管理者に対して秘匿性を保証したセキュアインスタントメッセージングプロトコル

菊池浩明[†] 多田美奈子[††] 中西祥八郎[†††]

本論文はインスタントメッセージングサービスにおけるセキュリティ強化についての研究である．セキュリティに関した要求条件と仮定を述べた後，インスタントメッセージングに適したDiffie-Hellman改良プロトコルを示す．提案プロトコルの主要な特徴は，不正なサーバ管理者が平文を入手し，データマイニング技術を適用して利用者のプライバシ情報を取得することを防止していることである．通信と計算量のコストの観点から，本提案プロトコルを評価し，いくつかの既存のプロトコルとの比較を行う．さらに，セキュアインスタントメッセージングシステムの試験実装について報告する☆．

Secure Instant Messaging Protocol Preserving Confidentiality against Administrator

Hiroaki Kikuchi,[†] Minako Tada[††] and Shohachiro Nakanishi[†††]

This paper studies a security enhancement of instant messaging service. After the requirements and the assumptions are addressed, a modified Diffie-Hellman protocol suitable to instant messaging is presented. The main feature of the proposed protocol is to prevent malicious administrator from intercepting plain message and applying data mining techniques to obtain privacy of end users. From the viewpoint of communication and computational costs, the proposed protocol are evaluated and the comparison of some existing protocols is given. In addition, a trial implementation of secure instant messaging system is demonstrated.

1. はじめに

インスタントメッセージング（IM）サービスのビジネスへの利用が浸透している．Nielsenによると，2002年7月時点で日本における主要な4つのIMサービス（MSN，Yahoo，ICQ，AOL）のユーザ数の合計が250万人に達する[1]．これは，勤務先や学校から接続するユーザ数2100万人[3]の約10%に達している．米国ではさらにオフィスの約70%に普及しているという報告[4]があり，今後も世界的に導入が進むことが予想される．オンライン状況を互いに交換するプレゼンス通知，動的にメンバを構成してのチャットなど，従来の電子メールでは実現できなかった機能を補うことが認識され始めているためであろう．

しかしながら，ビジネスとして利用するならば，そのセキュリティ強化は必然的である．Herbslebらは，グループウェアの観点から，IMの有効性とプライバシの課題を指摘している[11]．オープンソースでXMLベースのIMサービスを開発しているJabber Software Foundation[12]では，SSL encryptionを用いたシステムを開発している．商用の製品も多く出ており，Marsanのサーベイ[4]による製品一覧を**表1**に示す．これによると，多くはSSLによる暗号化を採用しているが，SSLは閉じた通信路の暗号化を提供するサービスであり，多くの利用者との複雑なマルチキャストをともなうIMに利用するためには，サーバを信頼点として，いったん復号化した後に中継することとなる．

このような実現方法をとると，サーバにはすべてのメッセージが集まるために，プライバシの観点からの問題が生じる．大規模なデータを解析して意味のある消費者動向を探るデータマイニング技術の導入も進ん

† 東海大学電子情報学部情報メディア学科
Department of Information Media Technology, School of Information Technology and Electronics, Tokai University
†† 東海大学大学院工学研究科
Graduate School of Engineering, Tokai University
††† 東海大学電子情報学部情報科学科
Department of Human and Information Science, School of Information Technology and Electronics, Tokai University

☆ 本論文の初期のバージョンは，文献9)，10)で発表している．

表 1 セキュアインスタントメッセージング製品[4)]
Table 1 Secure Instance Messaging products.

