Vol.2013-DPS-154 No.43
Vol.2013-CSEC-60 No.43
2013/3/15

DHTにおけるノード検証手法と匿名通信への適用

中井 俊作[1]　野々山 正峰[1]　齋藤 彰一[1]　松尾 啓志[1]

概要：インターネットにおけるプライバシー保護の重要性が増しており，様々な匿名通信方式が提案されている．匿名通信方式において，そのノード管理に Distributed Hash Table(DHT) を利用するケースが多数ある．DHT を利用したノード管理はスケーラビリティに優れるが，その一方で攻撃に対する脆弱性を持つ．本稿では DHT における近隣ノードと協調し，証明書を配布し合うことで攻撃を検知する手法を提案する．また提案手法は匿名通信システムでの利用を想定し，匿名性を低下させる事なく攻撃の検知を実現する．この手法によって悪意あるノードの攻撃を防ぎ，システム全体の安全性と匿名性を向上させる．

1. はじめに

インターネットには，多種多様なサービスが運用されており，その中には医療相談や人権相談など相談者の匿名性が重要なサービスもある．インターネットにおける通信では，SSL 等の暗号化技術によって通信の内容を秘密にすることは可能であるが，通信を行った事実は，時間と IP アドレスから調べることができる．しかし，サービス利用者のプライバシーを保護するためにはインターネットにおいて誰が，どこと通信を行ったかを秘密にする方法が必要である．そのため，Tor [1] をはじめとして，匿名通信を実現する様々な匿名通信方式が提案されている．その中で，Bifrost [2]，Cashmere [3] 等，多くの既存方式でそのノード管理に Distributed Hash Table(DHT) [4–7] を利用している．DHT を利用することでディレクトリサーバを利用した場合に比べ，高いスケーラビリティを得ることができる．しかし，DHT は攻撃に対する脆弱性がある．DHT のノード検索では，その応答が正しいかどうか検証する手段が無く，悪意を持ったノードに間違った応答をされる可能性がある．DHT を利用する匿名通信方式において，DHT のノード検索は中継ノードの指定にも利用されるため，ノード検索結果を改ざんすることで悪意を持ったノードが意図的に匿名通信路の中継ノードとなることができる問題がある．悪意を持ったノードが複数の中継ノードとなることは匿名通信の解析攻撃の機会を増やすことに繋がり，結果的に匿名性が低下する．

この課題を解決するためには，DHT に対する攻撃を検知して防ぐ方法が必要である．また，匿名通信システムで利用するために，攻撃の検知方法の匿名性を実現しなければならない．我々は，DHT における近隣ノードと協調し，証明書を配布し合うことで攻撃を検知する手法を提案する．この提案により匿名性と安全性を両立させたノード管理が可能となる．

提案方式は既存のノード検証手法である Myrmic [8] をベースとしている．Myrmic では検証対象となるノードの近隣ノードに直接アクセスするため，匿名性が低下する．提案方式では，匿名性の低下を防ぐために各ノード間で証明書を配布し合い，検証対象ノードから検証に必要な証明書を直接受け取ることによって匿名性を保ったまま Myrmic と同等の高いノード検証能力を実現している．

本稿では，2 章で関連研究について述べ，3 章で既存手法の問題点について述べる．4 章で既存手法の問題点を解決する提案方式について述べ，5 章で提案方式を匿名通信に適用する手法について述べる．6 章で評価を行い，7 章でまとめる．

2. 関連研究

本章では，既存の DHT に対する攻撃と，それに対する既存の解決手法について述べる．

2.1 既存の DHT に対する攻撃

既存の DHT に対する攻撃とその匿名通信に対する影響について述べる．DHT に対する攻撃は Sybil Attack [9]，Routing Attack と Storage Attack [10] に分けることができる．このうち，Sybil Attack と Routing Attack はノード検索に関する攻撃である．一方，Storage Attack は，DHT のノードに保管したデータに対する攻撃である．匿名通信において DHT を使用する理由は，ノード検索のスケーラビリティを活用するためであり，ストレージとしての利用

[1] 名古屋工業大学 466-8555 愛知県名古屋市昭和区御器所町

Vol.2013-DPS-154 No.43
Vol.2013-CSEC-60 No.43
2013/3/15

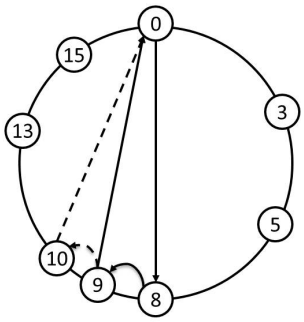

図 1　Routing Attack

ではない．よって，Storage Attack については除外する．以下，DHT のノード検索に対する 2 種類の攻撃について述べる．

2.1.1 Sybil Attack

攻撃者が多数のノードを参加させ，結託した攻撃ノードを増やす攻撃である．これにより，匿名通信路に占める攻撃ノードの割合が増加し，解析攻撃が容易になる．この結果，匿名性が低下する．しかし，SybilAttack は DHT 参加時に認証局にアクセスさせることによって比較的容易に防ぐことができる．

