匿名性と不正者の特定を両立させるP2P環境用認証方式

平野基孝[†1,†2] 首藤一幸[†3]
田中良夫[†4] 佐藤三久[†1]

我々は，安全で利便性の高いP2P集団通信基盤の確立を目的として，匿名性を保ちつつも，不正を行ったユーザの特定を可能とすることで，不正行為の抑制を促すことのできるP2P環境用認証方式**AUBReX（Authentication method Using Buddy-buddy relationship Represented by Cross certificate）**を提案する．AUBReXでは，2ユーザ間（友人）の信頼関係を，そのユーザ間以外では個人情報の特定ができないように生成されたエンドエンティティ名（SubjectDN内のCommonName）を持つX.509デジタル証明書を相互に発行しあうことで表現する．これを匿名相互証明書と呼び，匿名相互証明書からなる証明書チェインをP2P通信により生成，検査することで，直接の信頼関係を結んでいないユーザ間での，匿名性を確保したうえでのユーザ認証機構を提供する．我々はAUBReXのサンプル実装と，それを利用したAUBReXの認証機構の基本機能の動作検証を行い，良好な結果を得た．これより，提案手法が我々の目的とするP2P集団通信基盤において有効な認証方式たりうると考える．

An Authentication Method That Supports both Anonymity and Malicious User Identification for P2P Environment

Motonori Hirano,[†1,†2] Kazuyuki Shudo,[†3] Yoshio Tanaka[†4]
and Mitsuhisa Sato[†1]

In order to establish a secure, and useful P2P environment, we propose an authentication method called **AUBReX (Authentication method Using Buddy-buddy relationship Represented by Cross certificate)**, which supports both anonymity and malicious user identification, for P2P environment. In the AUBReX, a trusted relationship (fellowship, or buddy-buddy relationship) between two users is represented by issuing X.509 cross certificate each other. The cross certificate has a secure-hashed CommonName as an end entity, that can only be revealed between the users. By collecting such anonymous cross certificates via P2P connection and generating a certificate chain and verifying it, the AUBReX provides an authentication mechanism between users who don't have direct trusted relationship. We have made a sample implementation of the AUBReX and tested basic authentication functionalities with the implementation. With results of the test, we conclude that the AUBReX can be an effective authentication method for our desired P2P environment.

1. はじめに

近年のインターネット環境の発達により，分散コンピューティングは一般のコンピュータユーザにとっても身近なものとなってきている．なかでも，分散コンピューティング環境としてP2Pを利用するアプリケーション，特にファイル共有を目的とするものの普及は瞠目に値する．これら多くのアプリケーションの使用に際し，ユーザは基本的にユーザ登録，アカウント申請のような前準備を必要とせず，「思い立ったらすぐ使える」システムとなっている．しかし，そのファイル共有アプリケーションを通じた情報漏洩，ソフトウェア違法コピー等の問題が頻出していることも事実である．なぜこのような問題が生じるのか．我々は一因として，前述の，多くのP2Pアプリケーションがユーザ登録，アカウント申請を必要としないこと，すなわち認証機構の不備もしくは欠落があげられると考える．

†1 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†2 株式会社SRA
Software Research Associates, Inc.
†3 ウタゴエ株式会社
Utagoe Inc.
†4 産業技術総合研究所グリッド研究センター
Grid Technology Research Center, National Institute of Advanced Industrial Science and Technology

認証機構の不備は，ユーザに不必要な匿名性を与え，「自分が不正を行っても発覚することはない」という間違った考えをユーザにいだかせる．

認証手法として現在広く用いられている PKI ベースの認証方式では，X.509 デジタル証明書を用いる．X.509 デジタル証明書には通常，ユーザの所属組織，氏名，その組織で管理，発行された電子メールアドレス等が証明エンティティとして含まれるので，匿名性は存在しない．

しかし，不正の抑止を目的として，P2P 環境に PKI ベースの認証機構を導入することにも問題がある．X.509 証明書の証明エンティティは，他の情報と合わせて特定の個人を識別することのできる，いわゆる個人情報である．X.509 証明書を使用した相互認証では，X.509 証明書を互いに交換，署名を検査することで認証を行う．基本的に不特定多数のユーザ，計算リソースと通信を行うことが想定される P2P 環境で証明書を交換しあうことは，証明書内の個人情報の不特定多数への流出を意味する．たとえば証明書に直接電子メールアドレスが含まれていた場合，その流出は迷惑メールの送信先として使用されうるし，住所が特定された場合には，架空請求詐欺や振り込め詐欺等にも利用されうる．また，PKI ベースの認証機構もしくはアカウント管理機構は，トップダウン，かつ中央集権・集約的であり，P2P 環境のようなユーザ間での自律的なネットワーク構成を目指すシステムのアカウント管理機構，認証機構として有効に機能するか否かの議論も必要であろう．

我々は本稿で，P2P 環境の良い点，すなわち「思い立ったらすぐ使える」という利便性の高さもあわせ持ち，なんらかの認証機構を導入・改良することで，普段は高い匿名性を保ちながらも，不正を行ったユーザが出た場合には，そのユーザの特定を可能とし，ユーザに「不正を行えば発覚する」ことを自覚させることで不正の抑止力とするような P2P 集団通信基盤の確立を研究目的として，その集団通信基盤内で使用する認証方式 **AUBReX（Authentication method Using Buddy-buddy relationship Represented by Cross certificate）**を提案する．

AUBReX では，現実世界において互いに信頼しあった 2 者間において，その 2 者間でしかお互いの個人情報が特定できないように工夫された X.509 証明書を相互に発行しあう．この証明書を匿名クロス証明書，もしくは匿名相互証明書と呼ぶ．ユーザは自分の望む人数の他ユーザとの間で匿名相互証明書を発行しあってよい．これらより以下のことが可能となる．

- 相互信頼関係にないユーザへの個人情報の流出の抑制
- 現実世界での個人間の信頼関係のつながり方の，クロス認証への直接マッピング
- クロス認証とその信頼モデルの拡張による強固な認証パスの生成および検証機構の提供
- 他人の権利や規則を尊重しないユーザ，そのようなユーザを庇い立てするユーザの特定手段の提供
- トップダウン・中央集権的な機構を必要としない，ユーザコミュニティによる自律的，ボトムアップな認証機構の提供

まず 2 章で，本研究が目的とする分散コンピューティングのための P2P 集団通信基盤を示し，それが持つべき性質をまとめ，その通信基盤内での提案手法 AUBReX の認証機構としての役割を述べる．次に 3 章で，それらの性質を実現するための認証方式である提案手法 AUBReX の基本となる着想に関して説明する．次に 4 章で，認証の基礎となる 2 つのプロトコルに関して説明する．5 章では，提案手法のサンプル実装として，API ライブラリ `libabx` と，それを使用した AUBReX 認証アプリケーションを紹介する．6 章では，5 章であげたサンプル実装を使用した，実際の認証から SSL 通信路の確立までを順を追って説明する．7 章では，サンプル実装を使用した AUBReX の認証機構の基本機能の動作検証について述べる．8 章では，実装上の工夫による本件提案手法の有効性向上の手法に関して述べる．9 章では，本件提案手法の安全性について考察する．10 章では，関連研究について述べる．最後に 11 章でまとめと今後について述べる．

2. 目的とする P2P 集団通信基盤

本研究が目的とする分散コンピューティングのための P2P 集団通信基盤を示す．

任意の計算リソース（CPU，ストレージ，ファイル等）の提供者と，その他の任意の計算リソース使用者が，両者間でなんらかの約束のものとにそのリソースの使用の可否を決定でき，もしその約束を破るような不正者がいた場合には，できる限りコミュニティ内でその不正者の特定を可能とする，安全な通信方式を用いる利便性の高い，数十〜数千人規模の P2P ベースの集団通信基盤．

2.1 P2P 集団通信基盤が持つべき性質

まず我々は，本研究が目的とする P2P 集団通信基

盤が持つべき性質を考察した．以下にそれをまとめる．

(1) **利便性：**
誰もが比較的簡単に，通信基盤のメンバになれること．

(2) **ユーザ認証の必要性：**
通信基盤のメンバになるためには，なんらかのユーザ認証を得ていなければならない．

(3) **匿名性：**
通信基盤のメンバ間は，基本的に匿名であること．

(4) **通信の安全性：**
通信基盤のメンバ間の通信には，暗号化等によりデータ秘匿性が付与できること．

(5) **不正を行ったユーザの特定：**
通信基盤のあるメンバがなんらかの不正を行った場合には，そのメンバが現実社会で意味のある個人情報において特定可能であること．そのようなメンバを庇い立てするメンバについても，同様の措置が行えること．

(6) **自律性・自治性：**
上記を満たすための機構が，なるべく中央集権・集約的でなく，ユーザ認証・管理機構も含めて，ユーザ自身あるいはユーザ同士において行えるものであること．

2.2 想定アプリケーションと本件提案手法の役割

このような P2P 集団通信基盤の提供により，専用の分散環境（クラスタ，グリッド等）を持たない組織・ユーザが，比較的手軽に分散コンピューティングを行うことができると考える．

この P2P 集団通信基盤上において，ファイル交換，データ共有，これらを利用したジョブサブミッションシステム，RPC システム等を，ミドルウェアとして想定している[⋆1]．

実際のアプリケーションとしては，主に RPC システムを使用した大規模パラメタサーチ，最適化問題，データマイニング等を想定している．これらアプリケーションでは多くの場合マスタ・ワーカ型の並列プログラミングによる解法が効果的であり，RPC はマスタ・ワーカ型の並列プログラミングに対して有効に機能することも知られている．マスタ・ワーカ型並列プログラムにおいては，基本的にワーカノードの数が性能を支配する．前述の専用の分散環境を持たない組織・ユーザにおいては，ワーカノードを増やすために P2P 環境を利用することが非常に自然な解決となる．

⋆1 ファイル交換はそれ自体でアプリケーションとなりうる．

P2P 環境においては，グリッド環境等のように参加している組織・ユーザ間に組織的な信頼関係が存在するわけではないので，P2P 環境における RPC システム・アプリケーションは，ワーカが故意に不正確な結果を返す等の不正な動作（サボタージュ）に関して十分考慮する必要がある．サボタージュを行ったワーカノードの特定は，アプリケーションもしくはミドルウェアにおいて IP アドレスレベルで行うことになるが，今日ではインターネットカフェ等の IP アドレスからの個人の特定が困難な接続形態が普及しており，認証がなく不必要に高い匿名性を保ったままの P2P 環境においては，そのワーカノードの使用者の現実社会における特定に至り難い．しかしながら，本稿冒頭で述べたように，不正を行ったユーザを特定するために，個人情報を含みうる証明書等が集団通信基盤内に流通してしまうような手法をとることにも問題がある．

本件提案方式 AUBReX は，普段は高い匿名性を保ちつつも，不正があった場合には不正を行ったユーザの現実社会での特定を可能にする認証方式の提供により，ユーザに「不正を行えば発覚する」ことを自覚させることで，本研究が目的とする P2P 集団通信基盤における不正の抑止力を向上させる．

