通信路の部分的な再構築が可能な匿名通信方式の設計と実装

石黒　聖久 †　　田中　寛之 †　　近藤　正基 †　　齋藤　彰一 †　　松尾　啓志 †

† 名古屋工業大学
466-8555 名古屋市昭和区御器所町

あらまし　代表的な匿名通信手法の一つにメッセージを多重暗号化することによって匿名性を確保する手法がある．この手法では, 送信ノードがメッセージを多重に暗号化し，複数の中継ノードが復号しつつ, 受信ノードへメッセージを送る. しかしこの手法には，送信時の経路上にある中継ノードが 1 つでも離脱した場合，中継ノード同士による経路の修復が不可能という欠点を伴う．本研究では各中継ノードの代理ノードをあらかじめ選出しておくことで，匿名性を維持したまま経路を修復できるように拡張を行った．ノードの管理には Chord を用い，それぞれのノードの Successor を代理ノードとして選出する．

A Method for Recovery of Anonymous Communication from Node Secession

Kiyohisa Ishiguro†　　Hiroyuki Tanaka†　　Masaki Kondo †　　Shoichi Saito †
Hiroshi Matsuo †

†Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya, Aichi, 466-8555 Japan

Abstract　Some existing anonymity communication technique have been realized by multiple encoding. However if a relay node on a route secedes, the route can't be repaired by only relay nodes. In this paper, we expand an existing anonymous communication system into a recoverable the route by only relay nodes. Each relay node entrusts own routing information to a backup node. When a relay node seceded, the backup node salvages the anonymous route.

1　はじめに

近年, インターネットの普及により, 重要な情報の通信をインターネットで行う機会が増えている. これらの情報の通信には厳重な情報守秘が必要となる. 現在, 暗号化技術の発展により, それらの通信データは守秘されているが, 匿名性は完全でない. しかし, 電子投票や内部告発等, 匿名性を必要とした通信は多く, これからも増加する.

現在, 代表的な匿名通信手法の一つに Onion Routing[1] がある.Onion Routing では複数の中継ノードを介し, データを多重暗号化することで匿名通信を行う. しかしこの手法には, 通信路を構成する中継ノードが離脱した場合, 中継ノード同士による通信路の部分修復が不可能という欠点を伴う. 本稿ではこの Onion Routing のための新しい匿名通信路修復方式を提案する.

以下第 2 章では既存の匿名通信手法である Onion Routing と Tor[2] について説明し, その手法の持つ問題点を挙げる. 第 3 章では本稿の提案方式を説明する. 第 4 章では Tor と提案方

式を比較し, 第 5 章でまとめる.

2 既存手法

2.1 Onion Routing

Onion Routing 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. 送信者が受信者に対してデータを送信する際, 送信データを中継ノードの公開鍵を用いて多重に暗号化する. 各中継ノードは受信した暗号メッセージを復号し, 次ノードの情報を得る. これを受信者までの全中継ノードで繰り返し行う. 以上の中継処理によって, 送信者と受信者が中継ノードと区別出来なくなり, 匿名性を得る. また中継ノードは自身の前後のノードしか把握できないため, 中継ノードが一つでも信頼出来れば, 匿名性は維持される.

Onion Routing の問題点として, 各中継ノードの所持する経路情報が少ないために中継ノード同士による通信路の部分修復が困難という欠点がある. 例えば送信者→ノード X →ノード Y →ノード Z →受信者の通信路において, ノード Y が離脱した場合を考える. ノード Y の隣接ノードであるノード X と Z はその離脱を検知することはできる. しかし, ノード X の持つノード情報は送信者と Y のみであり, ノード Z も同様に Y と受信者の経路情報しか持たない. そのためノード Y の離脱により, 通信路の維持が不可能となる.Onion Routing では送信者が離脱を検知出来ない限り, 通信路の切断に対応することが出来ない. なぜなら送信者以外で修復に必要な経路情報を持つノードは存在しないからである.

2.2 Tor

Tor は Onion Routing を用いて実装された匿名通信システムである. Tor は送信者は受信者との間に複数の匿名通信路を定期的に生成する. 送信者はその通信路の内, 最も新しく生成された通信路を用いて通信を行う. 送信者は定期的に使用する通信路を切り替える. 切り替えの間隔はデフォルトで 1 分である.