2.1.2 Routing Attack

検索メッセージに対して自ノードが責任担当ノードでない ID に対して応答をする等，間違った応答を行う攻撃である．図 1 はノード 0 が ID10 を検索し，ノード 9 に Routing Attack を受けている状況を示している．本来であれば ID10 の担当はノード 10 であり，図では破線となっている通信が正常である．しかし，ノード 9 が，自身がノード 10 だと主張，もしくはノード 10 は存在しないと主張し，ノード 0 に応答している．DHT をノード管理に利用した匿名通信システムでは，匿名通信における中継ノードをノード ID で指定し，DHT の検索によって接続する．この時 Routing Attack を受けると意図しないノードが中継ノードとなる可能性がある．この攻撃によっても，匿名通信路の一部を攻撃ノードが占めることになるため解析攻撃が容易となり，匿名性が低下する．本稿では Routing Attack の検知手法について述べる．

2.2 Wide Path を利用する利点

DHT におけるノード検証手法に Multi Path 方式と Wide Path 方式がある．Multi Path 方式は，検索元ノードが検索先ノードに対して複数の経路で検索を行う方式である．攻撃ノードが検索途中に存在した場合でも，複数の検索の内いづれかは正しいノードに到達することが期待できる．しかし，Multi Path 方式を匿名通信に適用した場合は，送信ノードが検索元ノードとなり複数の検索を行う必要があるため，検索そのものが攻撃ノードに検知されやすくなるという問題がある．一方，Wide Path 方式では検索の経路は一本であり，各ノードが次ノードを検索する場合に多重化する方式 (図 2 参照) である．よって，送信ノードが複数の検索を行う必要がなく，Multi Path 方式と比較して攻撃ノードによる検知の可能性が低い．これにより，Wide Path 方式の方が匿名通信における DHT 検索に適していると言える．

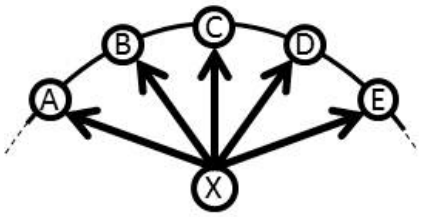

図 2　Wide Path 方式

2.3 Wide Path に基づくノード検証手法

ノード管理に DHT の Chord を用いた匿名通信手法 Torsk [11] では，DHT における攻撃対策として Myrmic と呼ばれる Wide Path 型の検索と認証局によるノード認証を組み合わせた方式を採用している．この方式を以降，Mrymic 方式という．また，Myrmic 方式は，NeighborhoodWatch DHT [12] でも同様な方式が提案されている．以下，Myrmic における Wide Path 型 Routing Attack 対策方式について述べる．

各ノードは DHT への参加時に Neighborhood Authority(以降 NA という) から nCert という証明書を受け取る．nCert にはノードの ID，IP アドレス，Successor List(以降 SL という)，Predecessor List(以降 PL という)，ノードの公開鍵が含まれている．PL とは，各ノードの Predecessor 側（反時計回り）の数ノードを記したリストであり，SL（時計回り）と同様の構造である．この nCert は NA によって署名されるため，各ノードで改ざんすることはできない．nCert は以下のように表せる．

$$nCert_R = Sign_{sk_{NA}}\{pk_R, nList_R, IssueTime_R\}$$
$$nList_R = I_{p^l(R)}, I_{p^{l-1}(R)}, ..., I_{p(R)}, I_R, I_{s(R)}, ..., I_{s^l(R)}$$
$$I_R = NodeID_R, Address_R$$

nCert は NA の秘密鍵 sk_{NA} を用いて電子署名 $Sign_{sk_{NA}}\{\}$ で署名される．また，ノード R の nCert は，ノード R の公開鍵 pk_R と nCert 発行時刻 $IssueTimes_R$，$nList_R$ を含む．$nListR$ はノード R 自身とノード R の SL と PL の ID，IP アドレスを表す．

ノードが新規参加する処理として，NA は参加ノードの ID を DHT で検索し，担当ノードの nCert を取得する．取得した nCert に含まれる SL と PL から参加ノードの nCert を作成する．また，ノードの参加や離脱によって SL と PL

に変更があった場合は，NA によって新たな nCert が作成され，関係する全ノードの nCert が更新される．このため，全ノードは常に最新の SL と PL を記した nCert を保持する．

ここで検証するノード (図 2 のノード X) を原告ノード，検証されるノード (図 2 のノード C) を被告ノードと呼ぶ．原告ノードが ID 検索を行うためには被告ノードに対して nCert を要求し，被告ノードは自身の nCert を返信する．次に原告ノードは受け取った nCert の SL と PL に含まれる全ノードに対して nCert を要求する．要求を受け取った各ノードも，自身の nCert を原告ノードに送る．原告ノードは受け取った全 nCert の SL と PL から，より最適な担当ノードがいないかを確認する．この検証により，被告ノードの SL と PL の中に少なくとも一つは正常なノードが含まれていることで，被告ノードが担当する ID 範囲を正確に確認することができる．これにより，DHT における Routing Attack を防止し，安全な匿名通信路の構築を可能としている．