2.3 認証と権限管理，および権限委譲

提案手法 AUBReX は，目的とする P2P 集団通信基盤に対し，匿名性の保持と不正を行ったユーザの特定という，基本的に相反する機能を両立させたユーザ間相互認証機構を提供する．目的とする P2P 集団通信基盤を利用するアプリケーションのためには，認証機構以外に，誰にどのようにリソースを使用させるか，すなわち権限（Authorization）管理機構が必要であるが，AUBReX は権限管理機構は含まない．

アプリケーションにおける権限管理は，そのアプリケーションごとに大きく異なるが，我々は，P2P 環境でのアプリケーションの権限管理に関し，大きく 2 種類に分類できると考える．1 つは基盤内の全ユーザが基本的に等価なリソースアクセス権限を持つようなアプリケーションで，ファイル共有アプリケーション等がこれにあたる．もう 1 つはこれより少し複雑な，リモートでの任意のプログラム実行を許すようなアプリケーション，たとえばリモートシェル，バッチ型ジョブマネジャ等での権限管理である．本研究が目的とする P2P 集団通信基盤を使用するアプリケーションにおいては，上記の 2 種類のアプリケーションの権限管理に対応でき，かつ，先にあげた匿名性と自律性・自治性を阻害しない権限管理機構の使用が必要である．このような管理機構として，RFC 2693 で規定された

SPKI 権限証明書[1] を使用する権限管理機構，もしくはそれと同等なものを AUBReX とともに使用することを想定している．

また，任意の時点，場所で得られた認証・権限を，異なる場所で使用することを可能にする権限委譲（Credential Delegation）機構に関して，現在，本研究が目的とする P2P 集団通信基盤で本当に必要な機能であるか否かの議論を進めている．権限委譲機構は，実現方式，使用方法を誤ると，認証・権限管理機構以上にセキュリティ脆弱性を孕むからである．

3. AUBReX の基本的着想

3.1 セキュリティ技術

目的とする集団通信基盤において，データ秘匿化通信機構は必須である．これらのために用いるセキュリティ技術としては，SSL と Kerberos が現実的な候補となる．本研究では SSL を用いることにした．以下にその理由を示す．

(1) SSL のユーザ認証は PKI に基づいている．PKI 認証ではユーザ証明書が必ず使用される．ユーザ証明書内には改竄不可能な方式で証明エンティティが保持されるために，不正を行ったユーザの特定のためにその証明エンティティを使用できる．

(2) 認証時において Kerberos のようにつねに認証サーバのような中央集権的リソースを必要としない．

P2P 環境において通信路として SSL を用いるということは，本件提案手法の認証機構は，認証の結果として必ず SSL 相互認証が可能な X.509 証明書を発行するものでなければならないことを意味する．P2P 環境では，通信基盤内の各ユーザ，リソースには，基本的にサーバ，クライアントの区別がないのであるから，一般に Web で用いられる，サーバ側証明書のみを用いた SSL 片側認証でなく，つねにピアどうしが相互認証を行う必要がある．

また，認証の結果として相互認証可能な SSL 通信用証明書を発行するという手法をとることで，1 度認証を得て SSL 証明書を入手すれば，その証明書が有効期限切れ等で失効しない限り，認証機構自体を再度使用する必要がなくなり，アプリケーション実行時の効率が向上する．

3.2 信頼モデル

我々はまず，従来の PKI ベースの認証機構，すなわち，認証の基本となる証明書が中央集権的に CA で発行されるという機構の，P2P 環境における有効性を考察した．P2P 環境では，基本的に不特定多数のユーザが単一の通信基盤に属することになる．このような環境では，単一の CA の CP/CPS をすべてのユーザに納得，了承させることは困難である．複数の CA を使用するとするとしても，それらの CP/CPS の違いにより，任意のユーザにおいて，ある CA は信用してもよいが，ある CA は信用しない，等の個々の信頼ポリシの違いによって，相互運用が困難となる．これらの理由により，P2P 環境においては中央集権的なユーザ管理，認証機構をとることは有効でなく．P2P 環境自身がユーザ間での自律的なネットワーク構成を指向するのであれば，その認証機構もやはり，ユーザ間同士での自律的でボトムアップな認証機構であるべきだと考えた．

さらに我々は，多くのユーザにおいては，分散コンピューティングを行うためのリソースの提供者，自分のリソースの使用に許可を与える相手として，お互いに現実社会においてよく素姓の分かった組織，知人等を第 1 の候補として選択すると考えた．これは，分散コンピューティング環境として確立しているグリッド環境においてもしばしば見受けられる．グリッド環境，特にグリッド環境構築用ミドルウェアとして業界標準となっている Globus Toolkit[2] においては，PKI ベースの認証方式が使用されているが，リソースを使用する側，提供する側の双方が，互いが信頼する任意の CA から発行された証明書を持っているからといって，いきなり見ず知らずの相手のリソースを使用したり，使用させたりすることはあまりない．通常は，現実社会においてお互いに既知である間柄のユーザ・組織の代表者等にまず電子メール，電話等で連絡しあい，お互いの了承のうえでリソースを使用する/される，という手順が踏まれることが多い．

これより，この現実社会での相互信頼関係を，そのまま電子認証に応用することはできないかと考え，ユーザ各個人が自分自身のルート CA を持ち，そのルート CA で信頼関係にあるユーザのルート CA 証明書を署名しあうという，クロス認証を基本とすることにした．クロス認証を基本とすることで，中央集権的 CA は必要とされなくなる．また，クロス認証という，既存の PKI ベースの認証技術を利用することで，まったく新規に考案した認証技術よりは信頼性の点で有利である．

しかしながら，単純なクロス認証には，信頼関係を結ぶ必要のある全ユーザとの間にクロス証明書を発行しあわなければならず，各ユーザの負荷が著しく高くなるという欠点がある．

そこで我々は，クロス認証の信頼モデルを拡張し，以下の条件での認証を追加することにした．

(1) もしユーザ A とユーザ B が信頼関係にあり，かつユーザ B とユーザ C が信頼関係にあるのであれば，ユーザ A とユーザ C は**信頼関係チェイン**でつながっているといい，お互いに信頼しあってもよい．換言すれば「友人の友人は友人であると見なそう」ということである．これを，**信頼関係チェインによる認証**と呼ぶ．

(2) 信頼関係チェインは任意の長さに延びてよい．前述の例であれば，さらにユーザ C とユーザ D が信頼関係にあるのであれば，ユーザ A とユーザ D は信頼関係チェインでつながっているといえる．

(3) 信頼関係チェインでつながった任意の 2 ユーザ間の距離を**ホップ**と呼ぶ．直接信頼関係にある，すなわち信頼関係チェイン上で隣どうしにあるユーザ間のホップを 1 とし，途中のユーザが増えるごとに 1 ずつ増加させる．ユーザ A からユーザ B までの信頼関係チェインの長さが n であれば，A から B までのホップは $n-1$ となる．

(4) 信頼関係チェインを利用して認証を行う場合，認証者は被認証者までのホップ数に制限を設ける，認証者から被認証者までのチェインに特定のユーザが現れているといっさい信頼しない，等の，認証側の任意の条件で認証するかしないかを制御できる．これにより，通常の PKI ベースの認証機構においては CRL が CA，もしくは CA の信頼した CRL 発行者により中央集権的に管理されるが，これをユーザ個々の管理に任せることになる．

信頼関係チェインはそのまま相互認証用の X.509 証明書チェインの概念の拡張となる．

3.3 匿名性の実現

しかしながら，通常の X.509 証明書は個人情報を含む．これでは匿名性が実現できない．

かといって，単純に証明書内の証明エンティティとして現実社会において個人を特定できない情報，たとえば任意のハンドルネーム，非常に簡単に入手できる無料電子メールサービスアカウント等を用いるようにするだけでは，不正を行ったユーザの現実社会における特定は困難である．不正を行ったユーザの特定は，通信時に使用された IP アドレスをもとに，その IP アドレスを付与した組織，ISP 等の協力によっても行えるであろう．しかし，我々はそれだけでは不十分であると考えた．理由は，フリーな公衆 WAN 回線，いわゆるネットカフェ，その他 IP アドレスから身元を特定するのが煩雑なインターネット接続形態が普及しており，不正を行うつもりのあるユーザがこれらの接続形態を使用するのが非常に簡単だからである．さらに，不正を行われた被害者，もしくは集団通信基盤自身に，不正を行ったユーザの身元を暴くための機構が備わっていれば，第三者の協力を必要としなくても，いわば自治的に事態の解決が行えるので，不正者の特定が迅速に行えると考えた．

必要となるのは，PKI ベースの X.509 証明書による認証を用いるが，認証時，通信路の確立時を通して，ユーザの個人情報がネットワークに直接露出せず，なおかつ問題があったときには，問題を起こしたユーザが，できる限りユーザコミュニティのみによる活動で特定できるような手法である．したがって，個人を特定する情報は，少なくとも自分自身以外の誰か 1 人には公開されていなければならないことになる．

前節で述べたように，本提案手法は，現実世界での個人間の信頼関係をそのまま認証モデルとして用いることを基本とする．現実世界において，ある個人を特定するための情報を持っている人物は，その個人本人と，その個人と互いに信頼関係にある人間（親，兄弟，友人等）であろう．

本手法においても，信頼関係にあるユーザ間では，現実世界において互いを特定するための情報，いわゆる個人情報が開示されているものとした．このようなモデルを導入することで，以下に述べる手法により，匿名性と不正を行ったユーザの特定という相反する目的を達成することができる．

(1) 各ユーザは，自分自身の正しい個人情報からなる SubjectDN を含んだ自己署名 CA と証明書を作成する．これを **OURCA（Original User Root CA），OURCA 証明書**と呼ぶ．

(2) 次にユーザは，OURCA 証明書ファイルの内容を SHA1 等の単方向ハッシュ関数で処理した値等を基にしてユニークな識別子を生成する．これを **AnonCN（Anonymous Common-Name）**と呼ぶ．十分に強力な単方向ハッシュ関数を用いることで，AnonCN から OURCA 証明書の内容は推測不可能となる．

(3) 次にユーザは，AnonCN が SubjectDN の CommonName であり，かつ OURCA 証明書の公開鍵がそのまま公開鍵となるような自己署名 CA と証明書を，OURCA 証明書の元となった秘密鍵で署名することで作成する．これを

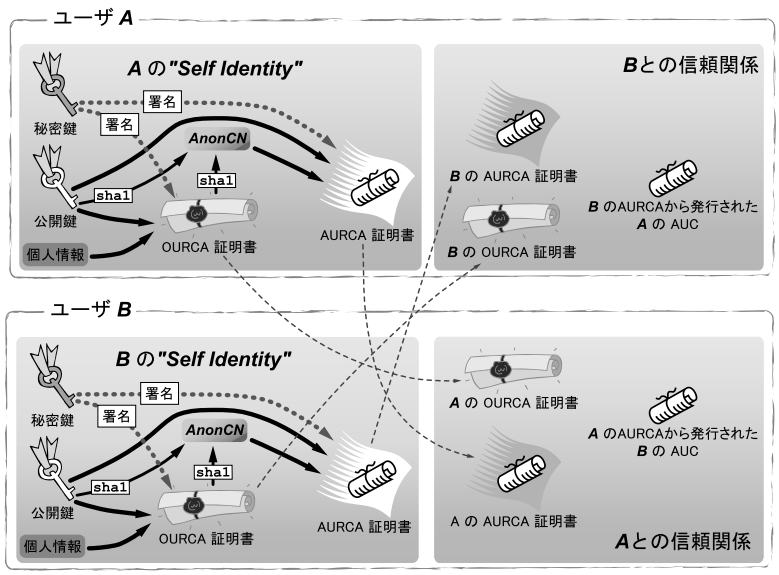

図 1 AUBReX における信頼関係の形成
Fig. 1 Establishment of a mutual trust by AUBReX.

