分散ハッシュテーブルを用いた公開鍵管理手法の設計と評価

武田 敦志 *†, チャクラボルティ デバシシュ‡, 北形 元 †‡, 橋本 和夫 †, 白鳥 則郎 †‡
* 東北文化学園大学科学技術学部知能情報システム学科
† 東北大学大学院情報科学研究科
‡ 東北大学電気通信研究所

近年，P2P ネットワークの普及が急速に進んでおり，P2P ネットワーク上で動作する多数のアプリケーションが開発されている．しかし，安全で効率的な公開鍵の分散管理手法が実現されていないため，大規模な P2P ネットワークにおいて公開鍵暗号技術を利用することは困難である．これに対し，我々は，分散ハッシュテーブルと信頼の輪を用いて効率的に公開鍵を管理することにより，ノード間の相互認証を実現する分散型公開鍵管理手法 Hash-based Distributed Authentication Method (HDAM) を提案してきた．しかし，従来の HDAM には不正な内部ノードに対する耐性が低いという問題があった．そこで本稿では，複数の分散ハッシュテーブルを並列に利用することにより，不正な内部ノードが存在していた場合，従来より確実に正しい公開鍵を入手する公開鍵管理手法 S-HDAM を提案する．また，コンピュータシミュレーションを通じて，不正な内部ノードが存在していた場合，S-HDAM が従来手法よりも確実に公開鍵を入手できることを確認する．

Public-Key Management Scheme using Distributed Hash Table and its Performance Evaluation

Atushi TAKEDA*†, Debasish CHAKRABORTY‡, Gen KITAGATA†‡,
Kazuo HASHIMOTO† and Norio SHIRATORI†‡
***Department of Intelligent Information System, Tohoku Bunka Gakuen University**
†Graduate School of Information Sciences, Tohoku University
‡Research Institute of Electrical Communication, Tohoku University

In recent years, P2P networks have been evolving at a rapid pace, and a lot of applications which runs on P2P networks have been developed. However, it is difficult to use public key encryption on large P2P networks, because a secure and efficient scheme for public key management is not realized. Therefore, we proposed Hash-based Distributed Authentication Method (HDAM) which is a decentralized public key management method. HDAM realizes an efficient decentralized public key management mechanism by using Web of Trust and Distributed Hash Table. HDAM, however, is not resistant to insider attacks which are performed by inside nodes of HDAM network. In this paper, we propose S-HDAM, which is a secure scheme of public key management. It makes a public key management system resistant to insider attacks by using several Distributed Hash Tables. Thus, in S-HDAM system, users can get an valid public key, even if attackers are in the system. Simulation result shows that proposal method realizes more secure communication than before.

1 はじめに

P2P ネットワークはサーバとなる計算機を必要としないネットワークであり，従来のサーバ・クライアントモデルのネットワークと比べて利便性などの点で優れている．一方，運用上の問題として，公開鍵暗号技術を用いたセキュアな通信を行うことが難しいという問題がある．公開鍵を管理し，通信先の計算機端末（ノード）の公開鍵を安全に入手することができれば，公開鍵暗号技術を用いたセキュアな通信を行うことができる．しかし，全てのノードがネットワークへの参加と離脱を繰り返す P2P ネットワークでは，永続的なサービスを提供できるノードが存在しないため，全てのノードから信頼される特定のノードで公開鍵を集中管理することは難しい．また，従来の公開鍵の分散管理手法はスケーラビリティが低いという問題があった．

そこで我々は，P2P ネットワークに参加しているそれぞれのノードが相互に公開鍵を管理する，スケーラブルな分散型の公開鍵管理手法 Hash-based Distributed Authentication Method (HDAM) を提案してきた [1, 2]．HDAM は P2P ネットワークに参加しているノード間の相互認証を実現するために，信頼の輪と分散ハッシュテーブルを用いて公開鍵を分散管理する．HDAM はスケーラビリティに優れた公開鍵の分散管理手法であり，P2P ネットワークに参加する各ノードに必要となるメモリ量と公開鍵管理のための通信データ量を大幅に削減する．一方，HDAM は信頼の輪に参加している複数のノードを経由して公開鍵を入手するため，信頼の輪に不正な公開鍵を配布するノードが含まれていた場合，正しい公開鍵を入手することが出来なくなる問題があった．