ベンダ	ソフトウェア名称	暗号技術
Bantu	Bantu Messnger and Presence Platform	独自暗号プロトコルおよび SSL
Divine Software	MindAliga	独自認証プロトコル
Ikimbo	Omniprise	SSL
Jabber	Communications Platform	SSL（XML encryption）
JabCast	Secure Realtime Communications	DES

でおり，ビジネスの機密が漏れる恐れが否定できない．実際，NetRating 社では IM を広告媒体の対象と見なし，IM トラフィックの月間利用実態レポートのサービスを開始している[1)]．かといって，サーバへの信頼をまったく持たない P2P に代表される分散技術では，オンライン状態を交換するために動的に通信相手を検索しなくてはならず，現在の IM との親和性が劣る．サーバを必要としない鍵管理プロトコルもいくつか提案されているが[5),6)]，それらのほとんどはグループメンバが固定ですべてオンラインであることが仮定されており，もはや IM のモデルとは整合しない．

そこで，我々は，現在普及している（コンシュマ型）IM のモデルへの適応を前提とし，オンライン状況などのプレゼンス通知やメッセージ交換には，サーバを介するモデルを考える．ただし，サーバに対してもメッセージの秘匿性が必要であると仮定する．サーバには前述のデータマイニングの例のように，メッセージの中味を知りたいという動機があるためである．本論文では，IM に固有の動的な同報グループ生成を効率良く実現するために，Diffie-Hellman（DH）の鍵交換プロトコル[2)]に基づいたグループ鍵交換プロトコルを提案し，その実行可能性を検証するために，提案プロトコルを試験実装して，実装上の課題や処理時間などの評価を報告する．本提案プロトコルは，次の特徴を持つ．

(1) 中継サーバは暗号文を復号化できない．
(2) （同報通信を提案する）イニシエータの処理は，同報するメンバの数 n に依存しない．

本論文では，まず，2 章で IM の概要とセキュリティ上の要求条件を明らかにする．3 章で，提案するグループ鍵交換プロトコルを示し，従来のプロトコルとの通信量と計算量の観点からの評価を行う．4 章では，提案プロトコルを実装した試験システムについて報告する．鍵交換のタイミングなど，実装に関するいくつかの課題があることや実行処理時間の測定結果などを示し，提案プロトコルが正しく動作することを検証する．5 章で，まとめと今後の課題を述べる．

2. インスタントメッセージング

2.1 概　　要

インスタントメッセージングとは，互いに登録しあっているユーザ間での，互いのオンライン状態の通知と，その中の任意のユーザとのリアルタイムの会話を提供するサービスである．そのサービスを実現するアプリケーションソフトを一般的にインスタントメッセンジャと呼ぶ．このなかでインスタントメッセージとは，一般的な電子メールに対して，比較的小さなサイズで，瞬時に相手に伝送されることを意図しているメッセージ交換機能である．メッセージを交換する基本機能に加えて，ファイルの送受信，音声動画チャット，株式や天気予報など情報の提供といったオプションサービスなどの拡張も行われている．

IM プロトコルの標準化させる動きも始まっている．コンシュマサービスを提供している MSN や Yahoo! などは，IMUnified という IM 標準化団体を結成し，共通の IM 規格を作成しようとしている[13)]．IETF からは，Informational な RFC として，Instant Messaging/Presence Protocol（IMPP）[15),16)]が標準化されている．

2.2 基 本 定 義

(1) メンバリスト
IM では，N 人の全ユーザの中から，n 人の閉じたグループ $L = \{\mathcal{U}_1, \ldots, \mathcal{U}_n\}$ を作り，この L 中だけでオンライン確認や，インスタントメッセージを交換する．この L をメンバリストと呼び，ユーザごとに任意に設定する．

(2) プレゼンス変更の通知
接続状態には，オンライン，オフラインのほかに，退席中，取り込み中，電話中，などがあり，メンバリストのユーザ間で各々の状態をリアルタイムに通知する．また，オンラインになる動作をサインインと呼ぶ．

(3) メッセージング
メッセージの送受信には，ピアツーピアと同報通信の 2 種類がある．オンラインになっている

表 2 メーリングリストとの比較
Table 2 Comparison between instant messaging and mailing list.