AURCA（Anonymous User Root CA），AURCA 証明書と呼ぶ．

(4) 任意の 2 ユーザ間の信頼関係は，OURCA 証明書を互いに交換しあい，さらに AURCA を用いて互いの AURCA 証明書を署名・発行しあうことで確立する．通常のクロス証明書の発行，クロス認証方式との違いは，発行・使用される証明書の SubjectDN の CommonName が，クロス証明書を発行しあった間柄，すなわち互いに信頼関係にある間柄でなければ完全に非透過である点にある．このことより，本研究ではこの AnonCN によって匿名化されたクロス証明書を，特別に**匿名クロス証明書**，もしくは**匿名相互証明書**と呼ぶことにした．また，互いに発行しあった匿名相互証明書を自己署名の AURCA，AURCA 証明書と区別するために，**AUC（Anonymous User Certificate）**と呼ぶことにする．通常のクロス証明書同様に AUC には方向があり，A，B 間で匿名相互証明書を発行したならば，A が発行した B の AUC，B が発行した A の AUC の 2 種類が必ず存在する．ここで重要なのは，信頼関係の確立前に，互いの OURCA 証明書に記載された個人情報が正しいものであること，さらにその OURCA 証明書から相手の AnonCN が正しく生成できることを，互いに十分に確認することである．

(5) 集団通信基盤およびその形成時，さらにユーザ認証時において，全ユーザは AnonCN により識別され，信頼関係チェインは AnonCN のリストで表現される．

(6) AUBReX での認証とは，信頼関係チェインを，チェインに含まれる各ユーザに関し，そのユーザの AUC がそのユーザと信頼関係にあるユーザの AURCA 証明書で署名発行されているかを検証することで行う（4.3 節で後述）．認証時において，生の個人情報を含む OURCA 証明書が使用されることはない．

(7) もし，ある AnonCN X’ で匿名化されたユーザ X が悪事を行った場合，X’ の元となった OURCA 証明書を有する，X と信頼関係にあるすべてのユーザが，X’ から X を特定可能である．

(8) 前述において，もし X と信頼関係にあるすべてのユーザが X の個人情報の提供を拒んだ場合，信頼関係チェインをたどることで，X を庇ったユーザ全員の個人情報の入手が可能であり，不正を行ったユーザに荷担するものにもペナルティを課すことが可能となる．

図 1 に，AUBReX での 2 ユーザ間の信頼関係の形成方法をまとめる．

前述の (6)〜(8) で述べたように，正しい個人情報を含んだ OURCA 証明書は，AUBReX での認証にはいっさい使用されない．OURCA 証明書は，不正が報告された場合に，その不正者の身元を公開するため

のみに存在する．OURCA が自己署名デジタル証明書であるため，1 度生成された後は，それに含まれる個人情報の改編が不可能である．信頼しあった間柄で OURCA 証明書を持ちあうことは，いわば「人質の交換」である．どちらか（もしくは互いの責任において信頼関係を結んだお互い以外の他人）が悪事を犯したときには，その責任が最終的には必ず悪事を犯した本人，もしくは犯した本人を庇う者に向けられるようにすることで，不正者の特定とともに，不正な行動の抑止も目的とする．

AnonCN は次の式で生成される．

```
base64encode(sha1(sha1(oCert) ||
                  sha1(oPubKey)))
    ... (1)
```

ここに，oPubKey は OURCA 証明書の公開鍵，oCert は PEM 形式で表現された OURCA 証明書そのものである．sha1(oPubKey) は鍵指紋（Key finger print）とも呼ばれる．

4. 認証プロトコル

信頼関係チェイン，匿名相互証明書を利用して任意の 2 ユーザ間の相互認証を行うために，AUBReX では次の 2 種類の認証プロトコルを使用する．

- **AIVP（Anonymous Identity Verification Protocol）**：
  任意のユーザ a の AnonCN a' の整合性，および AnonCN a' がユーザ a によって使用されているかを検査するためのプロトコル．
- **BRVP（Buddy-buddy Relationship Verification Protocol）**：
  任意のユーザが，AnonCN a' で示されるユーザと AnonCN b' で示されるユーザとが信頼関係にあるかどうかを検査するためのプロトコル．

AUBReX による認証とは，これら 2 つのプロトコルを使用して，任意の 2 ユーザ間の信頼関係チェインが署名関係によってつながるか否かを検証する処理をさす．3.1 節で述べたように，この検証が成功した場合，認証の結果として，その 2 ユーザ間でお互いに，自分の AURCA を発行者とし，相手の AnonCN を CommonName とする SSL 相互認証通信用証明書を発行しあう．認証処理自体を行うアプリケーションと，その認証結果を使用するアプリケーションを独立させるのであれば，認証処理自体を行うアプリケーションは，AUBReX の実装系で提供されなければならない．このアプリケーションを，**AUBReX 認証アプリケーション**と呼ぶ．これについては 5 章で説明する．

4.1 語 定 義

まず，以降の議論での簡便のために，以下の定義を行う．

- anoncn(x)：
  ユーザ x の AnonCN の値．
- x'：
  anoncn(x) によって識別されるユーザ x．
- sha1(x)：
  x の単方向ハッシュアルゴリズム SHA1 でのハッシュ値．
- orc(x)：
  ユーザ x の OURCA 証明書．
- arc(x)：
  ユーザ x の AURCA 証明書．
- auc(x, y)：
  ユーザ y によって署名・発行されたユーザ x の AUC．
- aivp(a, b')：
  a から b' への AIVP 発行．
- brvp(a, b', c')：
  a から b' への，b' と c' が 1 ホップの信頼関係にあるか否かの検査のための BRVP 発行．

4.2 AIVP

aivp(a, b') の手順を説明する．

(1) a，b' 間で P2P 通信路を確立する．
(2) b' が a に arc(b') と sha1(orc(b')) を送信する．
(3) a は arc(b') と sha1(orc(b')) を用い，検査用の AnonCN として bV' を計算する．AnonCN の整合性とは，(1) の P2P 通信路確立時に a が b' であると仮定した AnonCN と，b' から入手した arc(b') と sha1(orc(b')) とを利用して計算された AnonCN が一致するか否かを指す．a は b' と bV' を比較することで AnonCN の整合性を検査する．
(4) 次に a は arc(b') 内の公開鍵を用いて b' が arc(b') の生成時に使用した秘密鍵の正当な使用者であることを，適切な乱数等を用いた challenge-response により確認する．

sha1(orc(b')) の値が本当に orc(b) から計算されているか否かは a からは判断できない．したがって，たとえば 20 バイトの乱数等を sha1(orc(b')) として

使用することで任意の検査者からの AIVP を成功させる攻撃が考えられるが，ユーザが 3.3 節 (4) で述べた手法を正しく使用するのであれば，ユーザはつねに，自分自身の個人情報より正しく生成された OURCA 証明書を所有することになり，このような攻撃は成立しない．この理由に関しては 9 章で詳説する．

4.3 BRVP

`brvp`(a, b', c') の手順を説明する．

(1) a，b' 間で P2P 通信路を確立する．
(2) `aivp`(a, b') を行う．
(3) b' は a に `sha1`(`orc`(c'))，`arc`(c')，`auc`(b', c') を送信する．b' が c' と信頼関係にあるなら，これらの値はすべて b' にとって既知のはずである．
(4) a は `arc`(c') と `sha1`(`orc`(c')) より，`anoncn`(c) を再計算し，元々の `anoncn`(c) の値，`arc`(c') 内の AnonCN の値と比較する．
(5) さらに a は，`auc`(b', c') 内の公開鍵が，`aivp`(a, b') 時に得られている `arc`(b') の公開鍵と同一であるか検査する．
(6) 最後に a は，`auc`(b', c') が c' の AURCA によって発行された匿名相互証明書であることを，`auc`(b', c') の発行が `arc`(c') を使用して行われているか否かを検査することで確認する．

5. サンプル実装

我々は，提案手法 AUBReX の認証機構としての基本的な機能の検証を目的として，サンプル実装を行った．

開発は Windows Cygwin 環境で行われ，Windows Cygwin，Linux，NetBSD での動作を確認している．

サンプル実装は API ライブラリ `libabx`，AUBReX 認証アプリケーション，SSL テストプログラムからなる．

5.1 API ライブラリ `libabx`

`libabx` は C 言語で記述された約 70 個の関数を含むランタイムライブラリである．AUBReX 認証アプリケーション用 API，AUBReX 認証で得られた SSL 相互認証用証明書を用いて SSL 通信を行うアプリケーションが使用する SSL 通信用 API，AUBReX 認証プロトコル（AVIP，BRBP）用 API，信頼関係チェイン検証用 API 等が含まれる．`libabx` は X.509 証明書処理，RSA 鍵処理，SSL 通信のために，OpenSSL ライブラリを使用している．

5.2 AUBReX 認証アプリケーション

AUBReX 認証アプリケーションは，AUBReX での認証処理自体を行うアプリケーション群である．シェルスクリプト，C 言語を用いて記述されている．C 言語で記述されたものは，`libabx` 内の API を使用している．本サンプル実装では以下のものを作成した．

- `abxAnonCN`:
  指定された OURCA 証明書ファイルから，その OURCA 証明書の発行者の AnonCN を出力するコマンド．
- `abxRealID`:
  自分の持つ OURCA 証明書のうち，指定された AnonCN の元となった OURCA 証明書を検索し，その SubjectDN を出力するコマンド．
- `abxUserInit`:
  OURCA，AURCA 等，AUBReX 認証におけるユーザ基本情報の生成コマンド．
- `abxMakeBuddy`:
  1 ホップ信頼関係構築コマンド．
- `abxDirServ`:
  各ユーザの 1 ホップの信頼関係，ネットワーク接続状態等を保持するディレクトリサービス．
- `abxAuthd`:
  AIVP，BRVP 受理，SSL 相互認証用証明書の自動発行を行うユーザレベルデーモン．
- `abxDumpOnlineUser`:
  `abxDirServ` に保持されている情報のローカルデータベースへのダンプコマンド．
- `abxFindChain`:
  ローカルデータベースをスキャンし，指定されたユーザとの間に信頼関係チェインが存在するか否かを検査するコマンド．
- `abxVerifyChain`:
  `abxFindChain` で発見された信頼関係チェインの検証コマンド．検証が成功すれば，SSL 相互認証用証明書が出力される．

5.3 SSL テストプログラム

SSL テストプログラム `sslTest` は，前述の `abxVerifyChain` によって生成された SSL 相互認証用証明書を用いて，アプリケーションが最終的に SSL 通信が行えるか否かを検証するためのプログラムである．