そこで本稿では，信頼の輪に少数の不正ノードが含まれていたとしても正しい公開鍵を入手可能な分

散型公開鍵管理手法 S-HDAM を提案する．提案手法の基本は，信頼の輪に含まれる複数のノードから同一ノードの公開鍵を入手することにより，従来より確実に正しい公開鍵を入手することにある．S-HDAM では複数の分散ハッシュテーブルを並列に利用することにより，複数のノードから同一ノードの公開鍵を入手する．さらに本稿では，コンピュータシミュレーションによる S-HDAM の性能評価を通じて，S-HDAM に不正ノードが含まれていても HDAM よりも高い確率で正しい公開鍵を取得できることを示す．また，S-HDAM も HDAM と同様にスケーラビリティに優れており，他の手法に比べて少ない通信データ量で公開鍵の管理が可能であることを示す．

2 関連研究

Public Key Infrastructure (PKI) は最も有名な公開鍵管理手法である [3]．PKI では認証局と呼ばれるサーバで各ノードの公開鍵に電子署名を行う．また，各ノードでは通信相手の公開鍵の電子署名を確認することにより，通信相手の公開鍵の正当性を確認する．PKI システムでは認証局の公開鍵を用いて電子書名の確認を行うため，認証局の公開鍵をすべてのノードに対して安全に配布する必要がある．また，すべてのノードと認証局の間には社会的な信頼関係が必要となる．すべてのノードが参加と離脱を繰り返す P2P ネットワークでは，すべてのノードに対して認証局の公開鍵を配布することやすべてのノードと認証局の間に社会的な信頼関係を築くことは現実的ではない．そのため，P2P ネットワークに対して PKI を適用することは難しく，認証局のようなサーバを必要としない公開鍵管理手法が必要とされている．

認証局のようなサーバを必要としない認証手法として Pretty Good Privacy (PGP) がある [4]．PGP は信頼の輪と呼ばれるノード間の信頼関係を活用することによりサーバを必要としない公開鍵管理を実現している．この認証システムに参加している各ノードは，そのノードが信頼しているノードを介して新しい公開鍵を入手することができる．しかし，PGP は効率的に公開鍵を収集するための情報を提供しないため，目的の公開鍵を入手するために多くのメモリ量と多くの通信データ量を必要とする．そのため，大規模 P2P ネットワークに対して PGP を適用することは難しい．効率的な公開鍵の交換を実現するためには，公開鍵管理システムが公開鍵を効率的に入手するための情報を提供する必要がある．

Ad-hoc ネットワークにおいて動作するサーバを必要としない公開鍵管理手法として self-organized public-key management が提案されている [5]．この公開鍵管理システムでは，すべてのノードは隣接するノードより新しい公開鍵を自動的に取得する．しかし，このシステムも効率的に公開鍵を取得するための情報を提供していない．そのため，このシステムを用いて公開鍵の収集を行うためには多くのメモリ量と多くの通信データ量が必要となる．

Ad-hoc ネットワークや OSPF ネットワークなどの特定のネットワーク上において，サーバを必要とせずに効率的な公開鍵の交換を実現する分散型の公開鍵管理手法が提案されている [6, 7]．これらの手法では，信頼の輪とネットワーク通信のルーティング情報を用いることにより，公開鍵の交換に必要となるメモリ量と通信データ量を削減している．しかし，これらの手法は Ad-hoc ネットワークや OSPF ネットワークで用いられるルーティング情報を必要としているため，これらの手法を適用可能なネットワークの種類が限られるという問題がある．

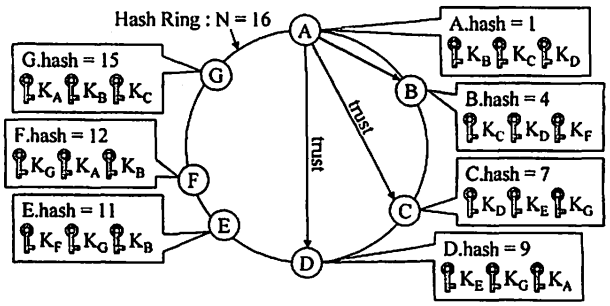

図 1: HDAM による公開鍵の分散管理

これらの手法に対して，我々が提案している HDAM では，信頼の輪を用いることにより分散型の公開鍵管理を実現し，分散ハッシュテーブルを用いることにより効率的な公開鍵の入手を実現する [1, 2]．そのため，HDAM は永続的なサーバを必要とせずに任意のコンピュータネットワーク上で効率的な分散型の公開鍵管理を実現する．