	IM	ML
メッセージ長	短い	長い
即時性	必要（オンライン）	不要（オフライン）
メンバリスト	ユーザによって異なる	共通
同報通信グループ	動的	静的
1 ユーザあたりのスケーラビリティ	小	大

メンバならば，同報通信に「招待」することで，グループに動的に追加することが許されている．ここでは，同報通信を開始し，追加のための招待を能動的に行うユーザを，そのセッションのイニシエータと呼ぶ．

2.3 IM のセキュリティモデル

セキュアな IM が満たすべき要求条件を次のように定める．

- 即時性
  メッセージ長は短く，ラウンド数は多い．それゆえ，処理は軽くなくてはならない．
- グループ内に閉じた認証
  メンバリストのメンバはすべてよく知っているユーザであり，その意味で第三者による厳密な身分証明などは必要としない．
- 機密性
  サーバはメッセージを盗聴する動機があり，サーバに対しても機密性を満たす必要がある．
- 弱いスケーラビリティ
  メンバリストへの追加は，互いに第三種的な手段で相手を認証しており，同意がとれているときに限り行われるものとする．したがって，単一のユーザが管理するリストの大きさは限定される．
- 動的な同報通信
  同報通信を行うグループのメンバは頻繁に追加退出を行うものとする．

同様な閉じたグループ内でのメッセージ交換サービスとして，メーリングリスト（ML: Mailing List）がよく知られている．しかし，上で述べたように，IM との間ではいくつかの違いがあり，これを**表 2** で整理する．

3. 提案鍵交換プロトコル

3.1 概　　要

提案プロトコルは，DH 鍵交換プロトコル[2]に基づいている．すべての 2 者間で DH を繰り返せば，サーバに知られることなく秘密を共有することができる．しかし，IM モデルにおいては，メンバリストのすべてがオンラインとなる確率は小さいので，すべてのメンバについて DH を準備するのは無駄が多い．そこで，通常は各自に 1 つずつ定める秘密情報を 2 つずつに拡張し，片方をサーバに管理させることとする．イニシエータはセッションごとに異なる乱数を生成し，サインインのときにサーバに送信する．サーバは，オンラインになっているメンバについてだけ，イニシエータの情報を各々のメンバの秘密情報を用いて変換して伝える．このような構成により，イニシエータの処理はメンバ数に依存せず最小化され，動的な同報通信を実現する．秘密鍵がすべて分からない限り，サーバには鍵情報は漏れない．

グループ内のメンバとイニシエータとの間の秘密情報が共有できた後，共通鍵暗号を用いて同報通信に用いるセッション鍵を交換する．このような，ハイブリッドな構成にすることにより，IM に必要な即応性を満たす．

3.2 準　　備

$\mathcal{S}$ ：鍵生成サーバ
$\mathcal{U}_i$ ：ユーザ $(i = 1, \ldots, n)$
L_i ：$\mathcal{U}_i$ のメンバリスト
p, q ：q が $p-1$ を割りきるような大きな素数
g ：p の乗法群 $Z_p^* = \{1, \ldots, p-1\}$ の位数 q の部分群の原始元
$\mathcal{E}_K, \mathcal{D}_K$ ：鍵 K を用いた共通鍵暗号アルゴリズム
$h(\cdot)$ ：一方向性セキュアハッシュ関数

3.3 登録プロトコル

Step 1-1： サーバ $\mathcal{S}$ は各 $\mathcal{U}_i$ について $a_i \in Z_q$ を定め，$g_i = g^{a_i} \bmod p$ を参加証として $\mathcal{U}_i$ に秘密に送信する．

Step 1-2： $\mathcal{U}_i$ は秘密鍵 $x_i \in Z_q$ をランダムに生成し，$y_i = {g_i}^{x_i} \bmod p$ をサーバ $\mathcal{S}$ へ送り返す．

Step 1-3： ユーザ $\mathcal{U}_1$ がリスト $L_1 = \{\mathcal{U}_1\}$ に，新規ユーザ $\mathcal{U}_2$ を加えたいとする．$\mathcal{U}_1$ は $\mathcal{S}$ から y_2 を送ってもらい，y_1 を $\mathcal{U}_2$ へ $\mathcal{S}$ 経由で送信する．$\mathcal{U}_2$ は，要求を受理するならば $\mathcal{S}$ へ伝え，自分のリスト L_2 に y_1 を追加する．これを繰り返し，$\mathcal{U}_1$ は $L_1 = \{\mathcal{U}_1, \ldots, \mathcal{U}_n\}$ と公開鍵 $y_1, \ldots, y_n$ を，$\mathcal{S}$ は各ユーザについての $a_1, \ldots, a_n$ を秘密に持つ．