4 章で述べたように，AUBReX による認証で得られた SSL 証明書は，自分の AnonCN を Common-

Name として相手の AURCA から発行された証明書である．このような証明書を用いて SSL 通信を行うために，SSL 通信のための前段階として，先にピアの AnonCN を入手し，そのピアから発行された SSL 証明書をユーザ証明書，自分の AURCA 証明書を含む証明書ストア（ディレクトリ）をルート証明書ストアとして使用するよう，SSL API 層（本件サンプル実装では OpenSSL）を設定する必要がある．`libabx` はこれらの処理を行い，生成された SSL コンテキストで SSL 通信を行うための API を提供しており，`sslTest` はそれら API を使用して C 言語で記述されている．

6. 実際の通信路確立までの手順

本章では 5 章であげた AUBReX 認証アプリケーションを用い，AUBReX の認証を介して，実際にピア間で SSL 通信が行えるようになるまでのシナリオを示すとともに，実際の認証がどのように行われるかの手順について説明する．

6.1 ユーザ基本情報の生成

まず，通信基盤に参加する前に，各ユーザが `abxUserInit` を実行する．`abxUserInit` はそのユーザの OURCA，AURCA を生成する．次に `abxUserInit` は，OURCA，AURCA を，AURCA 生成の元となった証明書署名要求（CSR）とともに，ABX ファイルと呼ばれる単一のファイルにまとめる．

6.2 1 ホップ信頼関係の確立

通信基盤のメンバになる可能性のあるユーザのうち，物理的に近くに存在し，直接会って 1 ホップの信頼関係を確立できる，たとえば同一部門，部署，研究グループのユーザ間で，`abxUserInit` が生成した ABX ファイルを直接会って人手で交換しあい，お互いが相手の ABX ファイルを `abxMakeBuddy` で処理することで，AUBReX での直接の信頼関係を構築する．このとき，`abxMakeBuddy` は，相手の OURCA 内の個人情報を出力させ，ユーザに確認を求めることで，3.3 節 (4) で述べた手順を確実に守るように促す．ユーザが相手の個人情報が正しいと判断して処理を続行すると，`abxMakeBuddy` は自分の AURCA で，相手の ABX ファイル内の CSR を署名して，AUC を生成する．この AUC を相手と互いに交換することで，1 ホップの信頼関係が成立する．

グループ内の全員が，その他のグループ内全員と直接の信頼関係を結ぶ必要はないが，全員と結ぶことは，AUBReX での認証を完全にクロス認証と等価とし，高い認証強度を与えることになる．

次に，地理的に離れた場所にあるグループの代表者誰か 1 人と，こちらの代表者 1 人が，直接会って ABX ファイルを交換しあい[⋆1]，やはり同様に `abxMakeBuddy` を用い，3.3 節 (4) の手順を守り，両者の間に AUBReX での直接信頼関係を結ぶ．生成された AUC の交換は，つねに電子メール等を介して行ってよい．

メンバとして通信基盤に参加する必要のあるグループごとに，グループの代表者各個が相互に信頼関係を結ぶことで，その代表者を介した信頼関係チェインを全グループ内の各ユーザ間に張ることが可能となる．重要なのは，最低各グループの代表者 1 人のみが，各グループの代表者 1 人ずつと直接の信頼関係を結べばよいという点であり，従来のクロス認証のようなフルメッシュのクロス証明書発行に比べて負荷が低い．

任意の 2 者間の信頼関係チェインは，短いほど信頼度が高い．したがって，全体としてなるべく信頼関係チェインが短くなるような信頼関係トポロジを構成することが望ましい．

2 章の冒頭で述べたように，本研究が最終的に目的としている P2P 集団通信基盤の規模は最大数千人である．ここでその規模の実現可能性に関して考察する．

本件提案手法においては，信頼関係チェインホップ数と認証強度は反比例の関係にある．しかし，認証のスケーラビリティは，信頼関係チェイン最大ホップ数の制限が緩い（許可される最大ホップ数が大きい）ほど良い．

信頼関係チェイン最大ホップ数の最小値は現実的に考えると 3 である．また，集団通信基盤内における信頼関係チェインホップ数の平均を小さく保つことが，基盤全体としての認証強度を高く保つことになるので，基盤を構成するメンバの物理的・地理的制約も考慮すると，本節で示したような，グループ単位で代表者を選出し，各グループの代表者同士が信頼関係を確立するような方法をとることが有効である．

信頼関係チェイン最大ホップ数がつねに 3 であるような数千人規模の集団通信基盤を構築するには，1 グループ数十人，数十グループが存在するとして，各グループの代表者が 1 度に一堂に会してフルメッシュで信頼関係を構築すれば可能であるが，この方法では各グループの代表者の作業負荷が大きく，かつ，新たなグループの追加を行うたびに全グループの代表者が再度一堂に会する必要があり，実現性は低い．

⋆1 もしくは，運転免許証等フォト ID を含んだ公的機関発行の証明書のコピーをを添付したうえで，非常に信頼性の高い通信手段を用いて ABX ファイルを転送することで代用してもよいが，なるべく避けるべきである．

もし信頼関係チェイン最大ホップ数として一時的に5を許し，最終的には4以下を目指すという段階的な構築方法をとるなら，次のような方法が可能である．

(1) まず構築開始時に会合可能なグループの代表者のみでフルメッシュの信頼関係を構築する．以降これら代表者を OR（Original Representative）とする．

(2) 後日，(1) に参加できなかったグループの代表者と，OR 1人以上が会合し，フルメッシュの信頼関係を構築する．この行為は各 OR と新たに参加しようとしているグループ群で可能であるから，基盤内の信頼関係チェイン最大ホップ数は5となる．

(3) 適当な時期に，(2) で追加されたグループの代表者のみが全員会合し，フルメッシュの信頼関係を構築することで，(2) の時点では5であった基盤内の信頼関係チェイン最大ホップ数を4にできる．

(4) グループの代表者全員が一堂に会する機会があれば，そこで (2) で追加されたグループの代表者と OR がフルメッシュの信頼関係を構築できれば，最良の場合，基盤内の信頼関係チェイン最大ホップ数を3にできる．

このような方法を用いれば，信頼関係チェイン最大ホップ数が低下するまでにかかる時間が増加するが，その間は基盤内各ユーザの信頼ポリシにより，ホップ数が5以上の信頼関係チェインによる認証は許可せず基盤に属するユーザ全体とは通信しない，信頼度が低くてもとにかくノードが必要であるからホップ数制限は5以下として基盤全体を使用する等の，基盤内の各ユーザが自身の要求に応じた運用を行えばよく，1グループ数十人，数十グループが存在する，最大おおよそ数千人規模かつ信頼関係チェイン最大ホップ数4以下の集団通信基盤の確立・維持において，本件提案手法は有効に機能する．

6.3 サインオン

あるユーザが通信基盤に参加したい場合，そのユーザは `abxAuthd` を起動することでサインオンを行う．`abxAuthd` は，その他のユーザからの AIVP，BRVP 受理，SSL 相互認証用証明書の自動発行を行うために，ユーザレベルデーモンとして常駐実行される．

AUBReX は集団通信基盤に参加可能なユーザがオンラインであるか否かを管理するためにディレクトリサービス `abxDirServ` を使用する．ディレクトリサービス自身は AUBReX の認証機構のためにユーザのオンライン状態を管理する．任意のアプリケーションにおいて使用可能なピアのリストを取得するために使用することも可能であろうが，そのアプリケーション固有に必要とされる情報（たとえばファイル共有アプリケーション等における各ユーザの提供ファイル一覧，ファイル転送サービスを受け付けるポート番号等）を持たないため，アプリケーションにおける使用可能ピアの管理は，アプリケーションごとに別個の機構で用意されるものとする．`abxDirServ` は，どこか適切な，ネットワークアクセス可能なホストにて事前に起動されていればよい．`abxDirServ` が稼働しているホスト名（IP アドレス），ポート番号も，すべてのユーザにとって既知であるとする．`abxDirServ` には，ユーザのサインオン時に，`abxAuthd` を介して以下のエントリ情報が登録される．

- AnonCN：
  そのユーザの識別子．
- **コンタクトポイント**：
  そのユーザの `abxAuthd` が使用している IP アドレスとポート番号で構成され，AIVP，BRVP，SSL 証明書発行のための P2P 通信に利用される．
- **信頼関係リスト**：
  そのユーザと1ホップの信頼関係にある他ユーザの AnonCN のリスト

本件サンプル実装では，上記エントリ情報以外に，主に機能検証の簡便のために，AnonCN へのエイリアスとして，任意のニックネームをサインオン時に登録できるようにした．これは AnonCN が 27 文字の base64 文字列であり，人間の判読には向かないためである．実運用を意識した実装においても，同様の機能が必要とされうるが，ユーザが固定的に同じニックネームを使用することで，そのユーザの集団通信基盤内での振舞い（頻繁に通信を行っているピアは誰か，等）の解析とあわせて個人が特定される可能性，ニックネームによるフィッシング詐欺脆弱性等を考慮すると，ニックネームの安易な使用は推奨できない．したがって，実運用を意識した実装においては，なるべくユーザに直接 AnonCN をキー入力させない，等のユーザ・インタフェース上の工夫を行うことが望まれる．

もしユーザの OURCA 証明書の元となった秘密鍵がパスフレーズで暗号化されているのであれば，`abxAuthd` の起動時にパスフレーズの入力が促される．

ディレクトリサービスが単一のサーバによって実装された場合，AUBReX 自身はいわゆるハイブリッド P2P を用いる認証方式となり，分散ハッシュのような手法で実装された場合にはピュア P2P を用いる認証方

式となる．本実装例の abxDirServ は，単一のサーバによって実装されているので，本実装例はハイブリッド P2P を用いる認証方式となる．

また，コンタクトポイントの IP アドレスはグローバルアドレスでなければならない．このため，ファイヤウォール/NAT ボックスの内側にいるユーザをサポートするためには，AUBReX の実装において，使用する通信層において，何らかのプロキシ機構を用意する，UPnP を用いたポート開放等を利用する等の手法が必要となるであろう．

6.4 ピアリストの取得とピアの決定

他のユーザと通信をする場合，そのユーザとの間で AUBReX の認証を介して SSL 相互認証証明書を取得しなければならない．まず abxDirServ から現在サインオンしているピアの一覧を得る．これをピアリストと呼ぶ．ピアリストはサインオン時に各ユーザが登録したエントリ情報の一覧である．ピアリストの取得のため，ユーザは abxDumpOnlineUser を実行する．ピアリストは abxDumpOnlineUser を実行したマシン上に，ローカルデータベースとして格納される．

ピアリストの中から適切な通信相手を決定したら，そのユーザと信頼関係チェインがつながるか否かを確認する．この操作のために，ユーザは abxFindChain を使用する．abxFindChain は abxDumpOnlineUser が生成したローカルデータベース内のピアリストをスキャンし，ピアリストのエントリに含まれる信頼関係リストを全探索，スパニングツリーを生成することで，その相手との間に信頼関係チェインが存在するか否かを検査する．信頼関係チェインが存在したら，それをチェインファイルとして出力する．選択した相手と信頼関係チェインがつながらない場合，その相手とは AUBReX による認証を経たうえでの通信路の確立は不可能であるから，適宜その他の通信相手を選択しなおし，信頼関係チェインがつながる相手を見つける必要がある．