3 Hash-based Distributed Authentication Method (HDAM)

3.1 HDAM の概要

そこで HDAM (Hash-based Distributed Authentication Method) では，信頼の輪と分散ハッシュテーブルを用いることにより，大規模な P2P ネットワークにおける公開鍵の効率的な分散管理を実現する．HDAM では，分散ハッシュテーブルを効果的に用いて信頼の輪を形成する．これにより，公開鍵の効率的な分散管理を実現し，各ノードに必要とされるメモリ量及び各ノードが送受信する通信データ量を従来手法より減少させる．HDAM は従来手法に比べてスケーラビリティの高い分散型の公開鍵管理手法であり，従来手法の適用が難しかった大規模な P2P ネットワークにおける公開鍵の分散管理を実現する．

3.2 DHT を用いた公開鍵の分散管理

図 1 に HDAM による公開鍵の分散管理の例を示す．ここで，$i.hash$ はノード i のハッシュ値，K_i はノード i の公開鍵，N はハッシュ値がとりうる最大の値 (最大ハッシュ値) を表す．HDAM では，ノードの識別子から一方向ハッシュ関数で求めたハッシュ値 ($i.hash$) を基に，1 から N までの指標を円形に配置したハッシュリング上に各ノードを仮想的に配置する．そして，各ノードはハッシュリングにおいて自身の位置から正の方向に $2^k (k = 0, 1, 2, \cdots, \log_2 N - 1)$ 以上離れたノードのうち最も近い位置に配置されたノードの公開鍵を管理する．図 1 の場合，A が管理する公開鍵は以下の 3 個となる．

- 正の方向に $2^1 (2^0)$ 以上離れたノードの中で最も近い位置に配置されたノードである B の公開鍵 K_B

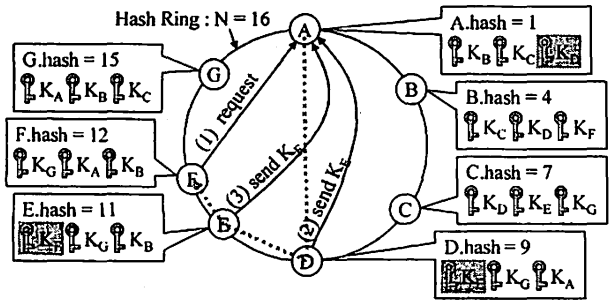

図 2: 公開鍵取得手順

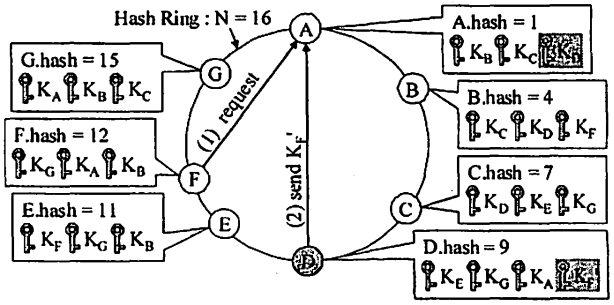

図 3: 内部ノードによる攻撃

- 正の方向に 2^2 以上離れたノードの中で最も近い位置に配置されたノードである C の公開鍵 K_C
- 正の方向に 2^3 以上離れたノードの中で最も近い位置に配置されたノードである D の公開鍵 K_D

参加ノード数が n のとき，各ノードが管理する公開鍵数は $O(log_2 n)$ である．

3.3 信頼の輪と DHT を用いた公開鍵取得

ノード n がノード d の公開鍵を保持していない時に d から n への暗号通信などの公開鍵を必要とする要求が行われた場合，以下の手順により n は d の公開鍵を取得する．