ノードの離脱が発生した通信路では, 前述した Onion Routing と同様に以後の通信を行うことは出来ない. しかし通信路の切断が発生した後に, 送信者が通信路を切り替えることで, 再び通信可能となる.Tor による通信路の切り替えは定期的に行われ, 送信者がノードの離脱を検知できない場合でも通信の再開が可能である.

しかしこの手法では送信者は定期的に通信路の生成を行う必要がある. 通信路の生成にはメッセージの多重暗号化, 多段中継と複数の復号処理を伴うため, 生成に必要なメッセージ数と, 送信者および中継ノードに掛かる負荷が大きい.

3 提案方式

本稿では通信路全体の生成なしに通信継続可能な, 新しい通信路維持方式方式を提案する.

3.1 提案方式の構成

提案方式では通信路生成時に通信路を構成する各中継ノードが, 自身の離脱に備えたノードを選出する. このノードを**代理ノード**と呼ぶ. 代理ノードは選出元の中継ノードと同じ経路情報を保有し, 通信路の切断時に離脱した元の中継ノードに代わり通信を復旧する.

しかし, 代理ノードの選出や代理ノードへの経路情報の保管, 離脱検知にはノード間で通信を行う必要がある. このためには, 各ノードは他の参加ノードの状態を保持する必要がある. しかし, 各ノードが独立してノード状態の保持を行うことは一般に困難である.

そのため提案方式では匿名通信を行う層とノード管理を行う層の二つの層に分離する. 以下, それぞれを匿名通信路層, ノード管理層と呼ぶ. 全体図を図 1 に示す. 匿名通信路層の実線矢印が Onion Routing による通信可能な匿名通信路を表し, 破線矢印がノード離脱により使用不可能な匿名通信路を表す. ノード管理層における破線矢印は, 離脱した中継ノードの代わりに代理ノードが中継ノードとなる様子を表す.

匿名通信路層では Onion Routing による匿名通信を行う. 前述の通り, この層では自身の前後

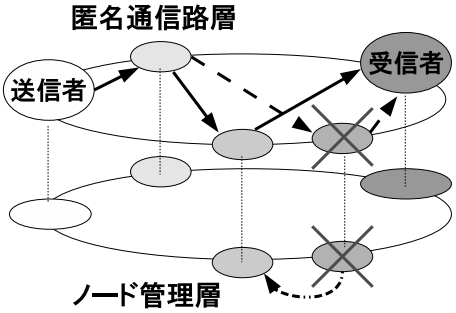

図 1: 全体構成

の経路情報しか持たない. そのためこの層では匿名通信以外の処理を行わない. ノード管理層では Chord[3] を用いてノードの管理を行う. ノード管理層では Chord による検索機能, 安定化処理による離脱ノードの検知機能を用いて, 代理ノードの選出とノードの離脱検知を実現する.

提案方式では Tor のように送信者が定期的に通信路を生成する必要も, 通信路の切り替えを行う必要もない. このため Tor に比べて低コストでの通信路の維持ができる. 以降, それぞれの層での機能の詳細を説明する.

3.1.1 匿名通信路層

匿名通信路層では Onion Routing による匿名通信を行う.2.1 で述べた通り, 各ノードは自身の前後のノードの経路情報 $Info_i$ のみを持つ (表 1 参照).$Node_i$ は送信者側から数えた i 番目の中継ノードを指す. 経路情報には前後のノードと通信を行うための情報とそのノード固有の情報がある. 最初の通信路生成メッセージの生成には Onion Routing と同様に自身の公開鍵暗号を用いる. しかし, 通信の度に公開鍵暗号を用いた通信では効率が悪い. そのため最初の通信路生成以降は暗号と復号がより高速な共通鍵を用いてる. これらの共通鍵は最初の通信路生成メッセージ生成の際に送信者が生成する.

3.1.2 ノード管理層

ノード管理層では Chord による環状ネットワークを形成し, 各ノードの管理を行う. 匿名通信路層と分離することで匿名性を維持しつつ, ノードの検索, 離脱検知が可能となる. 通信路生成時の代理ノードの選出処理, ノード離脱の検知からの通信路の修復処理はこの層で行う.

各ノードは自身の ID 番号周辺の IP アドレスのリスト SuccessorList を保持し, これらのノードとは 1 メッセージで通信できる.$Node_j$ は ID 番号 0 から環状経路に沿って j 番目のノードを指す. 表 2 にそれらリストとそのノードを示す.