また，信頼関係チェインの長さに制限を設ける，特定の AnonCN が信頼関係チェイン内に存在したら認証しない，等の仕組みも，abxFindChain にコマンドラインオプションを設けることで実現できる．

6.5 認証の開始

前述までの操作で，ユーザ a' が相手としてユーザ b' を選択し，a' から b' までの信頼関係チェインもつながっているとする．AUBReX による a'，b' の相互認証の手順を示す．これは，abxFindChain が出力したチェインファイルを引数として，abxVerifyChain を実行することで行われる．

まず以下の定義を行う．

- tchain(a', b')：
  a' から b' までの信頼関係チェイン．
- len(tchain(a', b'))：
  tchain(a', b') の長さ．a' から b' までの信頼関係チェインに含まれるユーザの総数（a'，b' を含む）であり，a' から b' までのホップ数 + 1.
- sc(a', b')：
  arc(b') を使用して発行された，a' の SSL 相互認証用証明書．

(1) a'，b' が 1 ホップの信頼関係にあれば，互いが発行しあった AUC があるはずなので，それをそのまま SSL 相互認証に用いればよい．

(2) a'，b' とも，自分自身を s'，相手を p' と記述することにする．

(3) a'，b' が 2 ホップ以上の信頼関係であれば，a' が b' のコンタクトポイントに接続し，互いに aivp(s', p') を行う．

(4) 次に a' は b' に tchain(a', b') を送信する．b' は受け取った tchain(a', b') に対して，b' 自身の信頼ポリシを適用して，信頼関係チェイン tchain(a', b') による認証を拒否することもできる．

(5) 次に，双方とも，i（$0 < i < n - 1$，$n =$ len(tchain(p', s'))）をループカウンタ（初期値 1）とし，i を 1 増やしながら，tchain(p', s') 内に存在する i 番目のユーザ（0 番目が最初のユーザ）x'_i について，s' から x'_i のコンタクトポイント，すなわち x'_i の abxAuthd に接続し，brvp(s', x'_i, x'_{i-1}) を行う．この操作により，tchain(p', s') を s' から p' に向かう方向で検査したことになる．

(6) 次に，双方とも，(5) の操作とは検査方向を逆にして，i（$0 <= i < n - 2$，$n =$ len(tchain(p', s'))）をループカウンタ（初期値 0）とし，i を 1 増やしながら，tchain(p', s') 内に存在する i 番目のユーザ（(5) 同様に 0 番目が最初のユーザ）x'_i について，s' から x'_i の abxAuthd に接続し，brvp(s', x'_i, x'_{i+1}) を行う．この操作により，tchain(p', s') を p' から s' に向かう方向で検査したことになる．

(7) (5), (6) において，すべての BRVP が成功したならば，AUBReX による a'，b' の間の相互認証が成功したことになる．

(8) a'，b' 間の AUBReX による相互認証が成功

したなら，s' は p' のコンタクトポイント（p' の abxAuthd）に，arc(p') の使用による s' の SSL 相互認証用証明書 sc(s', p') の発行を依頼し，入手する．このとき，お互いの abxAuthd は tchain(s', p') を保存することで，もし p' が s' に悪意ある行動をとった場合に，信頼関係チェインをたどることで s が p を特定可能にしておく．

a'，b' の認証においては，どちらかが認証要求元として両者間の信頼関係チェインを使って abxVerifyChain を実行すれば，両者で SSL 通信用証明書が作成される．

6.6 SSL 通信路の確立

4 章で述べたように，AUBReX による認証は，abxVerifyChain の実行によって，ピアどうしとなるユーザ双方が，BRVP によって，信頼関係チェインを，相手から自分に向かう方向で，お互いに検証した時点で終了しており，お互いの間での SSL 通信用証明書も AUBReX の認証正常終了時に生成されている．したがって，1 度 AUBReX で相互認証されたユーザ同士では再び AUBReX による認証を行う必要はなく，すでに生成済みの SSL 証明書を用いて，そのまま SSL 相互認証および通信を行えばよい．ただし，生成された証明書チェイン内の証明書の有効期限が切れる等の SSL 自身の認証に問題が生じている場合には，再度 AUBReX による認証を行って，新たにそのピアとの間で SSL 通信用証明書を発行しあう必要がある．

5.3 節で示したように，AUBReX による認証で得られた SSL 通信用証明書を使用して SSL 通信を行う場合，アプリケーションプログラムは libabx 内の当該 API を使用すればよい．

6.7 不正を行ったユーザの特定

ここで，a' と b' の間の通信を介して，b' が a' に対して不正を行ったとする．SSL 通信の開始時に a' は b' を AnonCN で識別できており，tchain(a', b') は a'，b' 間での認証時に abxAuthd が記録している．

a' が b' の不正を告発するにあたっては，b' が a' に不正を働いたという証拠が必要である．証拠の形式はアプリケーションごとに異なるが，基本的にログファイル，通信データダンプファイル等を想定している．

また，不正を行った者を特定するという行為自体が，自分自身とまったく面識のないユーザへの/からの告発を援助すべきか，自分自身と直接信頼関係にあるユーザへの告発をどう受け止めるか等，ユーザコミュニティ内において非常にデリケートな問題を提起することになる．

したがって，不正の告発，不正を行ったユーザの特定に関し，コミュニティとしてある程度の運用規程を制定しユーザに遵守させるとともに，不正の証拠をどのように評価するか，不正者特定を要求してきたユーザを信じるか否か，そのユーザと自分自身のホップ数等，ユーザ個々の不正者特定要求ポリシを適用できるようにもしておくものとする．

このように不正者特定要求の処理にはコミュニティ，ユーザ個人の判断が多く関わるため，プログラム等による処理の自動化を行うことは想定していない．

以下に，不正を行ったユーザ b' の現実社会での識別子，すなわち b の特定方法を説明する．

(1) i ($0 <= i < n-1$，$n =$ len(tchain(a', b'))) をループカウンタとし，tchain(a', b') 内の i 番目のユーザ x'_i において次の処理を行う．
  (a) x'_i は，x'_{i+1} で示される AnonCN が x'_i の持つ OURCA 証明書のうち，誰のものであるかを abxRealID で調べ，x_{i+1} を得る．この時点で，x'_i は x_{i+1} への連絡先として，電子メールアドレス等を得ることができる．
  (b) もし i が $n-2$ であれば，x_{i+1} は b であるから，ループを抜ける．
  (c) x'_i は運用規程と自己の不正者特定要求ポリシに照らし合わせ，要求が妥当であると判断したのであれば，「b' が a' に不正を働いたので b の特定に協力してほしい」の旨を，b が特定できた場合の x'_i への連絡先，tchain(a', b')，不正の証拠とともに x_{i+1} に送付する．x'_i と x_{i+1} は必ず信頼関係にあるので，x'_i の電話番号，電子メールアドレス等を連絡先として x_{i+1} に伝えることに問題はない．
  (d) i を 1 増やし，(a) に戻る．

(2) (1) のループ処理が終了した時点で，b の特定は x'_{n-2} が行っており，x'_{n-2} から a' までの，b の個人情報の返信経路もつながっている．この返信経路を使用して，x'_{n-2} から a' まで b の個人情報を返信する．x'_{n-2} が b の個人情報を返信するか，b の個人情報は知っているが返信しない等，自己の不正者特定要求ポリシに従って行動する．

もしここで，任意の x'_i が返信をしてこなかった場合，x'_{i-1} が a' に，x'_i の現実社会での識別子，すなわち x_i の個人情報を，不正を行ったユーザを庇い立てしているユーザとして返信

する，再度 x'_i に返信を要求する，x'_i が返信をしてこなかった旨を返信する等，これも運用規程と自己の不正者特定要求ポリシに基づいて行動する．

7. サンプル実装による認証機構基本機能動作検証

我々は提案手法である AUBReX の認証機構の基本機能がうまく動作し，認証の結果として SSL 通信用証明書を生成し，最終的にそれを用いたピア間での SSL 相互認証・通信が行えるかどうか，5 章で示したサンプル実装を用いて検証した．

AUBReX の認証機構の基本機能とは，以下にあげるものである．

- 1 ホップの信頼関係の確立．
- 信頼関係チェインを介した 2 ユーザ間の認証と，その結果としての SSL 通信用証明書の生成．
- 生成された証明書による SSL 通信の確立．
- 不正を行ったユーザの特定．

参加するユーザとして 5 人を想定し，**図 2** に示すような信頼関係トポロジを構成することにした．5 人は Group A と Group B に分かれて存在し，Group A には A0 と A1，Gropu B には B0，B1，B2 が所属していることを想定する．

検証に使用した環境ではファイルシステムが共有されており，1 ホップの信頼関係確立のための ABX ファイル，AUC の交換は，そのファイルシステムを介して行った．本来なら USB メモリのようなリムーバブルメディアを介して行うべきではあるが，基本機能の検証目的であれば，影響はないと考える．

検証は以下の手順で行った．

(1) まず 5 人全員で `abxUserInit` を行い，OURCA，AURCA，ABX ファイルを生成した．

(2) 次に，A0 - A1，B0 - B1，B0 - B2 の 1 ホップの信頼関係の確立を `abxMakeBuddy` で行い，それぞれが AUC を交換した．この時点で，A0 - A1，B0 - B1，B0 - B2 間それぞれで，`sslTest` によって SSL 通信が可能であることを確認した．

(3) この時点で `abxDirServ` を立ち上げ，5 人全員で `abxAuthd` を起動した．このとき，検証の簡便のために，サインオン時のニックネームとして，nA0，nB0 のような，ユーザ名の前に n を付加したものを用いた．`abxDumpOnlineUser`，`abxFindChain` を使用して，Group A のユーザと Group B のユーザ間では信頼関係チェインがまったく存在しないことが確認できた．さらに `abxFindChain` を使用して nB1 - nB2 間には信頼関係チェインが存在することを確認し，`abxVerifyChain` によって SSL 通信用証明書を作成，nB1 - nB2 間で SSL 通信が可能であることを `sslTest` で確認した．

(4) 次に，A0 と B0 が，互いのグループの代表として 1 ホップの信頼関係を結び，AUC を交換した後，A0 と B0 の `abxAuthd` を再起動する．

(5) この時点で，nB1 - nA1，nA0 - nB2 のような，グループ間をまたがる信頼チェインが存在することを `abxFindChain` で確認し，`abxVerifyChain` を使用して SSL 通信証明書を作成した．最終的にすべての信頼関係チェインにおいて認証を行い，両グループに所属する全ユーザに関して，全対全の SSL 通信が `sslTest` で行えることを確認した．

(6) さらに，nB1 が nA1 に不正を行ったと想定し，6.7 節で示した不正を行ったユーザの特定の手順を行い，nA1 に B1 の OURCA 証明書の SubjectDN を提示できた．

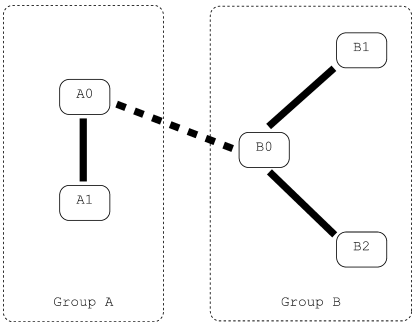

図 2 動作検証に使用した信頼関係トポロジ
Fig. 2 A trust network topology for verification.