1. n は自身が認証しているノードの中から，ハッシュリング上で最も d に近い位置に配置されたノード n^t に対し，d の公開鍵 K_d を要求する．
2. n^t が公開鍵 K_d を保持している場合，n^t は n へ K_d を送信する．
3. n^t が公開鍵 K_d を保持していない場合，n^t が認証しているノードの中からハッシュリング上で最も d に近いノード n' の公開鍵 $K_{n'}$ を n に送信し，手順 (1) に戻る．

図 2 にノード間の認証手順の例を示す．この例では，前述した手順に従い，ノード A がノード F の公開鍵を取得する．

1. F が A に対して公開鍵を必要とする要求を行う．
2. A が D に対して F の公開鍵 K_F を要求する．この要求に対して，D は A に対して E の公開鍵 K_E を返す．
3. A が E に対して F の公開鍵 K_F を要求する．この要求に対して，E は A に対して F の公開鍵 K_F を返す．

以上の手順により，A は公開鍵 K_F を取得する．参加ノード数が n のとき，公開鍵入手のために必要となる通信データ量は $O(log_2 n)$ となる．

3.4 内部からの攻撃に対する脆弱性

HDAM は信頼の輪を用いて公開鍵の分散管理を実現している．しかし，HDAM の信頼の輪は善良なノードのみで構成されることを前提条件としているため，HDAM にはネットワーク内部のノードからの攻撃に弱いという問題がある．図 3 に内部ノードによる攻撃の例を示す．この例では，悪意のあるノード D が偽造されたノード F の公開鍵 K'_F をノード A に送信した場合を示す．従来の HDAM は善良なノードだけで構成されることを前提としているため，ノード A はノード D から送られてきた公開鍵 K'_F をノード F の正しい公開鍵として扱う．また，従来の HDAM に受信した公開鍵が正しいかどうかを確認する仕組みが存在しないため，ノード A は公開鍵 K'_F が正しい公開鍵かどうかを確認することが出来ない．そのため，ノード A は不正な公開鍵 K'_F を用いてノード F との通信の暗号化や電子署名の確認を試みる．しかし，ノード D からノード A に送られた公開鍵 K'_F はノード F の公開鍵 K_F とは異なるため，ノード A とノード F の間では公開鍵暗号や電子署名を用いたセキュアな通信を行うことが出来ない．

4 提案: Secure-HDAM

4.1 S-HDAM の概要

他のノードから取得した公開鍵が正しいかどうかを確認するためには，複数のノードから同一の公開鍵を取得し，それらを比較すればよい．不正な動作を行うノードにより偽造や改竄が行われた公開鍵は，それ以外のノードから取得しか公開鍵とは異なる値になる．そのため，これらの公開鍵を比較することにより不正な公開鍵を判別することができる．HDAM では，公開鍵を取得するノードを分散ハッシュテーブルによって決定する．そこで本稿では，複数の分散ハッシュテーブルを並列に利用することにより，複数のノードから同一の公開鍵を取得する仕組みを持つ分散型公開鍵管理手法 Secure-HDAM (S-HDAM) を提案する．S-HDAM では，複数のノード配置の異なるハッシュリングを構築し，これらのハッシュリングを用いて公開鍵を管理する．これらのハッシュリングを用いることにより，複数のノードからの同一ノードの公開鍵を取得し，これらの公開鍵を比較することにより，偽造や改竄された公開鍵を判別することが可能となる．すなわち，従来の HDAM より確実に正しい公開鍵を入手することができ，従来の HDAM よりも確実に公開鍵暗号や電子署名を用いたセキュアな通信を行うことが可能となる．

4.2 複数ノードからの公開鍵取得

S-HDAM では，各ノードの配置が異なる複数のハッシュリングを利用する．それぞれのハッシュリング上でのノードの位置は，ノード ID とハッシュリングの No のハッシュ値を基に決定される．ハッシュ値は MD5 などの一方向ハッシュ関数を用いて求められるため，ノードの配置構成は各ハッシュリングで異なる．S-HDAM では，それぞれのハッシュリング上におけるノードの配置位置に基づいて公開鍵の