表 1: $Node_i$ の持つ経路情報 $Info_i$

情報の種類	前側ノード	後側ノード
IP アドレス	N_{i-1}	N_{i+1}
共通鍵	Key_{i-1}	Key_{i+1}
情報の種類	$Node_i$ の固有情報	
秘密鍵	$PriKey_i$	

表 2: $Node_j$ が有するノード情報

リスト	ノード
SuccessorList	$Node_{j+1}$〜$Node_{j+2logN}$
Predecessor	$Node_{j-1}$

3.2 匿名通信路生成時の処理

匿名通信路生成の際, 送信者は任意の数の中継者を選択し, 匿名通信路生成メッセージを生成する. このメッセージの生成手順は Onion Routing に準ずる. 送信者は生成したメッセージを中継者ノードを経由して受信者に送信する.

このときメッセージを受信した中継ノード $Node_j$ は,Onion Routing の手順に従って以後の通信に使用する $Info_j$ を取得し, 次のノードにメッセージの送信を行う (図 2 左参照).

ノード管理層 (図 2 右参照) では, 自身の代理ノードの選出処理と経路情報の保管処理を行う. ノード $Node_j$ は $Node_{j+1}$ を代理ノードとして選出する. 代理ノードには修復に必要な経路情報 $Info_j$ を保管する必要があるが, 安全性確保のために分割して別々のノードに保管する. 具体的には $Info_j$ を分割経路情報 $Hinfo_{j+1}$ と $Hinfo_{j+2}$ に分割する. 分割経路情報は $Hinfo_{j+1}$

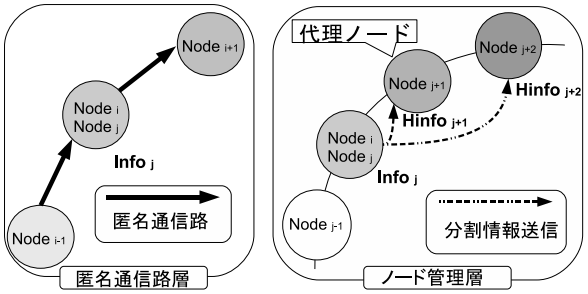

図 2: 代理ノード選出

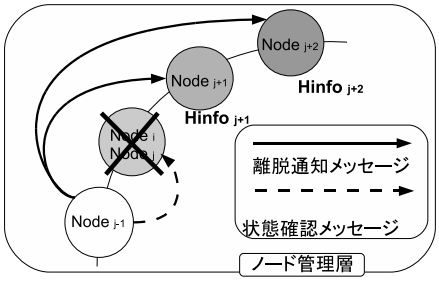

図 3: ノード離脱検知

と $Hinfo_{j+2}$ を結合しない限り, 経路情報として利用できない.$Node_j$ は $Hinfo_{j+1}$ を代理ノードへ,$Hinfo_{j+2}$ を $Node_{j+2}$（以下**保管ノード**とする）に送信する. 代理ノード, 保管ノードに $Node_{j+1}$,$Node_{j+2}$ を使用する理由は表 2 より,$Node_{j+1}$,$Node_{j+2}$ と 1 メッセージで通信可能であるためである. また後述するノードの離脱通知の際にも 1 メッセージで通信できる.

3.3 通信路修復処理

Chord は定期的にノードの状態を確認する安定化処理を行う. 安定化処理では各ノードは自身の Successor に状態確認メッセージを送信する. このとき Successor が離脱していた場合, 離脱したノードの Successor を自身の Successor に変更する. 図 3 の例では,$Node_{j-1}$ が $Node_j$ に状態確認メッセージを送信する. この時 $Node_j$ が離脱していた場合,$Node_{j-1}$ は $Node_j$ の離脱を検知し、代理ノード $Node_{j+1}$ と保管ノード $Node_{j+2}$ に $Node_j$ の離脱を通知する.