これらの検証により，本節冒頭であげた，提案手法 AUBReX の認証機構の基本機能がすべて動作することが確認できた．

8. 実装上の工夫による有効性の向上

8.1 署名付き信頼関係キャッシュ法

本件提案手法のサンプル実装では，認証時に複数のピアに接続を行う点と，認証時には信頼関係チェインに存在するユーザがオンラインでなければならない点において，有効性（Availability）が低い．この有効性の低さは，信頼関係を確立している証拠である OURCA 証明書，すなわち個人情報が中央集権的に管理されていないことが原因である．

ディレクトリサービスに「**ユーザ間の信頼関係の公証役場**」としての機能を付加するという実装上の工夫により，有効性の向上が可能である．この手法を以下**署名付き信頼関係キャッシュ法**とし，その概要を述べる．

(1) ディレクトリサービスは鍵ペアを持つ．公開鍵はディレクトリサービスを起動するサイトの適切なホスト（SSL サーバ認証が可能な Web サーバ等）より鍵指紋とともに公開されるものとし，各ユーザはそれを事前に入手しておく．

(2) 任意のユーザ a' のサインオン時，ディレクトリサービスはユーザ a' から提示された信頼関係リスト内の各 AnonCN $b_{a'_i}$ について，brvp$(ds, a', b_{a'_i})$ を行う（ds はディレクトリサーバ）．

(3) a' の信頼関係リスト内に存在した $b_{a'_i}$ のサインオン時，$b_{a'_i}$ が提示した信頼関係リスト内に a' が存在すれば，brvp$(ds, b_{a'_i}, a')$ を行う．これにより，(2) と合わせて a' と $b_{a'_i}$ の信頼関係が双方向で検証できたことになる．

(4) 信頼関係が双方向で検証できた 2 ユーザ a'，$b_{a'_i}$ について，ディレクトリサービスは a' || '.' || $b_{a'_i}$ を生成する．これを pairStr とする．sha1(pairStr) をディレクトリサービスの秘密鍵で署名し，pairStr とともに適当なフォーマットを持つ文字列で表現する．これを**署名付き信頼関係** とする．

(5) ディレクトリサービスは署名付き信頼関係をキャッシュしておき，abxDumpOnlineUser コマンドをユーザが発行した時点でローカルデータベースに格納される．

(6) 任意のユーザ u' は abxFindChain コマンドを使用して相互認証を行う相手であるユーザ w' までの信頼関係チェインを得る．このとき，ローカルデータベース内に任意の 2 ユーザ x'，y' の署名付き信頼関係が存在し，事前に入手しておいたディレクトリサービスの公開鍵を使用した署名付き信頼関係の署名の検証が成功したならば，x'，y' がピアリストに存在しない，すなわち両者がサインオンしていなくても，x'，y' を含む信頼関係チェインをつなげてよいものとする．

(7) abxVerifyChain コマンドは，u' - w' 間の信頼関係チェイン内で隣接する各 2 ユーザ x'，y' について，ローカルデータベース内に x'，y' の署名付き信頼関係が存在する場合には，ディレクトリサービスの公開鍵を使用して，署名付き信頼関係の署名を検証する．この検証が成功した場合，brvp(u', x', y')，brvp(u', y', x') を省略する．

ディレクトリサービスによる署名付き信頼関係のキャッシュ期間，リボケーションの方法等，さらに考察が必要であるが，この手法により，ディレクトリサービスにはいっさい個人情報を公開することなく，最良のケースではすべての BRVP が省略可能となり，認証の有効性向上が期待できる．ただし，ユーザが新たに 1 ホップの信頼関係を確立した後は，最低 1 回のサインオンを行わないと最高の有効性が発揮されないことになる．

また，署名付き信頼関係キャッシュ法を採用した実装を用いる環境においては，ディレクトリサービスを起動するホストマシンへのなりすましを行い，虚偽のディレクトリサービスによる虚偽の署名付き信頼関係をユーザに提供するという攻撃が考えられる．そのため，署名付き信頼関係キャッシュ法を採用したディレクトリサービスの実装にあたっては，ディレクトリサービスの公開鍵を用いて，信頼のおける第三者 CA から SSL サーバとデジタル署名の機能を持つ証明書の発行を受け，第三者 CA のルート証明書とディレクトリサービスの公開鍵を上記の手順 (1) で示したように適切にユーザに公開したうえで，ディレクトリサービスとユーザ間はつねに SSL 接続を用いてユーザ側がディレクトリサービスを SSL 片側サーバ認証する方式をとることを推奨する．

8.2 中央集権的 CA によるアカウント管理・認証方式との比較

署名付き信頼関係キャッシュ法を採用した場合，ディレクトリサービスがいわゆる認証サーバとして機能するように見えるが，次にあげる 2 点の理由より，署名付き信頼関係キャッシュ法を用いたディレクトリサービスと認証サーバとは，本質的に異なるものである．

(1) 署名付き信頼関係キャッシュ法を採用しても，ユーザ間での 1 ホップの信頼関係の構築という手法が，本件提案方式のユーザアカウントの管理方式であることにまったく変わりはなく，中央集権的 CA や認証サーバのように，アカウントデータベース等を自身の管理下に置くことはない．

(2) 本件提案手法における認証とは信頼関係チェインの検証であり，署名付き信頼関係キャッシュ法を採用しても，この検証自身がユーザのノードにおいて行われることに変わりはない．署名付

き信頼関係キャッシュ法は，任意のユーザノードにおける信頼関係チェイン検証時の BRVP 発行の削減のみを意図して，署名付き信頼関係をユーザに提供する機能をディレクトリサービスに付与するだけである．

しかしながら，署名付き信頼関係キャッシュを採用したディレクトリサービスを用いる場合，公証役場であるディレクトリサービスが検証・署名した任意の 2 ユーザ間の信頼関係を全ユーザが信頼して使用するためには，ディレクトリサービスという中央集権的リソースを全員が信頼しなければならないという点で，従来の中央集権的 CA によるアカウント管理と同様の，中央集権的リソースを信頼するか否かの問題が発生する．

上記の議論を含め，以下に中央集権的 CA もしくは認証サーバによるアカウント管理・認証方式と署名付き信頼関係キャッシュ法を用いた本件提案手法を比較し，署名付き信頼関係キャッシュ法を用いた場合の本件提案手法の得失を考察する．

利点：

(1) **P2P 環境との親和性**

3.2 節で述べたように，CA や認証サーバのような中央集権的リソースを用いてユーザアカウント管理が行われる認証方式では，通信集団基盤内の全ユーザがその中央集権的リソースを信頼し，ユーザ各自の個人情報（フォト ID も含まれうる）を CA や認証サーバ管理者に提供する必要があり，認証方式として P2P 環境との親和性が低い．

それに対し本件提案手法では，署名付き信頼関係キャッシュ法を採用したとしても，ユーザアカウント管理はユーザ個々が行う点，個人情報は現実社会において信頼関係にあるユーザ間にしか提供されない点において，認証方式として P2P 環境との親和性が高い．

(2) **認証機構の耐故障性・有効性**

中央集権的な認証サーバを用い，クライアント側に認証ケーパビリティとしてトークンを提供することで認証を行うような方式においては，認証サーバが停止した場合には認証機構が停止してしまい，P2P 環境内において通信相手となるピア群がオンライン動作しているにもかかわらず認証を介した P2P 接続が不可能となる．バックアップ認証サーバ等を使用してもこの問題は潜在的に存在する．

それに対し署名付き信頼関係キャッシュ法を用いた本件提案手法においては，本件提案手法の，認証時にはその信頼関係チェインに含まれるユーザがすべてオンラインでなければならないという有効性の低さが著しく改善され，かつ，本件提案手法においては認証結果として SSL 相互認証通信用証明書を発行するため，既存ユーザに関しては，ディレクトリサービスが停止していても，6.3 節で述べたようにアプリケーションレベルで使用可能なピアが管理されていれば，それらピアとの SSL 相互認証を用いた P2P 接続が可能である．

欠点：

(1) **信頼モデルと認証機構の複雑化**

中央集権的 CA を使用する場合，ユーザは CA の CP/CPS を了解し，その CA から発行された証明書を持つユーザ間での認証を行うために，その CA のみを信頼すればよい．

それに対し，署名付き信頼関係キャッシュ法を用いた場合，本件提案手法が元来採用している信頼関係チェインによる認証という信頼モデルに加えて，ユーザは署名付き信頼関係をキャッシュしているディレクトリサービスをも信頼する必要が発生し，信頼モデルが複雑化する．さらに，ユーザ証明書の CA による署名の検証という確立された認証機構と比べ，本件提案手法の認証機構は複雑であり，署名付き信頼関係キャッシュ法を用いることはさらなる複雑化を引き起こす．認証機構の複雑化は実装等において未知の脆弱性の混入の可能性を増加する．

9. 安 全 性

我々は，本件提案手法の安全性について考察し，以下の 5 種類の攻撃に対応できていれば，本件提案手法は安全であると考えた．

(1) 任意のユーザの AnonCN を不正に使用するいわゆるなりすましが可能か．

(2) 任意の悪意あるユーザ a が，実際には信頼関係を結んでないユーザ b' との間に信頼関係があるかのように振る舞い，a'，b' を含む信頼関係チェインを介して，a が b'，あるいは b' と真の信頼関係を結んだユーザ c' との間の通信を通して不正，もしくはサボタージュを行えるか．

(3) 任意の 2 ユーザ x，y が悪意の元に結託し，x'，y' を信頼関係チェインに含む任意の認証において，その信頼関係チェインの両端（片端が x' もしくは y' の場合も含む）にいるユーザ間の

通信を通して x または y が不正，もしくはサボタージュを行えるか．

(4) 任意の悪意あるユーザ a が，標的 t を欺いて，a，t 間に信頼関係を結ぶことができるか．

(5) 任意の悪意あるユーザ a が，正しい手法に則ってその他ユーザと真の信頼関係を結んだ後，適当な虚偽の個人情報を持つ架空ユーザ aF（複数も含む）を捏造し，aF，aF' をつねに使用することで a，a' の存在を隠蔽したまま不正を行えるか．さらに，悪意あるユーザ b が a と同様の手法で架空ユーザ bF を捏造し，a と悪意のうちに結託，aF と bF の間に信頼関係を構築し，aF'，bF' を含む信頼関係チェインを介して，a，a'，b，b' の存在を隠蔽したまま不正を行えるか．