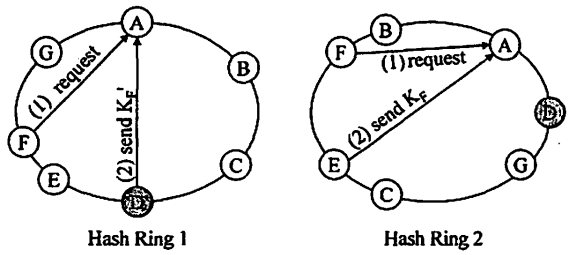

図 4: S-HDAM における公開鍵の取得

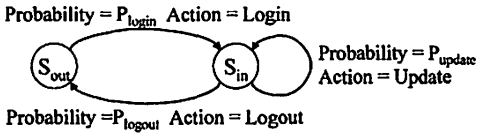

図 5: ノードエージェントの状態遷移図

管理を行うため，目的の公開鍵を取得するために問い合わせるノードは各ハッシュリングで異なる．すなわち，S-HDAM では，複数のハッシュリングを利用することにより，複数のノードから同一の公開鍵を取得することが可能となる．

図 4 に，S-HDAM における公開鍵取得手順を示す．この例では 2 つのハッシュリング (Hash Ring 1, Hash Ring 2) を構築し，これらのハッシュリング上でのノード配置から各ノードで管理する公開鍵を決定している．ノード A がノード F の公開鍵 K_F を取得する場合，それぞれのハッシュリング上で公開鍵 K_F を管理しているノードから公開鍵 K_F を取得する．ここで，ノード A は Hash Ring 1 でノード D から偽造された公開鍵 K_F' を取得し，Hash Ring 2 でノード E から正当な公開鍵 K_F を取得したとする．このとき，ノード A はこれらのハッシュリングで取得した公開鍵 K_F' と K_F を比較することにより，これらの公開鍵のどちらかが不正な公開鍵であることを確認できる．

4.3 必要メモリ量と通信データ量

S-HDAM は複数のハッシュリングを利用することにより複数ノードからの公開鍵取得を実現するため，複数のハッシュリングを構築・維持する必要がある．そのため，1 個のハッシュリングのみを利用していた従来の HDAM と比較して，S-HDAM のシステムには多くのメモリ量と通信データ量が必要となる．具体的には，m 個のハッシュリングを利用する S-HDAM で，各ノードに必要となるメモリ量と通信データ量は従来の HDAM の m 倍である．しかし，S-HDAM は HDAM と同様にスケーラビリティに優れた手法である．そのため，多数のノードが参加する大規模な P2P ネットワークに対して S-HDAM を適用した場合，S-HDAM に必要なメモリ量や通信データ量は HDAM 以外の従来手法に比べて少ない．

5 シミュレーション評価

5.1 P2P ネットワークシミュレータ

S-HDAM の特性を評価し，提案手法の有効性を示すために，Java を用いて P2P ネットワークの動作シミュレータを設計・実装した．このシミュレータで

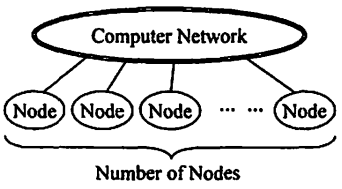

図 6: シミュレーションで想定したネットワーク構成

scenario	P_{login}	P_{logout}	P_{update}	P_{send}
no.1	1.0	0.45	0.05	0.5
no.2	1.0	0.01	0.01	0.98

表 1: ノードエージェントの動作設定値

は P2P ネットワークに参加するノードをエージェント (ノードエージェント) として実現している．必要なメッセージをノードエージェント間で送受信することにより，P2P ネットワークへの参加・ネットワークからの離脱・公開鍵の更新及び公開鍵の取得のために行われるノード間の通信をシミュレートする．

図 5 にノードエージェントの状態遷移図を示す．ノードエージェントは，ネットワークに参加していない状態 (S_{out}) とネットワークに参加している状態 (S_{in}) を持つ．ノードエージェントは状態 S_{out} の時に確率 P_{login} で状態 S_{in} に移行し，状態 S_{in} の時に確率 P_{logout} で状態 S_{out} に移行する．また，状態 S_{in} の時に確率 P_{update} で公開鍵を更新し，確率 P_{send} で任意のノードにメッセージを送信する．送受信する全てのメッセージには電子署名が付加されており 3.3 で述べた手順に基づいて必要な公開鍵を取得し，その公開鍵を用いてメッセージに付加された電子署名の正当性を確認する．また，ネットワークへの参加・ネットワークからの離脱・公開鍵の更新は文献 [1] に記述された手順で行う．