3. Myrmic 方式における匿名性

Myrmic 方式を匿名通信に取り入れた場合，以下の問題点がある．

- SL と PL の各ノードへのアクセスによる匿名性の低下
- 匿名通信路経由で検証する場合の先端ノードによる情報の改ざん
- 匿名通信路経由で検証する場合の確認時刻での有効性の確認

本節ではこの問題の詳細について述べる．

3.1 SL と PL の各ノードへのアクセスによる匿名性の低下

Myrmic 方式では，ノード検証を行う際に被告ノードの SL と PL の各ノードに nCert を要求する．このため，原告ノードがノード検証を行っていることが被告ノードの SL と PL の各ノードに伝わる．匿名通信システムにおいて，ノード検証は匿名通信路を作成する時に用いられることが考えられるため，ノード検証を行っていることが被告ノード以外に伝わることは匿名通信路の経路情報の漏えいを意味し，匿名性の低下に繋がる．

3.2 匿名通信路経由で検証する場合の先端ノードによる情報の改ざん

多重暗号による匿名通信路は，暗号化と復号を繰り返しながらメッセージを中継する複数の中継ノードで構成されている．匿名通信路を通して Myrmic 方式のノード検証を行う際の通信の様子を図 3 に示す．ノード X がノード Y とノード Z を中継ノードとする匿名通信路を通して，ノード C を被告ノードとする検証を行っている．ここで，原告ノードが送信ノード (図 3 のノード X) の場合，匿名通信路の先端部分に当たる中継ノード (先端ノード，図 3 のノード Z) が被告ノードと被告ノードの SL と PL(図 3 のノード A，B，D，E) から nCert を受け取る．先端ノードが受け取った nCert は NA によって署名されているため，改ざんすることはできない．しかし，先端ノードは過去に受け取った nCert を利用し，nCert 自体を過去のものと入れ替えて原告ノードに中継することで情報の改ざんが可能である．この改ざんが行われた場合，新規に参加したノードの存在を隠蔽するという形の Routing Attack が可能である．

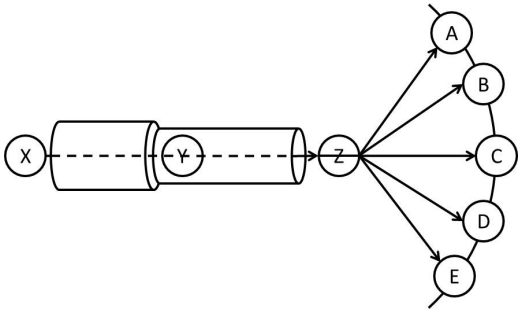

図 3 匿名通信における Myrmic 方式

3.3 匿名通信路経由で検証する場合の確認時刻での有効性の確認

前節同様に，匿名通信路を通して Myrmic 方式のノード検証を行う場合，原告ノードが被告ノードおよび被告ノードの SL と PL の各ノードから nCert を受け取るには，中継ノードを介するため，通信に時間を要する．このため，原告ノードは受け取った nCert がどの時点での情報か判断できないという問題がある．

4. 提案手法

近隣ノードと協調することで，匿名性を保ったまま攻撃を検知する方法を提案し，匿名性と安全性を両立したノード管理を実現する．ノードは自身の nCert と合わせて近隣ノードの nCert を原告ノードに示すことで，検索に対して正しい返答をしていること，つまり Routing Attack をしていないことを証明する．本章では，近隣ノードの nCert を取得し，nCert を利用して攻撃を検知する具体的な方法を述べる．

4.1 概要

本提案手法は，3.1 節で述べた nCert を近隣ノードから受け取ることによる匿名性の低下を防ぐために，各ノード間で nCert を配布し合い，原告ノードは被告ノードから近隣ノードを含めた nCert を受け取る．また，3.2 節で述べた nCert の入れ替えを防止するために，nCert 群に署名を施す．さらに，3.3 節で述べた確認時刻での有効性を確認するために，nCert の配布時刻を付加する．nCert を受け

取った原告ノードは Myrmic と同様の SL と PL による担当ノードの検証を行い，さらに，各ノードによって付加された時刻を用いて nCert の有効性を検証する．