離脱検知後の処理手順を図 4 に示す. ノード

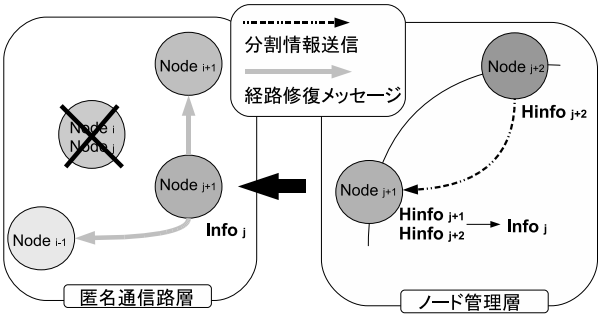

図 4: 通信路修復処理

管理層 (図 4 右参照) で通知を受けた保管ノードは, 代理ノードに自身の保持する分割経路情報 $Hinfo_{j+2}$ を送信する.$Hinfo_{j+2}$ を受信した代理ノードは, それを自身の保持する $Hinfo_{j+1}$ と結合し, 経路情報 $Info_j$ を取得する. 代理ノードは $Info_j$ のよって判明した $Node_j$ の匿名通信路層 (図 4 左参照) での前後のノード $Node_{i-1}$ と $Node_{i+1}$ の IP アドレスを取得する. 代理ノードはこれらのノードに経路修復メッセージを送信し, 通信路の修復を行う.

4 評価

評価として Tor の通信路修復手法との比較を行う. 比較は、通信路修復に必要なメッセージ数と、通信路修復に要する最長時間である. 以下, それぞれの算出方法について述べる.

4.1 Tor の通信路生成メッセージ数

Tor では通信路生成の際に Onion Routing を用い, 中継ノードを介する度に 1 ホップづつ通信路を生成する. この手順を図 5 に示す. 送信者ノード $Node_0$ は最初の中継者ノード $Node_1$ に対して通信路生成メッセージを送信する. 通信路生成メッセージを受けた $Node_1$ は $Node_0$ に対して通信路生成完了メッセージを返信する (図 5 の (1) 参照）.

次に送信者ノード $Node_0$ は最初の中継者ノード $Node_1$ を経由して次の中継者ノード $Node_2$ に対して通信路生成メッセージを送信する.$Node_2$

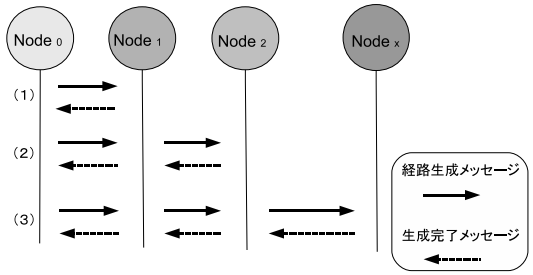

図 5: 通信路生成

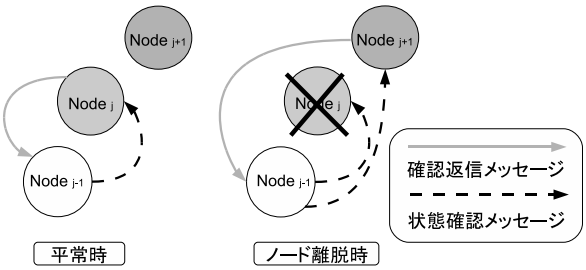

図 6: 安定化処理

も先ほどと同様に中継ノード $Node_1$ を通じて送信者に通信路生成完了メッセージを返信する（図 5 の (2) 参照）. 以降同様の手順を受信者ノード $Node_x$ に到達するまで続行する（図 5 の (3) 参照）. 以上の手順により, 通信路生成のためのメッセージ数は, 中継ノードの数と受信ノードの数に依存する. 中継ノード＋受信ノードの数を x とした場合, 必要なメッセージ数を式 (1) に示す.

$$\sum_{k=1}^{x} 2k = x^2 + x \tag{1}$$

4.2 提案方式の通信路修復メッセージ数

提案方式では, ノードの離脱が発生した場合に, ノード管理層の Chord 修復コストと匿名通信路修復コストの二種類の通信が必要となる.

4.2.1 Chord 修復コスト

提案方式における Chord の修復コストを, 平常時と比較して, 離脱時の各ノードの安定化処理に必要なメッセージ数の増分をコストとする.