3.4 ピアツーピアの鍵交換

ユーザ $\mathcal{U}_i$ と $\mathcal{U}_j$ の 2 者間での秘密鍵を共有するプロトコルを示す．

Step 2-1： $\mathcal{U}_i$ はそのセッションにおける乱数 $r \in$

Z_q を生成し，$z = g^r \bmod p$ を計算して $\mathcal{S}$ に送信する．

Step 2-2： $\mathcal{S}$ は，$\mathcal{U}_j$ に対応する a_j を用い，$b_j = z^{a_j} \bmod p$ を生成し，$\mathcal{U}_i$ の ID とともに，$\mathcal{U}_j$ に送信する．

Step 2-3： $\mathcal{U}_i$ は $\mathcal{U}_j$ の公開鍵 y_j を用いて，

$$k_{ij} = h({y_j}^r \pmod p)$$

を計算する．一方，$\mathcal{U}_j$ は

$$k_{ji} = h(b_j^{x_j} \pmod p)$$

を計算し，鍵を共有する．

ピアツーピアだけで通信するのであれば，ここで得られた k_{ij} を用いてメッセージ m を暗号化 $\mathcal{E}_{k_{ij}}(m)$ する．

3.5 同報通信

イニシエータ $\mathcal{U}_1$ が，そのメンバリスト $L = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_n\}$ との間で同報通信をするプロトコルを示す．ただし，ここで，$\mathcal{U}_1$ が Step 2-1 を実行するのは 1 度だけであり，$i = 2, \ldots, n$ について $\mathcal{S}$ のみが Step 2-2 を実行することに注意せよ．すなわち，$b_2 = z^{a_2}, , \ldots, b_n = z^{a_n}$ を各々に送信する．

Step3-1： イニシエータ $\mathcal{U}_1$ はセッション鍵 k^* をランダムに生成し，$i = 2, \ldots, n$ の各々について，$E_i = \mathcal{E}_{k_{1i}}(k^*)$ を送信する．

Step3-2： $\mathcal{U}_i$ は $k^* = \mathcal{D}_{k_{1i}}(E_i)$ を得る．k^* でメッセージ m を暗号化し，同報通信を開始する．

3.6 評　　価

提案プロトコルの特徴を以下に示す．

- $\mathcal{S}$ はセッション鍵 k_{ij} が分からない（サーバに対する機密性）．
- イニシエータ（$\mathcal{U}_i$）の処理（通信コスト，ラウンド）はメンバリスト L_i の大きさに依存しない．L_i のメンバのオンライン状況によって，処理を新たに引き起こされることもない（動的な同報通信）．
- 鍵共有の安全性は DH と同等．
- メンバリストへの追加は双方の同意のもとで行われるので，ここで Step 1-3 を実行するのは合理的である（弱いスケーラビリティ）．

それゆえ，提案プロトコルが IM の要求条件のいくつかを満たしていることが示された．残された要求条件には，即時性，すなわち，鍵交換の処理にどれだけのコストがかかるかという評価項目がある．そこで，次の代表的な同報通信プロトコルとの，通信量と計算量の観点からの比較を行う．

Needham-Schroeder（NS）[7] 鍵配送サーバを用いた古典的な認証プロトコル．サーバに対する機密性を満たしていないが，比較のために用いる．$\mathcal{S}$ は，全メンバとの間で秘密鍵 $k_1, \ldots, k_n$ を共有しており，イニシエータ $\mathcal{U}_1$ の要求に応じて同報鍵 k^* を作り，$\mathcal{E}_{k_1}(k^*), \ldots, \mathcal{E}_{k_n}(k^*)$ を送り返す．$j = 2, \ldots, n$ の各々について，$\mathcal{U}_1$ が暗号文 $\mathcal{E}_{k_j}(k^*)$ を渡すことで鍵を交換する．