5.2 シミュレーションのシナリオ

図 6 にシミュレーションで想定するネットワーク構成を示す．このシミュレーションでは，すべてのノードはインターネットなどのコンピュータネットワークで相互接続されており，パケットロスなどのネットワーク障害は発生しないものとする．また，このシミュレーションにおけるノード数は，このコンピュータネットワークに参加しているノードの数を指す．

S-HDAM の特性を評価するために，2 種類のシナリオについてシミュレーションを行った．表 1 に，それぞれのシナリオで用いたノードエージェントの動作設定値を示す．シナリオ 1 で用いるノードエージェントはネットワークへの参加と離脱を繰り返すノードエージェントである．このノードエージェントが必要とする公開鍵の数は少ないため，公開鍵取得のために送受信するメッセージ量は少なくなる．これは，センサーデバイスなどに搭載されている小型の通信アプリケーションを想定している．シナリオ 2 で用いるノードエージェントはネットワークに参加してから離脱するまでに多くのメッセージを送信するノードエージェントである．このノードエージェントは受信メッセージの電子署名を確認するために多くの公開鍵を必要とするため，公開鍵取得のため

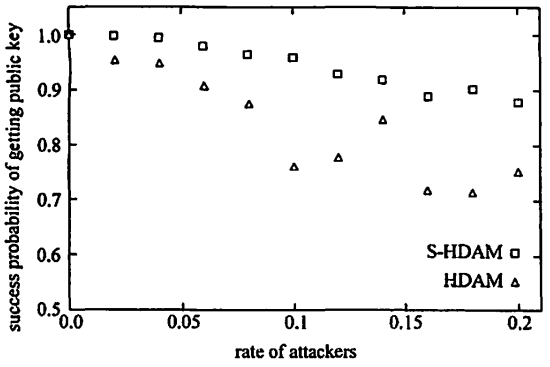

図 7: 公開鍵配布の確実性

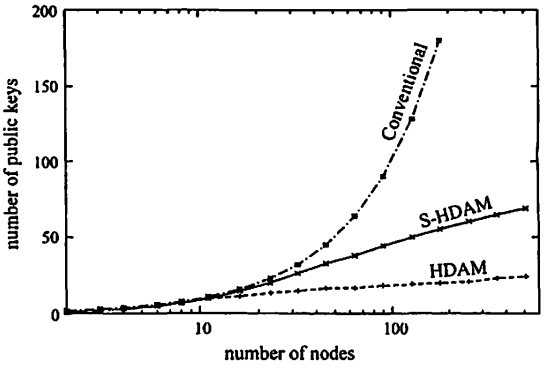

図 8: 各ノードで管理する公開鍵の数

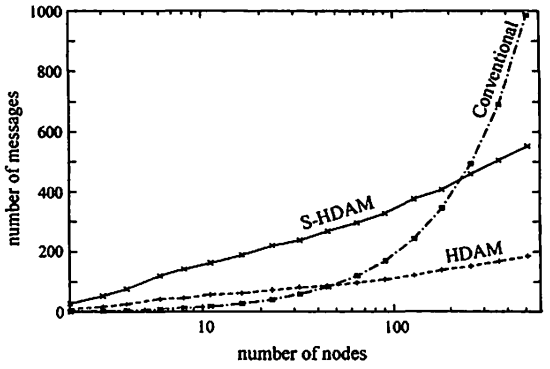

図 9: シナリオ 1 における通信メッセージ数

に送受信するメッセージ量は多くなる．これは，インスタントメッセンジャーなどの通信アプリケーションを想定している．

5.3 公開鍵配布の確実性

図 7 に，不正な内部ノードが存在した場合に，正しい公開鍵を取得できる確率を示す．このシミュレーションでは，S-HDAM が公開鍵管理のために用いるハッシュリングの数は 3 個であり，P2P ネットワークに参加しているノード数は 500 となっている．ここで，“rate of attakers” とは P2P ネットワークに参加しているノードの中に含まれる不正ノードの割合を示す．具体的には，“rate of attackers” が 0.1 の場合，50 個の不正ノードが P2P ネットワークに参加している．S-HDAM・HDAM ともに，不正な内部ノードの数が増えると正しい公開鍵を取得できる確率が低下する．しかし，S-HDAM の正規公開鍵を取得する確率は HDAM のそれに比べて大きい．すなわち，S-HDAM は HDAM よりも確実に正当な公開鍵を取得することが出来る．