ノード離脱の有無による安定化処理の動きの違いを図 6 に示す. 平常時の Chord の安定化処理 (図 6 左参照) では $Node_{j-1}$ から $Node_j$ に対して状態確認メッセージを送信する. 状態確認メッセージを受信した $Node_j$ は $Node_{j-1}$ に対して確認完了のメッセージを返信する. ノード $Node_j$ が離脱した時の安定化処理を説明する (図 6 右参照).$Node_{j-1}$ が $Node_j$ に対して状態確認メッセージを送信したときに,$Node_{j-1}$ は $Node_j$ の離脱を検知する. 離脱を検知した $Node_{j-1}$ は $Node_j$ の次のノードである $Node_{j+1}$ に対して状態確認メッセージを送信する. 状態確認メッセージを受けた $Node_{j+1}$ は, 平常時と同様に $Node_{j-1}$ に対して確認完了メッセージを送信する. 以上よりメッセージの増分は離脱検知に必要な 1 メッセージ分となる.

4.2.2 匿名通信路修復コスト

匿名通信修復の流れは図 4 の通りとなる. 必要なメッセージは離脱を検知してからの 1)$Node_{j-1}$ から代理ノードへの離脱検知メッセージ,2)$Node_{j-1}$ から保管ノードへの離脱通知メッセージ.3) 離脱通知を受けた保管ノードによる代理ノードへの $Hinfo_{j+2}$ の送信. 代理ノードによる $Hinfo_{j+1}$, $Hinfo_{j+2}$ の結合後,4) 代理ノードによる $Node_{i-1}$ への通信路修復メッセージの送信,5) 代理ノードによる $Node_{i+1}$ への通信路修復メッセージ, これら 5 メッセージが必要となる.

結果, 提案方式全体において必要な修復メッセージ数は,Chord 修復コストと合わせて 6 メッセージが必要である.

4.3 通信復旧遅延

通信復旧時間について述べる.Tor の通信復旧時間は, ノードの離脱検出機能がないため, 通信路切り替え間隔に依存する. 文献 [2] によると, 通信路切り替え間隔は 1 分である.1 分より短いと通信路の生成が困難な場合があるため,1 分未

表 3: Tor との通信路復旧コストとの比較

	通信路修復 メッセージ数	通信再開 最長時間
Tor	$x^2 + x$	1 分
提案方式	6	30 秒

注) x は 中継ノード数+受信ノード数 を示す

満での切り替えは困難である. 次に, 提案方式では,Chord の安定化処理間隔に依存する. 文献 [3] によると安定化処理間隔は 30 秒である. 以上より, 通信復旧は提案方式が短時間に行うことが可能である.

本稿では提案方式を OverlayWeaver[4] 上に実装し, 通信復旧時間を測定した. 測定環境は,CPU Sempron 2800+, メモリ 1GB, ネットワーク速度 100Mbps の LAN 環境で, 全 4 ノードによる匿名通信路である.OverlayWeaver の Chord 実装における安定化処理間隔は最長 64 秒であるため, 測定結果は最長 55 秒, 最短 2 秒となり平均 28 秒であった. 安定化処理間隔（64 秒）を考慮すると, この結果は適当である. このように実装依存となるため, 今後匿名通信路用に最適化した安定化処理間隔を求めることが課題である.

4.4 比較

以上の評価のまとめを表 3 に示す. 提案方式では Tor よりも少ないコストでの通信再開が可能であることが分かる. また, 通信路復旧遅延時間は, 通信路切り替え間隔と安定化処理間隔に依存することがわかる. しかし, より短い間隔でこの処理を実行できる提案方式 (Chord) の方が, 短時間での通信路復旧が可能であり, 安定した通信路を提供できると言える.

5 まとめ

本稿では, 匿名通信路のノード離脱による通信路切断の復旧手法として, 代理ノードによる通信路の部分修復手法を提案した. 提案方式ではノード管理層を用いて, 代理ノード選出と経路情報保管を可能にした.

匿名通信路生成の際, 各中継ノードはノード管理層で代理ノードの選出し, 通信路の修復に必要な経路情報を保管する. ノードの離脱が発生した場合, 代理ノードはノード管理層で離脱を検知する. 離脱を検知した代理ノードは保持する経路情報を用いて, 離脱したノードの役割を引き継ぎ, 通信路を部分修復する.

既存手法の Tor との比較した結果, 通信路全体の生成が必要な Tor に対し, 部分修復を行う提案方式の方が低コストでの通信再開が可能である. 今後は Tor の通信の復旧に必要な時間の計測を行い, 提案方式とのより詳細な比較を行う.

参考文献

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

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

[3] 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).

[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 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).