El Gamal（EG）[8] 公開鍵暗号を用いて，イニシエータが他のメンバの各々の公開鍵で暗号化して同報鍵を個々に送信する方式．公開鍵暗号の代表として，DH と同程度の処理量である El Gamal 暗号を選んだ．$j = 2, \ldots, n$ の各々について，$\mathcal{U}_1$ が暗号文 $E_j(k^*)$ を渡すことで鍵を交換する．ここで，E_j は $\mathcal{U}_j$ の公開鍵 $y_j = g^{x_j}$ による暗号化関数を表す．

Ingemarsson-Tang-Wong（ING）[6] DH の最も自然な拡張．サーバもイニシエータもなく，リングになったメンバが隣り合ったメンバに繰り返し通信することで同報鍵 $k^* = g^{x_1 \cdots x_n}$ を確立する．たとえば，ラウンド 1 で，$\mathcal{U}_1$ から $\mathcal{U}_2$ へ g^{x_1} を送る．同時に，$\mathcal{U}_2$ も g^{x_2} を $\mathcal{U}_3$ へ送る．次のラウンドでは，U_1 は，受信した g^{x_n} に自分の秘密情報を冪乗し $x^{x_n x_1}$ を $\mathcal{U}_2$ へ送る．これを，$(n-1)$ ラウンド繰り返す．

Steiner-Tsudik-Waidner（STW）[5] ING は全メンバが同期している必要があり，現実的でない．そこで，一度に送る情報を増やすことにより，隣り合うメンバへ情報を伝搬していき，1 往復で鍵を確立するようにした方式．往路は，$\mathcal{U}_i$ が受け取った $\{g^{x_1}, \ldots, g^{x_1 \cdots x_{i-1}}\}$ の各々に，x_i を冪乗して $\mathcal{U}_{i+1}$ へ渡していく．復路は，$\mathcal{U}_{i+1}$ が受け取った情報の各々に自分の秘密情報 x_{i+1} を冪乗し，$\mathcal{U}_i$ の秘密情報が冪乗されていないものを次に渡す．たとえば，$n = 3$ の場合は，

$$\mathcal{U}_3 \to \mathcal{U}_2 : \{g^{x_3}, g^{x_1 x_3}\}$$
$$\mathcal{U}_2 \to \mathcal{U}_1 : \{g^{x_3 x_2}, g^{x_2}\}$$

のようになる．

このとき，各々の方法における同報鍵交換に生じる通信量と計算量を**表 3** に示す．ただし，リストメンバへ公開される鍵 y_i については考慮されていない．ここで用いられる記号は次のとおりとする．

c_1 ：共通鍵暗号の暗号化・復号の計算コスト
c_2 ：公開鍵暗号の暗号化（冪乗）の計算コスト
l_1 ：共通鍵のサイズ（e.g., 64 bit）
l_2 ：公開鍵のサイズ（e.g., 1,024 bit）
n ：イニシエータのリストメンバの人数

この評価より，通信量計算量の両方において，最も総処理量が小さいのが NS であるが，これはサーバの

表 3 鍵交換における通信量と計算量
Table 3 Communication and computation costs for key exchange.

	NS [7]	EG [8]	ING [6]	STW [5]	Proposed
rounds	$n+1$	$n-1$	$(n-1)n$	$2(n-1)$	$2n-1$
(communication)					
$\mathcal{U}_1 \to \mathcal{S}$	1	–	–	–	$l_2 \cdot 1$
$\mathcal{S} \to \mathcal{U}_1$	$l_1 \cdot n$	–	–	–	$l_2 \cdot n$
$\mathcal{U}_1 \to \mathcal{U}_2, \ldots, \mathcal{U}_n$	$l_1(n-1)$	$l_2(n-1)$	$l_2(n-1)$	$l_2(n-1)$	$l_1 n$
(computation)					
$\mathcal{U}_1$	c_1	$c_2(n-1)$	$c_2 n$	$c_2 2$	$c_2 + c_1(n-1)$
$\mathcal{S}$	$c_1 n$	–	–	–	$c_2(n-1)$
$\mathcal{U}_j$	c_1	c_2	$c_2 n$	$c_2(n+1)$	$c_2 + c_1$

