装する形でシステムを実現した．

4.2 性 能 評 価

本方式における遅延時間と通信速度を調べる実験を行った．実験では，100Mbps のイーサ

Vol.2009-ARC-183 No.22
Vol.2009-OS-111 No.22
2009/4/23

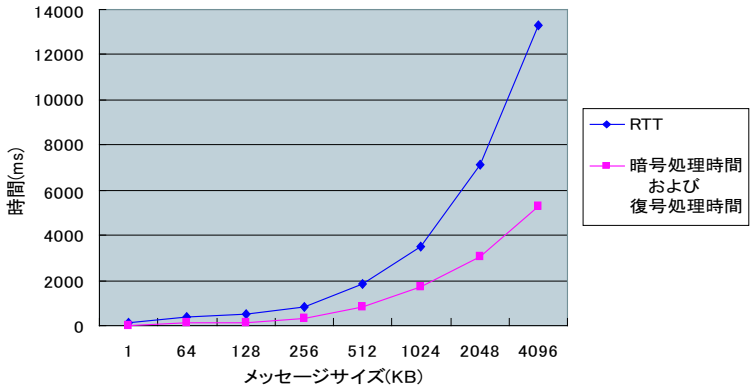

図 10 通信データサイズによる RTT の変化と暗号処理と復号処理による処理時間の変化

ネットにネットワークスイッチを介して接続した 32 台の計算機を用いて，環状なオーバーレイネットワークを構成した．実験した計算機は Sempron/ 2800+，メモリ 1GB，OS は Linux である．32 台の計算機のうち，どの計算機も送信者と受信者になることができる．

今回行った実験の評価パラメータはメッセージサイズ (実験 1)，受信エリア数 (実験 2)，中継ノード数 (実験 3) であり，それぞれを変化させ評価を行った．なお 3.3.1 で示した制御メッセージは実装していない．以下に各実験の概要を示す．

実験 1 往路復路ともに受信エリア数を 2(往復で 4)，16 ホップの経路 (往路で 32 ホップ) を生成し，送信データサイズを $L_M = 1, 64, 128, 256, 512, 1024, 2048, 4096[KB]$ と変化させた場合の RTT と，暗号化処理と復号処理にかかる時間を計測する．

実験 2 経路を 16 ホップ (往復で 32 ホップ) と固定し，受信エリア数を m=1,2,3,4,5 と変化させた場合の経路生成時間と送信データ (100KB) に対する処理時間を計測する．

実験 3 受信エリア数を 5 と固定し，往復で経由するノード数 n=5,10,15,20,25,30 と変化させた場合の経路生成時間と送信データ (100KB) に対する処理時間を計測する．

実験 1 の結果を図 10 に示す．また，実験 2 と実験 3 の経路生成時間をそれぞれ図 11 の (a)，(b) に，共有鍵を用いた 100KB の匿名通信処理時間をそれぞれ図 12 の (a)，(b) に示す．

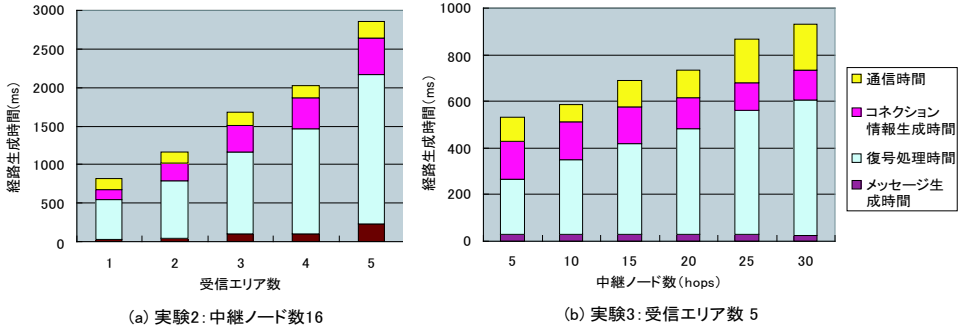

図 11 経路生成時間

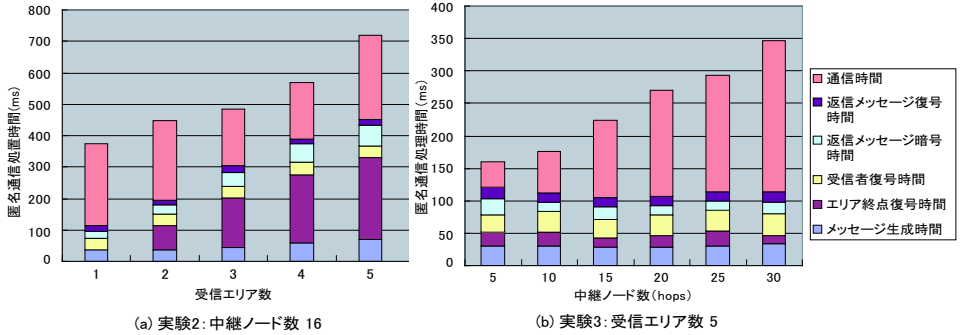

図 12 匿名通信処理時間

4.3 考　　察

図 10 から RTT はメッセージサイズに比例していることが分かる．しかし，送信データサイズが 1MB であっても 2 エリア (往復 4 エリア)，16 ホップ (往復 32 ホップ) の場合に 3.4 秒である．現在運用されている匿名通信である Tor[7] では，4 ホップで RTT が約 2 秒であり[8]，千田氏が実装した手法[9] では 2 ホップで 1MB の通信で約 2 秒である．今回の実