4.2 nCert の拡張

Myrmic 方式で利用した nCert を拡張する．Myrmic 方式の nCert の構成要素であるノードの ID と IP アドレスとノードの公開鍵とそのノードの SL と PL に加えて，NA との時差，nCert 初回配布時刻，nCert 配布周期の 3 つを追加する．これを NA が署名をしたものを本提案手法の nCert とする．nCert は以下のように表せる．

$$nCert_R = Sign_{sk_{NA}}\{pk_R, nList_R, Times_R\}$$
$$nList_R = I_{p^l(R)}, I_{p^{l-1}(R)}, ..., I_{p(R)}, I_R, I_{s(R)}, ..., I_{s^l(R)}$$
$$I_R = NodeID_R, Address_R$$
$$Times_R = IssueTime_R, TimeDiff_R, FirstDelivery_R, DeliveryInterval$$

nCert は NA の秘密鍵 sk_{NA} を用いて電子署名 $Sign_{sk_{NA}}\{\}$ で署名される．また，ノード R の nCert はノード R の公開鍵 pk_R と $nList_R$，$Times_R$ を含む．$nListR$ はノード R 自身とノード R の SL と PL の ID，IP アドレスを表し，$Times_R$ は発行時刻 ($IssueTime_R$)，ノード R の NA との時差 ($TimeDiff_R$)，nCert 初回配布時刻 ($FirstDelivery_R$)，nCert 配布周期 ($DeliveryInterval$) を含む．

NA との時差は，ノード参加時に NA に対して参加ノードの現在時刻を送信し，NA が自身の現在時刻と比較することによって作成する．NA と各ノード時刻差が明確になることによって，ノード間の時刻差を各ノードが算出することができる．これにより，証明書の有効期間の確認などをノード間の時刻同期を行うことなしに実現できる．nCert 初回配布時刻と nCert 配布周期は NA によって指定され，4.4 節の nCert の配布で利用する．なお，nCert 配布周期は全ノード共通の値である．

4.3 NA によるノード参加処理

NA によるノード参加処理について述べる．システムに参加するノードは NA に自身の現在時刻と自身の公開鍵を送信し，NA からの返答として nCert を得る．NA は参加ノードの現在時刻を基に求めた参加ノードと NA の時差，参加ノードの公開鍵，NA が保持している全参加済みノードのリストから nCert を作成し，参加ノードに送信する．同時に，ノードの参加によって nCert が更新される参加済みノード，つまり参加ノードの SL と PL に当たるノードの nCert を作成して，これらのノードの nCert を更新する．また，nCert の更新に必要になるため，NA はノードと NA の時差，ノードの公開鍵，割り当てたノード ID，アドレスを自身で保持する．

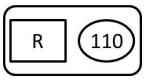

図 4　nCertDlv

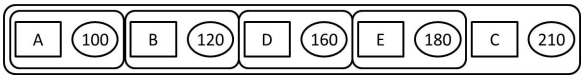

図 5　nCertSet

4.4 nCert の配布形式の配布時刻

各ノードは自身の nCert を SL と PL のすべてのノードに対して配布する．この時，配布する nCert に各ノードの現在時刻を付加し，更にノードの秘密鍵で署名する．この時配布される証明書 $nCertDlv$ は以下のように表せる．

$$nCertDlv_R = Sign_{sk_R}\{nCert_R, CurrentTime_R\}$$

$CurrentTime_R = 110$ の時の $nCertDlv_R$ を図 4 のように示す．$CurrentTime_R$ は各ノードが基準となった時刻であり，それぞれのノードで異なる時間軸である．そこで，nCert に含まれる NA の時差を用いて $CurrentTime_R - TimeDiff_R$ を計算することによりすべてのノードの時間軸を NA を基準とした時刻に合わせる．

nCertDlv の配布は nCert に含まれている nCert 初回配布時刻に従って開始される．この nCert 初回配布時刻は，NA によって近隣ノード間で証明書配布が可能な限り均一になるように指定される．また，nCertDlv の配布は nCert 配布周期毎に繰り返され，ノードが DHT から離脱するまで続けられる．

4.5 nCert の提示方法

ノードは近隣ノードから受け取った nCertDlv と自身の証明書を合わせて現在時刻を付加し，更にノードの秘密鍵で署名したもの (以降 nCertSet という) を原告ノードの検証要求に対する返答として提示する．ノード R の nCertSet を以下に示す．

$$nCertSet_R = Sign_{sk_R}\{nCertDlv_{p^l(R)}, ..., nCertDlv_{p(R)}, nCertDlv_{s(R)}, ..., nCertDlv_{s^l(R)}, nCert_R, CurrentTime_R\}$$

nCertSet の一例を図 5 に示す．図 5 はノードが A，B，C，D，E の順に並んでおり，ノード C が被告ノードとなっている場合の nCertSet である．SL と PL の長さはそれぞれ 2 ずつで，$CurrentTime_A = 100$，$CurrentTime_B = 120$，$CurrentTime_D = 160$，$CurrentTime_E = 180$，$CurrentTime_C = 210$ となっている．

ここで，提案手法の通信の様子を図 6 に示す．図 2 と比較すると原告ノードから被告ノードの SL と PL にアクセスが発生していないことが分かる．nCertSet を被告ノードから原告ノードへ送信することによって，原告ノードが被告