機密性を満たさない．一方，EG，ING，STW はサーバを必要としないので機密性を満たした鍵交換プロトコルであるが，いずれも，グループメンバの数 n に比例した公開鍵の通信コストと計算コストが避けられない．これらに対して，提案プロトコルは，イニシエータ（$\mathcal{U}_1$）の通信コストと計算コストでは NS と同等であり，即応性が十分期待できる．計算コストで $n-1$ に比例する項があるが，その係数は共通鍵暗号の c_1 であり，それほど大きくならない．公開鍵の計算コストはサーバに加算されてしまっているが，オンライン状態にあるユーザの数は限られているという IM モデルの仮定がある．n は全ユーザの数ではなく，イニシエータのメンバリストのユーザの数であることに注意されたい．

3.7 サーバの不正に対する考察

本提案は，管理者によるメッセージの盗聴という通常考慮しないものを対象とした高い安全性を目指す試みである．それゆえ，管理者の他の不正についても安全性の検討を行う必要がある．ここでは，管理者が起こす可能性のある次の不正行為について，その危険性を評価する．

(1) Step 1-1 において，偽りの g_i を配布する．
(2) Step 1-3 において，偽りの y_i を配布する．
(3) Step 2-2 において，偽りの $b_j = z^{a_j}$ を配布する．
(4) ユーザの秘密情報 a_i を横流しする．

これらの情報はいずれも管理者が介する処理であるので，不正な管理者がここの処理を偽ることは可能である．しかしながら，本提案で想定しているインスタントメッセージングにおいては，利用者が不正を検出さえすれば，それ以降のメッセージ交換をやめればよいので，不正行為の検出が可能であることを示せば十分である．まず，偽りの g_i，y_i，b_j を送付した場合は，Step 2-3 の鍵交換に失敗して，メッセージの交換そのものができないので即ユーザが検出できる．仮に検出されないまま鍵交換のプロトコルを実行しても，それ以降のメッセージの復号化に失敗するだけなので，利用者の危険性は小さいと考える．次に，サーバが a_i を横流ししたり，他のユーザと結託したりしても，結局，利用者の秘密鍵 x_i が分からない限り盗聴はできない．また，Step 1-1 で極端に小さな $g_i << p$ を与えて y_i から秘密鍵 x_i を推測する攻撃も考えられるが，g_i のビット長を注意しておくだけで十分検出できるだろう．

以上の考察によって，想定した不正情報の配布による影響は問題にならないことが示された．

4. 実　　装

4.1 実装目的

提案プロトコルは実際に IM に適用することができ，そのときのオーバヘッドなどが実用上問題にならない程度に収まることの検証を，本実装の目的とする．

まず，鍵交換時のタイミングの課題を検討し，現実の IM との整合性やネットワーク障害などを考慮したシステム構築について述べる．次に，開発したクライアントソフトウェアと仕様を示し，その処理時間を報告する．

4.2 システム設計

4.2.1 鍵交換のタイミングの課題

IM の特徴的な点は，各エンティティのセッションへの参加と退出が非同期で頻繁に生じることである．しかも，エンティティとの間の複数の通信路を同時に確立する必要があり，単純なクライアントサーバ型のモデルでは説明できない．この独特のモデルが鍵交換に関係する問題を議論するために，**図 1** のタイムチャートを考えよう．

図 1 (a) は，A，B，C の 3 人のユーザがオンラインになっている時間を示している．まず，A が時刻 1 でオンラインになり，鍵共有プロトコルに従ってイニシエータとなって z_A を作る．時刻 2 と 3 で，各々 B