Vol.2009-ARC-183 No.22
Vol.2009-OS-111 No.22
2009/4/23

験では LAN 接続で中継数が 16 ホップであることを考慮に入れると，これらと大差はない．また，Web サービスとして見た場合，十分実用的な性能であると考えられる．

次に図 10 より約半分が暗号化処理時間と復号処理時間であることが分かる．そこで実験 2 と実験 3 により詳細な分析を行う．図 11 と図 12 を比較すると，全体的な処理時間は匿名通信処理時間の方が経路生成時間と比べ高速であることが分かる．これは，経路生成時は公開鍵を用いて暗号化処理と復号処理が必要だが，データ通信では共有鍵を用いるため高速な通信を行うことができるためである．図 11 と図 12 においてはそれぞれの (a) と (b) を比較すると，受信エリア数を変化させる (a) の場合はメッセージ生成時間に増加が見られたが，中継ノード数を変化させる (b) の場合は増加が見られなかった．これは，受信エリア数が多重暗号化の処理回数となるため，受信エリアの増加に伴い送信メッセージの暗号化処理時間が増加するためである．復号処理時間は図 11 の (a) の場合，(b) と比べて大幅に増加していることが分かる．受信エリアが増加することは復号することが可能なノード (= エリア終点ノード) が増えるため，そこでの復号処理コストが増加することを意味する．図 11 の (a) は (b) と比べて復号処理時間の増加量が多いことから，エリア終点ノードの処理コストは一般中継ノードのコストより大きいと考えられる．そのため受信エリア数を増加させると，中継ノード数を増加させた時より通信遅延が大幅に増加する．

ここで，受信エリア数と中継ノード数を決定するための指針について検討する．攻撃者が本方式の受信者を推定するには，まず受信者がいるエリアの推定を行い，その後にエリア内の受信者を推定するという手順が必要である．3.3.2 でも述べたように，受信エリア数が増えると多重暗号の回数が増えるため，受信者がいるエリアの推定が難しくなる．逆に，受信エリア数が少ない場合は，受信者のいる位置を推定される可能性が高くなる．一方，中継ノード数が増えると受信者である可能性のノードが増えるため受信者の推定が難しくなる．以上から，秘匿性の高い通信を行う場合には，受信エリア数を増やし受信者エリアの推定自体を困難にする．また，秘匿性が比較的低く高速性が重視されるような通信では受信エリア数を減らして，中継ノード数を増やすことで，一定の匿名性を確保しつつ高速な通信を行う．

なお，実験 2 と実験 3 からも暗号化処理時間と復号処理時間が全処理時間の大部分を占めていることが分かる．これは，本実装が Java を用いたことによると考えられる．C 言語を用いて，暗号化と復号処理を高速化することで，さらに高速な通信が期待できる．

5. まとめと今後の課題

本論文では，多重暗号化を用いた通信と Chord を用いたノードの管理二層による匿名通信方式について述べ，評価を行った．多重暗号を行う回数と中継ノード数を変化させた場合の経路生成時間と匿名処理時間の測定と，データサイズを変化させた場合の処理時間を測定した．結果，100KB の通信であれば 16 ホップの通信を行ったとしても 1 秒以下，1 MB の通信であっても 3.4 秒であり，Web サービスとしても一般的に十分実用的な性能であると言える．今後，鍵の交換等の制御メッセージの実装と，そのオーバーヘッドの評価とインターネット環境による性能評価実験を行う予定である．

謝辞 本研究の一部は，財団法人堀情報科学振興財団の研究助成，および文部科学省科学技術研究補助金基盤研究 C(課題番号:20500064) によるものである．

参 考 文 献

1) Pfitzmann, A. and Waidner, M. : Networks without user obserbility, Eurocrypt'85, LNCS 219, pp. 245-253 (1986).
2) 近藤 正基, 齋藤 彰一, 松尾 啓志 : DHT を用いた双方向匿名通進路の提案, 情報処理学会研究報告書, 2008-CSEC-42 , pp. 195-202 (2007).
3) 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)
4) 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).
5) Stoica, I. , Morris, R. , Karger, D. , Kaashoek, F. and Balakrishnan, H. : Chord : A Scalable Peer-To-Peer Lookup Service for Internet Applications, Proc. 2001 ACM SIGCOMM Conference, pp. 149-160 (2001).
6) 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 OverlayWeaver, 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. SIG12(ACS 15), pp. 358-367 (2006).
7) Dingledine, R. and Mathewson, N. : Tor : The Second-Generation Onion Router, Proceedings of 13th USENIX Security Symposium, pp. 303-320 (2004).
8) Andriy Panchenko, Lexi Pimenidis, and Johannes Renner. : Performance Analysis of Anonymous Communication Channels Provided by Tor, The Third International Conference on Availability, Reliability and Security, pp. 221-228 (2008).
9) 千田 浩司, 小宮 輝之, 塩野入 理, 金井 敦 : 不正者追跡可能な匿名通信方式の実装と評価, 情報処理学会研究報告書, 2005-CSEC-29 , pp. 25-30 (2005).