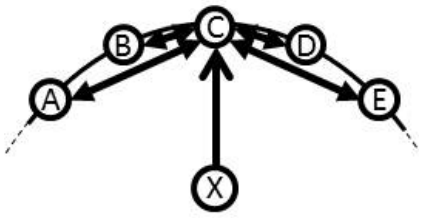

図 6　提案手法における通信

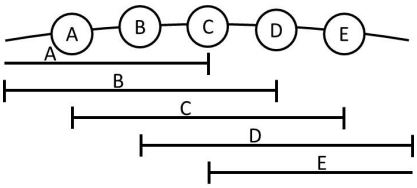

図 7　ノード配置が確認できる範囲

ノードの SL と PL にアクセスする必要が無くなり，3.1 節で述べた Myrmic 方式の問題点の一つである被告ノードの漏えいを防ぐことができる．更に複数の nCert を nCertSet として一つにまとめて被告ノードが署名を行うことにより，3.2 節で述べた匿名通信路経由で検証を行う場合の先端ノードによる改ざんを防ぐことができる．

4.6　確認時刻での有効性確認

本節ではノードが受け取った nCertSet の検証方法を述べる．被告ノードから受け取った nCertSet から得られる情報は，各ノードの SL と PL と nCertDlv を配布した時刻である．

原告ノードは被告ノードとその近隣ノードの SL と PL から，適切な担当ノードが検索に対して返答しているかを確認する．図 5 に示す nCertSet を受け取った場合，nCertSet に含まれているそれぞれの nCert の SL と PL から図 7 で示される範囲のノード配置が確認できる．ノード A，B，C，D，E のどれからでもノード C の配置が確認できることがわかる．このため，ノード A，B，C，D，E のいずれかが正常なノードであればノード C による Routing Attack を検知できる．もし，より適切な担当ノードが存在しているにも関わらず被告ノードが検索に対して返答した場合は，被告ノードは悪意を持ったノードであり，Routing Attack を行っていることが分かる．

以上のように正しい担当ノードが返答しているかの確認を行うと同時に，3.3 節で述べた問題に対応するために各ノードが nCertDlv を配布した時刻を確認する．nCertDlv には各ノードと NA との時差が含まれているため，各ノードが基準となっている時刻を NA 基準の時刻に揃えることができる．各ノードは一定時間毎に nCertDlv を配布しているため，被告ノードが nCertSet を作成した時刻から nCert 配布周期前の時刻より過去の nCertDlv は存在しないはずである．nCert 配布周期 100 とし，図 5 を見ると，$CurrentTime_A$，$CurrentTime_B$，$CurrentTime_D$，$CurrentTime_E$ のすべてが $CurrentTime_C$ から nCert 配布周期である 100 を引いた時刻より後の時刻となっている．よって，ノード C が nCertSet を作成した時刻では正常に nCertDlv の配布が行われ，最新の nCertDlv がノード C に届いていたことが確認できる．次に，異常な場合の例を示す．図 8 を見ると，$CurrentTime_D$ が 60 になっており，$CurrentTime_C$ から nCert 配布周期を引いた時刻，つまり 210-100=110 より過去の時刻となっていることが確認できる．これはノード D が時刻 60 で nCertDlv を配布した後，時刻 160 で再び配布したはずの nCertDlv をノード C が隠蔽していると考えられ，過去の nCertDlv を利用して nCertSet を作成していることを検知できる．

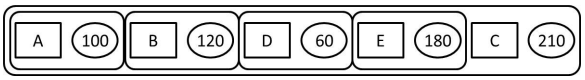

図 8　不正な nCertSet

このように，もし被告ノードが nCertSet を作成した時刻から nCert 配布周期前より過去の nCertDlv が nCertSet に含まれていた場合，被告ノードが最新の nCertDlv を使わず，過去の nCertDlv を利用して nCertSet を作成していることが分かる．

4.7　DHT 検索

提案手法における DHT のノード検索は通常の Recursive Lookup と Iterative Lookup を利用可能である．両 Lookup において検索で得られた担当ノードに対して提案手法のノード検証を行うことで Routing Attack を受けていないか確認することができ，安全なノード検索となる．

また，ノード検索時の経由ノードからも nCertSet を受け取ることによって，より安全にノード検索を行うことができる．Recursive Lookup で検索メッセージに検索元メッセージを含めない場合，Myrmic 方式では匿名通信で利用する場合と同様の改ざんの可能性がある．しかし，提案手法ではすべての情報が nCertSet として一纏めとなって署名されているため，改ざんを防ぐことができる．

4.8　穴空き証明書

提案手法を利用する際の問題として穴空き証明書の問題がある．これは，被告ノードが nCertSet を作成する際に必要な nCertDlv の一部を含まずに作成する問題である．必要な nCertDlv が欠けている nCertSet を以降穴空き証明書と呼ぶ．ノードが A，B，C，D，E の順で並んでいる時のノード C の nCertSet が穴空き証明書となっている例を図 9 に示す．ノード C の PL はノード A とノード B であり，SL はノード D とノード E である．しかし，この nCertSet にはノード B の nCert が含まれていないため，必要な nCertDlv が欠けている状態である．