これは，1 個のハッシュリングしか使用しない HDAM では取得した公開鍵が正当なものであるかを確認することが出来ないのに対して，複数のハッシュリングを利用する S-HDAM では取得した公開鍵が正当なものであるかを確認することが可能となるためである．具体的には，シミュレーションで用いた 3 つのハッシュリングを使用する S-HDAM の場合，3 つの異なるノードから同じノードの公開鍵を入手することが可能である．取得した 3 つの公開鍵の中の 2 つ以上が正当な公開鍵であれば，それを用いて通信を暗号化したり電子署名を確認したりすることが出来る．

5.4 性能評価

S-HDAM の性能を評価するために S-HDAM の性能を測定し，その結果を HDAM と HDAM 以外の従来手法と比較した．従来手法は PGP などの分散型の公開鍵管理手法を想定している [4]．従来手法は分散ハッシュテーブルを用いて信頼の輪を構築することは行わず，ネットワーク参加時に信頼の輪を用いてすべてのノードの公開鍵を収集する．そのため従来手法では，それぞれのノードは公開鍵管理のために多くのメモリを必要とし，ネットワーク参加時に多くのメッセージを必要とする．しかし，それぞれのノードがすべてのノードの公開鍵を所有しているため，メッセージに付加された電子署名を確認するために新たな公開鍵を取得する必要はない．

5.4.1 各ノードに必要なメモリ量

図 8 に，それぞれのノードが管理する必要のある公開鍵の数を示す．ここで，S-HDAM が公開鍵管理のために用いるハッシュリングの数は 3 個である．それぞれのノードが管理する公開鍵の数は各ノードに必要なメモリ量を意味している．S-HDAM では複数のハッシュリングを構築・管理する必要がある．そのため，HDAM と比べて各ノードで管理する必要のある公開鍵の数が多い．このシミュレーションでは，S-HDAM は 3 つのハッシュリングを利用しているため，S-HDAM システムの各ノードで管理される公開鍵の数は HDAM の場合の 3 倍になっている．すなわち，S-HDAM システムの各ノードは，HDAM の場合に比べて 3 倍のメモリ量を必要とする．しかし，S-HDAM は HDAM と同様にスケーラビリティに優れた公開鍵管理手法である．そのため，ノード数が十分に大きい場合，S-HDAM の各ノードに必要なメモリ量は，HDAM 以外の従来手法の各ノードに必要なメモリ量より少ない．すなわち，S-HDAM は多数のノードが参加する大規模 P2P ネットワークに対して適用可能である．

5.4.2 公開鍵管理のための通信データ量

図 9 にシナリオ 1 において各ノードで送受信されるメッセージ数を示す．また，図 10 にシナリオ 2 において各ノードで送受信されるメッセージ数を示す．ここで，S-HDAM が公開鍵管理のために用いるハッシュリングの数は 3 個である．それぞれのノードが送受信するメッセージ数は，公開鍵管理に必要