9.1 攻撃 (1) への対応

まず攻撃 (1) であるが，AIVP によって AnonCN の生成に使用された公開鍵と対になる秘密鍵の所有が検査されることより，その秘密鍵（およびパスフレーズ）が漏洩しない限り，その AnonCN で示されるユーザへのなりすましによるセキュリティ脆弱性は発生しないと考える．また，本件提案手法は基本的に PKI ベースの認証手法の拡張であるので，秘密鍵およびそのパスフレーズの盗用によるなりすましには対処できない．これは，既存の PKI ベースの認証においても，ユーザ個人の秘密鍵およびそのパスフレーズが流出した場合は同様である．ただし，本件提案手法においては，ユーザ個人の秘密鍵はそのユーザの OURCA/AURCA，すなわち CA の秘密鍵でもある点が，既存の PKI ベースの認証システムと異なっている．本件提案手法におけるユーザ個人の秘密鍵が流出すると，本来の秘密鍵所有者の権限の悪用以外に，秘密鍵（およびそれから生成された OURCA/AURCA 証明書．秘密鍵が流出する程度の攻撃を受けたのであれば，自分自身の OURCA/AURCA 証明書だけでなく，自分自身と 1 ホップの信頼関係にあるユーザの OURCA 証明書すべても流出していると考えるのが妥当であろう）を利用して悪意ある他ユーザと 1 ホップの信頼関係を構築する等の悪用も可能になる．したがって，AUBReX における秘密鍵（および AURCA/OURCA 証明書）は，既存の PKI ベースの認証システムにおけるユーザ個人の秘密鍵（および証明書）と同等以上の注意をもって扱われなければならない．

9.2 攻撃 (2) への対応

次に攻撃 (2) であるが，AUBReX での任意の信頼関係チェイン $\texttt{tchain}(u0', uN')$ の検査が 6.5 節 (2)〜(7) で述べたように，BRVP を利用して，$u0'$ から uN'，uN' から $u0'$ の双方向のチェックで行われることより否定できる．$\texttt{tchain}(u0', uN')$ が例示した a'，b' を含み，a が b' との信頼関係の存在を主張したとしても，b が a' との信頼関係の存在を表明しない限り，a'，b' を含む信頼関係チェインの検査は成功しない．また，$\texttt{brvp}(u', a', b')$ を欺くためには，a は，$\texttt{sha1}(\texttt{orc}(b'))$，$\texttt{arc}(b')$，$\texttt{auc}(a', b')$ を知らなければならない．$\texttt{sha1}(\texttt{orc}(b'))$，$\texttt{arc}(b')$ は a が $\texttt{aivp}(a, b')$ を b' に発行することで a にとって容易に入手可能であるが，$\texttt{auc}(a', b')$ は，a' と b' 間に真の信頼関係が存在するときのみに，a 側に存在する．

9.3 攻撃 (3)，(4) への対応

まず攻撃 (3) のように，悪意ある 2 者 x，y が結託した場合について考える．この場合，「AUBReX で信頼関係を結ぶとき，お互いの $\texttt{sha1}(\texttt{orc}(x))$，$\texttt{sha1}(\texttt{orc}(y))$ を単なる 20 バイトの乱数として交換する，もしくは虚偽の個人情報を記述した $\texttt{orc}(x)$，$\texttt{orc}(y)$ を交換し，$\texttt{arc}(x')$，$\texttt{arc}(y')$，$\texttt{auc}(x', y')$，$\texttt{auc}(y', x')$ をすべて生成・交換してしまう」という攻撃が考えられる．この攻撃は，AIVP，BRVP ともに，本人，信頼関係にある友人の OURCA 証明書の有無が，OURCA 証明書の SHA1 ハッシュ値の提示によって，間接的にしか検査できないことを利用するものである．x，y の悪意は，x' と y' を含む信頼関係チェインで認証される，1 人以上の「標的」に向けられることになる．ここで重要なのは，x，y がこのような標的を得るためには，必ず自分たち以外の誰か，標的そのもの，もしくは標的と信頼関係チェインでつながる誰かのうち，1 人以上と信頼関係を結ばなければならないという点である．このように考えると，攻撃 (3) と攻撃 (4) は同義であることが分かる．ここで，この「騙されて悪意のあるユーザと信頼関係を結ぶ誰か」を v とし，騙す側（x もしくは y）を A とする．

まず，OURCA 自身を交換せず，OURCA のハッシュ値を適当な乱数として交換することで虚偽の信頼関係を確立した x，y のどちらかが，v を騙して信頼関係を確立しようとたくらんだとする．この場合，3.3 節 (4) で述べたように，v が A との OURCA 証明書の交換時に，OURCA 証明書の有無を正しくチェックすればよい．A が OURCA 証明書を提示しない，できないのであれば，当然 v はこれらのユーザと信頼関係を結ばない．

次に，x，y が交換しあった OURCA 証明書内の個人情報が虚偽である場合を考える．この場合には，さらに 2 つの事態が考えられる．第 1 は，A と v が既

知の間柄で AUBReX での信頼関係を確立すること，第 2 は，A と v が当初初対面に近い状態であり，A が現実世界において，奸計をもって虚偽の個人情報を v に信じさせ，AUBReX での信頼関係を確立することである．

前者に対しては，やはり 3.3 節 (4) で述べた，OURCA 証明書の内容のチェックを十分行うことで対応できる．逆に後者への対応は，現実世界の信頼関係を認証に使用するという本件提案手法では不可能である．この事態は，言葉巧みに標的を騙したうえで得た信用を悪用することで悪事を働くという点で，どのような認証方式においても起こりうると考える．

ここで，最悪の事態として v が A の奸計によって騙されて結ばされた信頼関係を利用した信頼関係チェインを介した認証が原因で，標的 t が被害にあった場合への対応を考察する．信頼関係チェインは t' - v' - y' - x' のようにつながっているものとする．

t は攻撃者の AnonCN である x' さえ分かれば，認証の元となった信頼関係チェインをたどることで，不正を働いたユーザ x' の個人情報を持っているはずである y' に対し，x' の個人情報の開示を要求することができる．

まず y' が t に x の個人情報を開示した場合を考える．

(1) y' と x' が交換しあった OURCA 証明書内の個人情報は虚偽であるため，y' が t に開示した x の個人情報も当然虚偽である．この場合，不正の捜査者がその個人情報が虚偽であることを看破するのは時間の問題である．

(2) x の虚偽の個人情報を提供したのは y' であり，y' は捜査の撹乱を行ったことになるので，t ならびに不正の捜査者は，その責任を追求するために y' と直接信頼関係にある v' に y' の個人情報の開示を要求することになる．

(3) しかし v は y の虚偽の個人情報に騙されて y との間に信頼関係を確立しているので，v' が t に開示した y の個人情報も虚偽であり，最終的に t ならびに不正の捜査者は v に責任を追求することになる．

(4) この時点で v が y に騙されていたことが発覚し，y の悪意が露呈する．しかし，y と v が本件提案手法での信頼関係を構築しているのであれば，y は v とは当初初対面に近い状態から v が y に個人情報を渡してよいと決心するに足るほどの信頼を勝ち得るために，v と現実社会において接触を行っているはずである．このような状況であれば，v が不正の捜査者に y の身体的特徴，対面した日時，場所等を捜査情報として提供することは容易である．これらの情報は，現実社会での個人の特定において，場合によっては個人情報より有効な情報となる．

一方，y' が t に x の個人情報を開示しなかったとしても，t および不正の捜査者は v' から y という虚偽の個人情報を入手することになり，やはり v へ責任が追求される．以降，前述の y' が t に x の個人情報を開示しなかった場合と同様に，v から y の身体的特徴等の捜査情報が不正の捜査者に提供されることになる．

このように，本件提案手法では，

- 個人情報が正しく記載された OURCA 証明書を持たなければ，他人と信頼関係を結べない，
- 悪意を持ったユーザが結託して虚偽の個人情報を含む OURCA 証明書を使用する，もしくは OURCA 証明書をまったく使用しないで，「悪意ある信頼関係」を確立したとしても，誰かに悪事を働くためには，その悪意のユーザらが標的としている人間と信頼関係チェインでつながる「善意の第三者」と正しく信頼関係を結ばなければならない，
- 最悪，悪意のユーザらがうまく「善意の第三者」を欺いて信頼関係を結び，その第三者，その他悪意の標的に不正を働いたとしても，最終的にはその悪意のユーザらの誰かに対して責任が追求できるだけの機構が用意されている，

の 3 点の理由で，本件研究が目的とする集団通信基盤において攻撃 (3)，攻撃 (4) に対応できると考える．

9.4 攻撃 (5) への対応

最後に攻撃 (5) について考える．ここで，a，b のように架空ユーザ aF，bF を使用する悪意のユーザを総称して x とし，x が生成する架空ユーザを xF と総称する．

xF が善意の第三者 v と信頼関係を結ぶ場合には，x 本人が xF を詐称して v と接触することになる．まず，v が x と既知の間柄であった場合を考える．

(1) v と xF が 3.3 節 (4) の手順に基づいて信頼関係を確立しようとするならば，x 本人を xF とする詐称はただちに露呈し，v は xF と信頼関係を結ばない．

(2) (1) により，v が x と既知の間柄である場合，x

と v が信頼関係を確立するためには，x は必ず x の真の個人情報を使用した OURCA/AURCA 証明書を使わなければならないことになる．したがって，xF' は必ず $\cdots$ - v' - x' - xF' のように，x' を次のホップに持つ末端としてのみ信頼関係チェインに存在することになる．

(3) 悪意の標的が t であるとし，t' - $\cdots$ - v' - x' - xF' のような信頼関係チェインによる認証を介して，xF' が t' に不正を行ったとする．t は信頼関係チェインをたどることで，x' に対して xF の個人情報の開示を要求する．

(4) もし x' が xF' という自分が作成した架空ユーザの個人情報の開示を拒むのであれば，x' を不正を行ったユーザを庇う者として x' の責任を問うために，t および不正の捜査者は v' から x の個人情報を提供してもらうことができるため，結局 x が特定できる．

(5) 逆にもし x' が xF の個人情報を開示したとしても，その個人情報は x によって捏造されたものである．この場合，不正の捜査者がその個人情報が虚偽であることを看破するのは時間の問題である．

(6) xF の虚偽の個人情報を提供したのは x' であるから，捜査の撹乱を意図した悪意の責任を x' に問うために，t および不正の捜査者は v' から x の個人情報を提供してもらうことができるため，結局 x が特定できる．

(7) ここで x が a であったとして，a が架空ユーザ $aF2$ を生成し，aF と $aF2$ の間に信頼関係を構築し，$aF2$ を使用して t' - $\cdots$ - v' - a' - aF' - $aF2'$ の信頼関係チェインによる認証を介して $aF2'$ が t' に不正を行う場合と，$aF2$ ではなくて a の共犯者である b の架空ユーザ bF を使用して t' - $\cdots$ - v' - a' - aF' - bF' の信頼関係チェインによる認証を介して bF' が t' に不正を行うという両者の場合を考える．どちらの信頼関係チェインにおいても，t' に近い（t' から見てホップ数の小さい）側に x' が存在するので，不正を行った実行犯である aF'，bF' の側から架空ユーザの個人情報が虚偽であることを暴いていけば，最終的に x の責任を追求することができる．x' が aF' 以降を庇って aF の虚偽の個人情報を提供しなかったとしても結果は同様である．

v と x が既知の間柄でなかった場合，v が x に騙されて架空ユーザ xF と信頼関係を確立してしまう可能性がある．これは攻撃 (4) において，善意の第三者 v と悪意のユーザ y が y の虚偽の個人情報で v を騙して両者の間に信頼関係を確立させた場合と同様の状況である．9.3 節で述べたように，このような状況であっても，最終的には v から v 自身を騙した人間，すなわち x の身体的特徴等の捜査情報を不正の捜査者に提供でき，不正の捜査者はこの情報から x が特定できる．