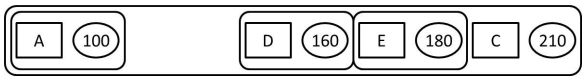

図 9　穴空き証明書

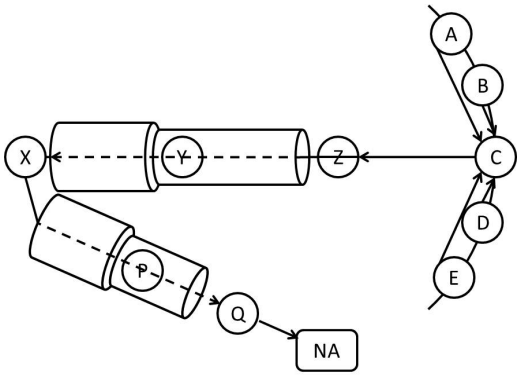

図 10　NA への問い合わせ

この問題が発生する原因には次の 3 つがある．1 つ目は被告ノードの近隣ノード (図 9 ではノード B) が悪意を持ったノードの場合である．悪意を持ったノードが意図的に自身の nCertDlv の配布を行わないことによって近隣ノードが nCertDlv を得ることが出来ず，やむを得ず nCertSet に穴を空けることが考えられる．

2 つ目は被告ノード (図 9 ではノード C) が悪意を持ったノードの場合である．被告ノードが自身に都合の悪い情報を隠蔽するため，nCertSet に穴を空けることが考えられる．この場合はノード B とノード C の間では通常通り通信が行われ，実際にはノード C はノード B から nCertDlv を受け取っている．しかし，ノード B が，ノード C がノード B から nCertDlv を受け取ったことを証明する方法はない．

3 つ目は通信に失敗した場合である．通信の失敗によって nCertDlv を得ることが出来ず，やむを得ず nCertSet に穴を空けることが考えられる．この場合は悪意のあるノードが存在しなくても発生する．

穴空き証明書から得られる情報では，これらの 3 つの内のいずれが原因であるかを特定することができない．しかし穴空き証明書を許容した場合，悪意を持ったノードが自身に都合の悪い情報を隠蔽することも許容することになるため，安全性の低下に繋がる．また，穴空き証明書を提出したノードをブラックリストに入れる方法を取ることもできない．この方法を取った場合，悪意のあるノードが nCertDlv を配布しないことによって近隣の正常ノードをブラックリストに入れることができるようになり，結果的に悪意を持ったノードが中継ノードになる確率を上げることになる．

この問題を解決するために，NA に匿名通信路を通して問い合わせるという手段を用意する．NA に問い合わせる動作を図 10 に示す．ノード C からノード Y と Z で構成された匿名通信路を介して穴空き証明書となっている nCertSet を受け取ったノード X は，ノード Y と Z とは別のノード P を中継ノードとした匿名路を介して NA に問い合わせる．提案手法では，各ノードは参加時に必ず NA にアクセスすることから，NA は全ノードの配置を把握している．そのため，NA に問い合わせることによって正確なノード配置を知ることができ，Routing Attack を受けていないことを確認できる．なお，NA に問い合わせる時に匿名通信路を利用する理由は，NA に対してノードの検証を行っているノード (匿名通信路の送信ノード) を漏えいしないためである．

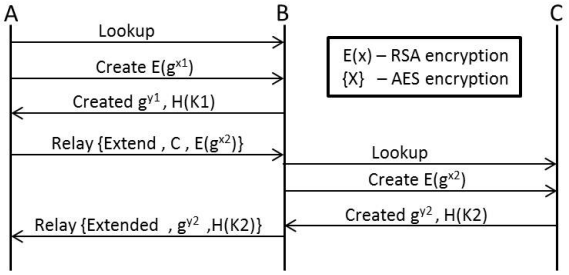

図 11　匿名通信路構築手順

しかし，この解決方法が頻繁に使用されることはないと考える．悪意を持ったノードの目的は匿名通信路の中継ノードとなって情報を集め，送信者と受信者を特定することである．悪意のあるノードが穴空き証明書を渡した場合，NA に検証され，不正が発見された場合には中継ノードにはなれない．また，悪意のあるノードが近隣ノードの nCertSet を穴空き証明書にした場合も，NA に問題無しと判断されるため，悪意のあるノードは中継ノードになれず，メリットが無い．よって，穴空き証明書の問題は NA に問い合わせるという解決策を用意することで防ぐことが可能であると考える．