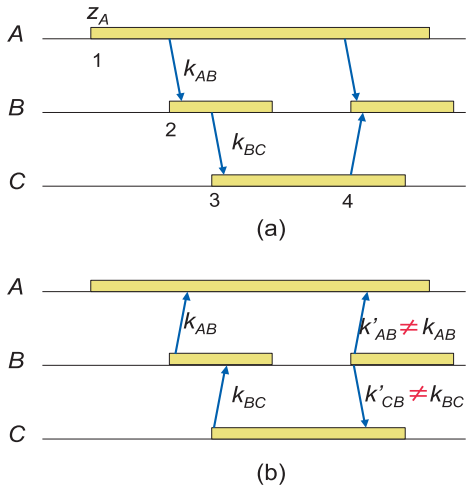

図 1 オンライン状況を示すタイムチャートと鍵交換のタイミング（A, B, C はユーザ，1, 2, 3, 4 は時刻，k_{AB}，k_{BC} はユーザ AB，BC 間で共有した秘密鍵を表す）

Fig. 1 A time chart for key exchange and state of on-line (We denote by users by A, B, C, time by $1, 2, 3, 4$, session keys shared between A and B by k_{AB}).

と C との間で鍵 k_{AB}，k_{BC} が共有される．しかし，時刻 4 の B のように，A がオンラインの間にいったんオフラインになって再び鍵共有が必要になる場合がある．このとき，B は k_{AB} を再利用するべきだろうか？ もしそうだとすると，その有効期限はいくらに定めるべきだろうか？ さらに複雑なことに，B と C の間の鍵交換についていうと，時刻 3 においては B がイニシエータであるにもかかわらず，時刻 4 では B と C の関係が逆になってしまう．このように，IM の鍵共有においては，コネクションを呼ぶクライアントとサーバの能動と受動の役割がつねに固定ではない．

そこで，本実装においては，オンラインになるときはつねにイニシエータとなって鍵共有プロトコルを実行することとした（図 1 (b)）．すなわち，後からオンラインになっている方が z_i を作る．たとえば，時刻 2 では，B がイニシエータになり A との鍵を新たに作る．同様に，時刻 4 では B がイニシエータである．たとえ同一の B がイニシエータとしても，時刻 2 と 4 とでは乱数 r が違うので鍵も異なる．このような実装にすることで鍵生成のオーバヘッドは増えるが，各々が stateless となり処理が軽量化される．さらに，提案プロトコルではイニシエータの処理はリストのサイズに依存しないという性質があるので，何度もイニシエータ処理を行ってもコストは小さくおさえられる．

4.2.2 システム構成

図 2 に本実装の構成要素を示す．ここで，U_1，U_2，U_3 はユーザ，y_1，y_2，y_3 は各々の公開鍵，m はメッセージで，$E(m)$ は暗号文を表す．

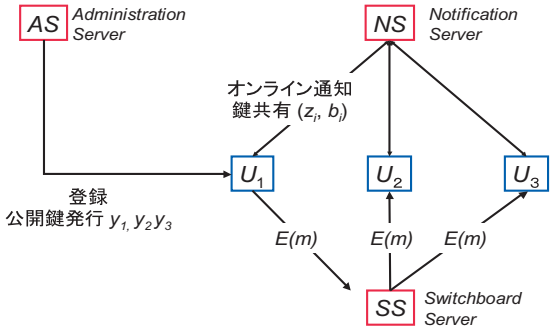

図 2 実装システム構成

Fig. 2 System architecture of the implementation.

AS（Administration Server） 新規ユーザの登録にともなう，ユーザ ID の管理，秘密情報管理，パスワード管理，公開鍵の発行を行う（server1）．