また，1 人で架空ユーザを複数作成したり，架空ユーザ同士が結託して，彼らの真の個人情報を持つユーザを隠蔽しようとしたりする行為は，いたずらに信頼関係チェインのホップ数を増加させることにつながり，結局悪意の標的となるユーザの認証ポリシであるホップ数制限に抵触してしまうことになり，認証自体が却下される可能性がある．

10. 関 連 研 究

本件提案手法は PKI ベースの認証機構である．PKI ベースの認証に利用される X.509 証明書の証明主体は SubjectDN，もしくはそれに含まれる CommonName 等の，いわゆる「名前」である．これに対し，2.3 節であげた SPKI では，証明主体を公開鍵そのものとすることで，基本的に「名前」を必要としない権限管理機構の実現を目的としている．特にこの SPKI の「名前」を必要としないことによる匿名性の高さに着目した，プライバシを重視したアクセス制御機構の実現に関して，梅澤ら[3] の研究がある．匿名性の高さに関し，本件提案手法と SPKI には共通点があるが，本件提案手法は認証機構であり，SPKI は権限管理機構である点が異なっている．SPKI を規定する RFC 2693 の “3.3 SPKI Certificate” では，基本である “権限 - 公開鍵” のマッピング以外に，“権限 - 公開鍵 - 名前” のマッピングを行うことで，不正を行ったユーザの特定を可能にする手法に関しても触れているが，名前をマッピングに含ませた場合に，どのように実際に不正を働いたユーザを特定するかに関しての規定は行われていない．一方，本件提案手法は，匿名性の保持と不正を行ったユーザの特定の両者を可能とする認証機構として，現実社会で信頼関係にあるユーザ同士が，現実社会おいて相手を特定できるような個人情報が正しく記載された OURCA 証明書を交換しあうこと，そのようにして確立された信頼関係を信頼関係チェインとして表現し，それを保持しておいたうえで，何か不正があった場合には，信頼関係チェインをたどることで不正を行ったユーザ，さらにそれを庇うユーザの特定を行う，という手法を，認証機構の一部として規定

している．

また，RFC 4683[4] では，X.509 証明書の subjectAltName 拡張部分に SIM と呼ばれる方法でエンコードした文字列を使用することで，プライバシ的に慎重な扱いを要する識別子を安全に相手に公開する手法を規定している．認証主体である識別子をエンコードするという点では，RFC 4683 と本件提案手法は非常に類似している．RFC 4683 SIM の目的は，公開したい相手にのみ自身の個人情報を安全に公開することであり，公開する側が相手への個人情報の公開を許した場合のみ，その個人情報が相手側でデコード可能になる．これに対し，本件提案手法の AnonCN の目的は，個人情報を直接的に識別子としないことで証明書の匿名性を高めることであり，必要なとき（ユーザが不正を行ったとき）には 1 ホップの信頼関係にあるユーザのみが AnonCN からその生成の元となった個人情報，すなわち OURCA 証明書の内容を知ることが可能である．この点で，RFC 4683 と本件提案手法は大きく異なっている．

現実世界の個人間の信頼関係を記述するための枠組みとして，FOAF[5]（Firend Of A Friend）がある．FOAF では自分自身と，自分自身が持つ友人の個人情報を RDF/XML を用いて記述する．FOAF を利用する P2P 用の認証方式に関する研究も行われている[6]．FOAF による信頼関係記述を利用する認証方式は，友人関係を直接記述できる点に関し，本研究での提案方式のようにクロス証明書を用いるよりは直裁的である．本研究の提案方式ではクロス証明書を用いることにより，最終的な認証成功の結果として，そのクロス証明書をルート CA とする SSL 用証明書を相互発行する．生成された SSL 証明書は任意のアプリケーションで使用可能であり，SSL をサポートした既存のアプリケーションとの親和性が高い．

本研究での提案手法である AUBReX の信頼モデルは，3.2 節で述べたように，信頼関係チェインの概念によって拡張されたクロス認証である．通常クロス認証で生成される CA トポロジは水平（メッシュ）型となる．ユーザの信頼関係がメッシュとして表現される信頼モデルとして，Java による P2P 通信ライブラリ JXTA[7] 用に考案された信頼モデルである Poblano[8] が採用している Web of trust（「信頼の輪」）がある．Web of trust は元来 PGP において公開鍵の署名関係を表現する概念であり，Poblano はそれを X.509 証明書ベースの認証に適用している．Web of trust と本研究の信頼モデルの違いは，前者では信頼関係が単方向であってよい，すなわち任意の 2 ユーザの間で公開鍵/証明書がお互いに署名/発行されていなくてもよいが，後者では証明書が必ず相互に発行されていなければ信頼関係を成立させないという点である．本件提案手法では，相互に信頼する，すなわち証明書を相互に発行することで，信頼関係チェインに方向の概念が発生する．AUBReX ではこの性質を利用して信頼関係チェインの双方向の検証を行うことで，9 章の (2) であげた，単独犯での虚偽の信頼関係の報告による攻撃に対する堅牢性を実現している．

また，Poblano の信頼モデルは，さらに P2P コミュニティ内の任意のピアの評判（Reputation）を信頼度に反映させている．この評判を認証に直接反映させる手法の研究として，Hyunrok[9] らの AARP（Adaptive Authentication Protocol based on Reputation）がある．AARP では，P2P 通信のピア間の認証において，ピアどうしが交換する情報の重要度に応じて，認証方式が動的に変更される．重要度は商取引か否か，ピアどうしの趣味嗜好，さらに発呼側のピアのその趣味に関するコミュニティ内ので評判をもとにして算出し，最終的には 3 段階にカテゴライズされ，それぞれ商用 CA から発行された証明書，（そのシステム，アプリケーションの管理側が持っている）非公式な CA から発行された証明書，自己署名証明書の 3 つの証明書を使い分けるものとしている．商用 CA，非公式な CA を使うことから中央集権・集約的なアカウント管理が行われることになり，使用できるようになるまでの手順が比較的煩雑になる等，3.2 節で触れた従来の PKI ベースの認証機構の問題点をそのまま有することが考えられる．さらに，個人情報が入ったユーザ証明書が直接見ず知らずの第三者にわたる可能性が高いことも，プライバシに敏感なユーザにとっては問題となろう．

11. ま と め

本稿では提案手法 AUBReX について，その信頼モデルや匿名性の実現手法，不正を行ったユーザの特定方法，認証プロトコルの概要と実際の認証手順等を説明した．またサンプル実装を用いた認証機構の基本機能の動作検証を行い，良好な結果を得た．これより，我々は，本件研究が目的としている P2P 集団通信基盤の確立のための認証方式として，提案手法 AUBReX が，匿名性の保持と不正を行ったユーザの特定という相反する目的を両立させるための有効な認証方式であると考える．

今後，サンプル実装をさらに実運用を意識した実装へと改良し，それを利用した認証スケーラビリティの

検証，8 章で述べた手法による有効性の向上，CRL 機能の設計および実装，実際のアプリケーションへの適用等を行いたいと考えている．また，関連研究で触れた，コミュニティ内での評判を認証に取り入れる手法に関し，AUBReX での採用を検討したいと考えている．

謝辞 本研究の一部は，文部科学省科学研究費補助金基盤研究 A 課題番号 172002，特別研究員奨励費課題番号 177324，および JST 戦略的国際科学技術協力推進事業「日米サイエンスグリッドにおけるセキュリティ基盤の構築」による．

参 考 文 献

1) Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B. and Ylonen, T.: SPKI Certificate Theory, RFC 2693 (1999).
2) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *Supercomputer Applications*, Vol.11, No.2, pp.115–128 (1997). htpp://www.globus.org/
3) 梅澤健太郎，齋藤孝道，奥乃 博：プライバシーを重視したアクセス制御機構の提案，情報処理学会論文誌，Vol.42, No.8, pp.2067–2076 (2001).
4) Park, J., Lee, J., Lee, H., Park, S. and Polk, T.: Internet X.509 Public Key Infrastructure Subject Identification Method (SIM), RFC 4683 (2006).
5) The ‘friend of a friend’ project: FOAF. http://www.foaf-project.org/
6) Golbeck, J., Parsia, B. and Hendler, J.: Trust Networks on the Semantic Web, Lecture Notes in Computer Science, Vol.2782/2003, pp.238–249 (2003).
7) Project JXTA. http://www.jxta.org/
8) Chen, R. and Yeager, W.: Poblano: A Distributed Trust Model for Peer-to-Peer Networks, White Paper, Sun Microsystems, Inc. (2001). http://www.jxta.org/docs/trust.pdf
9) Lee, H. and Kim, K.: An Adaptive Authentication Protocol based on Reputation for Peer-to-Peer System, *SCIS 2003*, Vol.1/2, pp.661–666 (2003).

(平成 19 年 7 月 23 日受付)
(平成 19 年 11 月 9 日採録)

平野 基孝（学生会員）

昭和 40 年生．昭和 63 年図書館情報大学図書館情報学部図書館情報学科卒業．同年シャープ（株）入社．平成 2 年（株）SRA 入社．計算機性能評価ツール，コンパイラ，ネットワーク通信ライブラリ等の開発に従事．平成 17 年筑波大学大学院理工学研究科理工学専攻修了．同年同大学院システム情報工学研究科コンピュータサイエンス専攻後期博士課程編入学．分散並列プログラミング，分散並列コンピューティング環境に興味を持つ．

首藤 一幸（正会員）

平成 8 年早稲田大学理工学部情報学科卒業．平成 10 年同大学助手．平成 13 年同大学大学院理工学研究科情報科学専攻博士後期課程修了．同年産業技術総合研究所入所．平成 18 年ウタゴエ（株）取締役最高技術責任者．博士（情報科学）．分散処理方式，プログラミング言語処理系，情報セキュリティ等に興味を持つ．SACSIS2006 最優秀論文賞．IPA 未踏ソフトウェア創造事業スーパークリエータ認定．情報処理学会平成 18 年度論文賞．情報処理学会平成 19 年度山下記念研究賞．日本ソフトウェア科学会，IEEE–CS，ACM 各会員．

田中 良夫（正会員）

昭和 40 年生．平成 7 年慶應義塾大学大学院理工学研究科後期博士課程単位取得退学．平成 8 年技術研究組合新情報処理開発機構入所．平成 12 年通産省電子技術総合研究所入所．平成 13 年 4 月より独立行政法人産業技術総合研究所．現在同所グリッド研究センター主幹研究員．博士（工学）．グリッドにおけるプログラミングミドルウェアおよびグリッドセキュリティに関する研究に従事．IC 1999，HPCS 2005，SACSIS 2006，平成 18 年度情報処理学会論文賞．ACM 会員．

佐藤 三久（正会員）

昭和 34 年生．昭和 57 年東京大学理学部情報科学科卒業．昭和 61 年同大学大学院理学系研究科博士課程中退．同年新技術事業団後藤磁束量子情報プロジェクトに参加．平成 3 年通産省電子技術総合研究所入所．平成 8 年新情報処理開発機構並列分散システムパフォーマンス研究室室長．平成 13 年より筑波大学システム情報工学研究科教授．平成 19 年より同大学計算科学研究センターセンタ長．理学博士．並列処理アーキテクチャ，言語およびコンパイラ，計算機性能評価技術，グリッドコンピューティング等の研究に従事．IEEE，日本応用数理学会各会員．