5. 匿名通信への適用

本章では 4 章で提案した手法を実際に匿名通信に適用する方法について述べる．

5.1 Telescoping

Telescoping は Tor で利用されている匿名通信路構築手法である．Telescoping は多重暗号方式を取っており，送受信ノード間に複数のノードを配置し，それらがメッセージを暗号化と復号しつつ中継を行うことにより実際の送信ノードと受信ノードを隠蔽する．Telescoping は，Diffie-Hellman 鍵交換を利用して生成した共通鍵で暗号化と復号を行い，匿名通信路を 1 台ずつ延長する方式である．匿名通信路作成の手順を図 11 に示す．A，B，C はノードを示す，縦軸は下向きに時間経過を示している．Lookup は DHT の検索を示す．この検索結果を匿名通信路の中継ノードとして利用する．Tor では DHT を用いていないため，管理サーバ

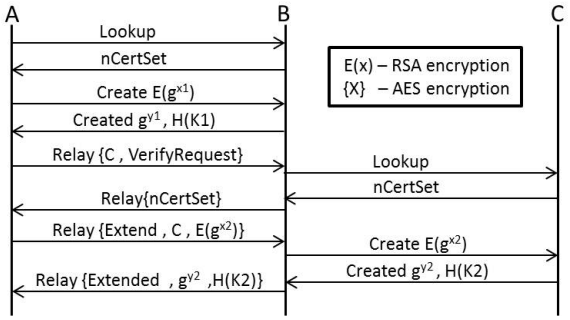

図 12 提案方式を適用した匿名通信路構築手順

よりノード情報を得ている．Create，Created は DH 鍵交換を示しており，これで共通鍵を作成する．Relay は多段暗号によるデータを示し，そのデータは DH 鍵交換によって作成された共通鍵で暗号化されている．Extend は匿名通信路の拡張要求であり，受け取ったノードは指定されたノードに Create メッセージを送信する．その後 Created メッセージを受け取り，結果を中継する．

5.2 Telescoping への適用

Telescoping への提案手法の適用について述べる．Telescoping では一台ずつ中継ノードを増やすため，提案手法を適用することですべての中継ノードを対象として検証するが可能である．提案手法を適用した Telescoping による匿名通信路作成の手順を図 12 に示す．匿名通信路の中継ノードを増やす Create メッセージを送る前に，Lookup で得られた結果を検証する nCertSet の通知を追加している．この nCertSet の検証によって Lookup 結果のノードが Routing Attack を行っているかを検証できる．もし Routing Attack を行っている悪意のあるノードだった場合，解析攻撃を受ける可能性が高まり，匿名性の低下に繋がる．このため，提案手法による検証によって匿名通信の安全性を高め，匿名性向上を実現することができる．

6. 実装と評価

6.1 実装

提案手法を実装し，Telescoping に適用した．提案手法の実装について述べる．実装は openssl を用いて C++で実装した．暗号化は公開鍵暗号に RSA，共通鍵暗号に AES 暗号を使用し，鍵長は 1024bit とした．以下の機能を実装した．

- NA
- DHT のノード検索
- nCertDlv の配布
- Telescoping に提案手法を適用した匿名通信路構築

NA は，予め公開鍵を公開し，nCert を検証できるようにする．DHT のノード検索は Recursive Lookup である．なお，穴空き証明書に関する処理の実装は今後の課題である．

表 1 計測環境

CPU	Core2Duo 3GHz
メモリ	3GB
ネットワーク	1000BASE-T

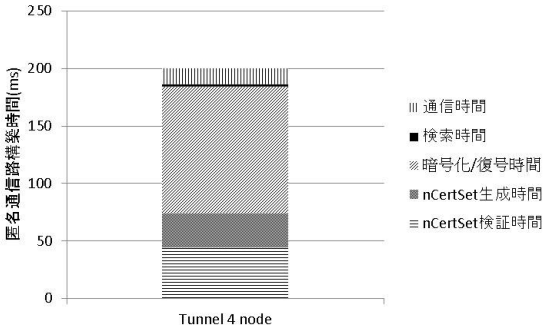

図 13 匿名通信路構築時間

6.2 評価

速度評価に用いた計算機の構成を表 1 に示す．この計算機 32 台を提案方式のシステムに参加させて，匿名通信路構築時間と nCert 発行時間の計測を行った．

6.2.1 匿名通信路構築時間

提案方式による匿名通信路構築時間を計測した．中継ノード数を 4 とした匿名通信路を構築し，以下の各時間を計測した．DHT のノード検索を行う時間，検索結果を検証するための nCertSet を作成する時間，nCertSet を検証する時間，暗号化と復号を行う時間，全体からそれらを引くことで求めた通信時間である．

計測結果を図 13 に示す．提案手法によるオーバーヘッドは nCertSet 生成時間と nCertSet 検証時間と通信時間の一部である．通信時間を除く 2 つの増加時間は約 70ms である．匿名通信路構築後に行われる実際の通信では提案手法のオーバーヘッドは無いため，実用の範囲内であると言える．全体で 200ms 程度であるが，実際のインターネット環境では通信時間と検索時間が大幅に増大すると考えられるため，インターネット環境での評価が今後の課題である．