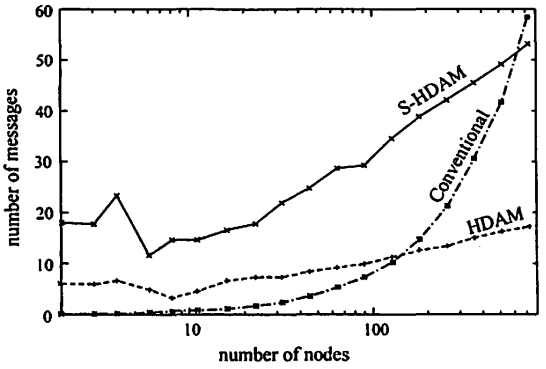

図 10: シナリオ 2 における通信メッセージ数

となる通信データ量を意味している．S-HDAM では複数のハッシュリングを用いて公開鍵の送受信を行うため，公開鍵管理のために必要となる通信データ量が HDAM に比べて多い．このシミュレーションでは，S-HDAM は 3 つのハッシュリングを利用しているため，S-HDAM の各ノードで送受信されるメッセージ数は HDAM の 3 倍になっている．すなわち，S-HDAM は HDAM の 3 倍の通信データ量を必要としている．しかし，S-HDAM は HDAM と同様にスケーラビリティに優れているため，ノード数が十分に大きい場合，S-HDAM に必要な通信データ量は HDAM 以外の従来手法に必要な通信データ量に比べて少ない．図 9 より，シナリオ 1 の場合，ノード数が 250 以上のときの S-HDAM の通信データ量は HDAM 以外の従来手法より少なくなっている．また，図 10 より，シナリオ 2 の場合，ノード数が 700 以上のときの S-HDAM の通信データ量は HDAM 以外の従来手法より少なくなっている．これらの結果より，S-HDAM は HDAM と同様にスケーラビリティに優れた手法であり，多数のノードが参加する大規模 P2P ネットワークに対して適用可能だといえる．

6 おわりに

信頼の輪と分散ハッシュテーブルを用いた公開鍵の効率的な分散管理手法 HDAM はスケーラビリティに優れた手法であるが，不正な内部ノードによる攻撃対して脆弱であるという問題があった．そこで，本稿では，同一ノードの公開鍵を複数のノードから取得することにより，システムに少数の不正なノードが含まれていたとしても正しい公開鍵を取得可能な公開鍵分散管理手法 S-HDAM を提案した．S-HDAM は，複数のハッシュリングを構築・利用することにより，複数ノードから同一の公開鍵を取得することを可能にしている．これにより，S-HDAM では，従来の HDAM よりも確実に正しい公開鍵を取得することが出来る．また，本稿では，コンピュータシミュレーションを通じて S-HDAM を評価することにより，システムに不正ノードが含まれている場合，S-HDAM では HDAM に比べて高い確率で正しい公開鍵を取得できることを確認した．また，シミュレーション結果より，S-HDAM は HDAM と同様にスケーラビリティに優れた手法であり，HDAM 以外の従来手法に比べて少ないメモリ量と通信データ量で公開鍵の管理が可能であることを確認した．

今後の課題としては，公開鍵管理フレームワークの実装と HDAM を適用したシステムの開発が挙げられる．さらに，このシステムを運用することにより，HDAM の運用モデルや端末の信頼モデルを確立する．

謝辞 本研究の一部は，情報通信研究機構 (NICT) の委託研究「ダイナミックネットワーク技術の研究開発」，及び，文部科学省科学研究費補助金若手研究 (20700069) の助成を受けて実施したものである．

参考文献

[1] Atushi Takeda, Kazuo Hashimoto, Gen Kitagata, Salahuddin Muhammad Salim Zabir, Tetsuo Kinoshita, and Norio Shiratori. A new authentication method with distributed hash table for p2p network. *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*, 2008.

[2] Atushi Takeda, Debasish Chakraborty, Gen Kitagata, Kazuo Hashimoto, and Norio Shiratori. A new scalable distributed authentication for p2p network and its performance evaluation. *The 12th WSEAS International Conference on COMPUTERS*, 2008.

[3] R. Housley, W. Polk, W. Ford, and D. Solo. Rfc 3280: Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile, 2002.

[4] Simson Garfinkel. *PGP : Pretty Good Privacy*. Oreilly and Associates Inc., 1994.

[5] Srdjan Capkun, Levente Buttyan, and Jean-Pierre Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2, No.1:52–64, 2003.

[6] Yuko Kitada, Akira Watanabe, Iwao Sasase, and Keisuke Takemori. On demand distributed public key management for wireless ad hoc networks. *Communications, Computers and signal Processing, 2005. PACRIM. 2005 IEEE Pacific Rim Conference on*, pages 454– 457, 2005.

[7] Jeremy Goold and Dr. Mark Clement. Improving routing security using a decentralized public key distribution algorithm. *Internet Monitoring and Protection, 2007. ICIMP 2007. Second International Conference on*, 2007.