6.2.2 nCert 発行時間

NA による nCert 発行時間を計測した．nCert 発行時間は NA が保持している全ノードリストを参照して SL と PL を特定し，その他の nCert に含める情報と共に秘密鍵で署名するまでの時間である．計測結果は，nCert 発行 1 回あたり 0.9ms であった．提案方式では，ノードが参加する時に必要な処理は NA から nCert を取得することのみであるため，nCert 発行時間を基にシステムが処理可能な参加と離脱のノード数を推測できる．nCert はノード参加時に，参加したノードとその SL と PL の各ノードに対して発行されるため，SL と PL の合計ノード数によってノード参加処理時間は変化する．SL と PL の長さを変化させた時のノード参加処理時間を図 14 に示す．SL と PL を 8 ずつ，

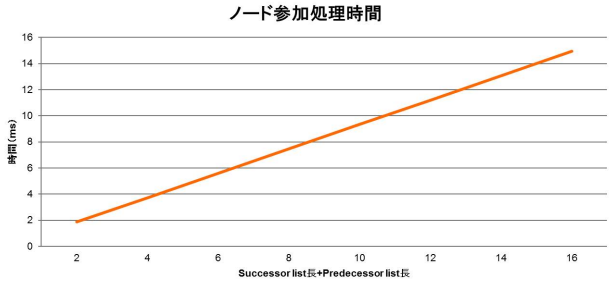

図 14 ノード参加処理時間

合計 16 ノード取った場合で参加ノード処理時間は 15ms 程度であり，この場合 1 秒間に 63 ノード程度の参加を処理できる．

7. まとめ

本稿では，各ノードが自身の近隣ノードへ証明書を配布することで，匿名性を低下させることなく悪意のあるノードからの攻撃を検知し，安全性を向上させる手法を提案した．提案方式では各ノードが証明書に現在時刻を付加して自身の近隣ノードに配布する．被告ノードは近隣ノードから配布された現在時刻が付加された証明書をまとめて提出する．原告ノードは提出された証明書と，それらに付加された時刻を基に被告ノードの近隣を含めたノード配置と，受け取った証明書自体の有効性を検証することができる．

また，提案方式を既存の匿名通信手法である Telescoping に適用し，その適用方法と動作について述べた．提案方式は匿名性を低下させることなしに安全性を高め，悪意のあるノードから解析攻撃を受ける機会を減らすことができる．したがって，参加者の匿名性を向上させている．

提案方式を実装し，正しく動作していることを確認するとともに，評価を行った．評価は LAN 環境で中継ノード 4 台の匿名通信路を構築する時間と，認証機関である NA の証明書発行時間についてである．匿名通信路構築は LAN 環境で 200ms となり，提案手法による時間の増加は 70ms 程度となった．NA による証明書発行は約 0.9ms となり，SL と PL を 8 ノードずつとした場合において 1 秒間に約 60 ノードのノード参加離脱処理が可能である．

今後の課題としては，NA の分散化が挙げられる．NA はノードの参加離脱の際にアクセスされるため，スケーラビリティにおいてボトルネックとなっているため，これを改善する必要がある．

参考文献

[1] Dingledine, R. and Mathewson, N.: Tor: The Second-Generation Onion Router, *Proceedings of 13th USENIX Security Symposium*, pp. 303–320 (2004).
[2] Kondo, M., Saito, S., Ishiguro, K., Tanaka, H. and Matsuo, H.: Bifrost: A novel anonymous communication system with DHT, *2009 International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp. 324–329 (2009).
[3] Zhuang, L., Zhou, F., Zhao, B. and Rowstron, A.: Cashmere: Resilient anonymous routing, *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, USENIX Association, pp. 301–314 (2005).
[4] 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).
[5] Rowstron, A. and Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, *Middleware 2001*, Springer, pp. 329–350 (2001).
[6] Zhao, B. Y., Kubiatowicz, J., Joseph, A. D. et al.: Tapestry: An infrastructure for fault-tolerant wide-area location and routing (2001).
[7] Maymounkov, P. and Mazieres, D.: Kademlia: A peer-to-peer information system based on the xor metric, *Peer-to-Peer Systems*, pp. 53–65 (2002).
[8] Wang, P., Osipkov, I., Hopper, N. and Kim, Y.: Myrmic: Secure and robust dht routing, *U. of Minnesota, Tech. Rep* (2006).
[9] Douceur, J.: The sybil attack, *Peer-to-peer Systems*, pp. 251–260 (2002).
[10] Sit, E. and Morris, R.: Security considerations for peer-to-peer distributed hash tables, *Peer-to-Peer Systems*, pp. 261–269 (2002).
[11] McLachlan, J., Tran, A., Hopper, N. and Kim, Y.: Scalable onion routing with torsk, *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 590–599 (2009).
[12] Bender, A., Sherwood, R., Monner, D., Goergen, N., Spring, N. and Bhattacharjee, B.: Fighting spam with the NeighborhoodWatch DHT, *INFOCOM 2009*, pp. 1755–1763.