NS（Notification Server） ユーザがオンラインになるときの以下の処理を担当する．

（1） ユーザ認証

（2） プレゼンス通知（オンライン，オフライン状況をメンバリストに通知する）

（3） リスト交換マネージメント（公開鍵 y の配布，メンバリスト L_i の配布と追加削除の更新処理）（server2）

（4） 鍵共有（ユーザ i の z_i の保持，メンバリストのユーザ j のサインインにともなう b_j の送信）（server3）

（5） 呼び出し（メッセージ交換のための SS の割当て）

SS（Switchboard Server） ユーザ間のメッセージの中継と他のメンバへのマルチキャストを実行する（server4，server5）．

クライアントは NS と SS との間に通信路を結び，直接他のクライアントとは通信しない．複数ユーザ間でのチャットの場合でも，SS がメンバの各々にメッセージを複製して伝送する．このような構成になっていると，ユーザ数の増加にともなって SS への負荷が集中して過負荷になることが懸念されるが，メッセージの暗復号化処理は各クライアント間で行われるので暗号化処理の負担はなく処理は軽い．しかも，NS と SS が分離しているので複数の SS へ処理を分散させることで負荷の集中を避けることができる．

4.2.3 通信プロトコル

IM の特徴は限定されたメッセージ長をリアルタイムに交換する点にある．そこで，多少効率は悪くても開発の容易な，ASCII によるテキストベースの通信

表 4 NS のデータベース
Table 4 Database at NS.

属性	値
data	g, q, p
user	ID の最終シリアル番号
ID_i	a_i, y_i
$pass_i$	U_i のパスワード
$mlist_i$	メンバリスト L_i
s_i	オンライン U_i の IP, port
$chatlist_i$	ID_i のチャットリスト

表 5 クライアント U_i のデータベース
Table 5 Database at client $\mathcal{U}_i$.

属性	値
userdata	$ID_i, x_i, g, p, q, g_i, y_i$
mlist	L_i
chatkey	チャットメンバとの共通鍵
y_j	U_j の公開鍵
key_j	U_j との共通鍵

プロトコルを採用した．すべてのメッセージは，インターネットで広く採用されている CR と LF で終端される行単位ベースで行われる．暗号化処理で生じるバイナリデータについては，16 進数の文字列に符号化して転送する．NS や SS とクライアントとの通信には，毎回使い捨ての TCP コネクションを確立する．このような構成にすることにより，チャットの際に，一部のメンバの障害が全体に及ぼす影響を最小化することを期待できる．

クライアントは，サーバからのオンラインの通知がいつ届くか分からないので，ユーザからの入力と交互に処理をしていてはデッドロックに陥ってしまう．そこで，Java のマルチスレッドを利用し，サインインと同時に受信用のスレッドを用意して，ユーザからの入力とは並列に処理を行うことで解決した．

4.2.4 リスト管理・鍵管理

メンバリストやユーザ属性情報をどこで管理するかは実装に依存する．MSN Messenger では，クライアントが複数の PC からサインインすることを想定し，すべてサーバで管理している[14]．本実装では，サーバとクライアントで独立に管理して手動で同期をとっているが，リスト更新のためのプロトコルを用意して集中管理することを計画している．

サーバ S と U_i のクライアントとが管理するデータを各々**表 4** と**表 5** に示す．

4.2.5 認証プロトコル

IM のモデルでは，つねにサーバを経由して通信が行われるため，サーバを信頼点としてサーバに対する一般的な認証を実行すればよく，技術的な難易度はない．たとえば，MSN Messenger では，セキュアハッシュ関数として MD5 を用いたチャレンジアンドレスポンスを採用している[14]．本実装では未対応であるが，登録の際に公開鍵を提出しているので，それを用いたディジタル署名も利用することができる．

表 6 実装システム技術仕様
Table 6 Technical specification of the implementation.

項目	値
公開鍵（提案プロトコル）	1,024 bit (p)，160 bit (q)
共通鍵/モード/パディング	DES/CBC/PKCS#7
開発言語	Java SDK v.1.3，Cryptix 3.2.0
認証	パスワード認証
ネットワーク環境	数 kbps から数 Mbps の LAN
通信プロトコル	TCP/IP によるソケット通信

図 3 クライアント実行例
Fig. 3 Client software.

4.3 実装システム

実装システムは，**表 6** の技術仕様に従い，2 章で提案したプロトコルを実行するプログラム群によって実現されている．本システムは次の一連の動作を実行する．

(1) ユーザ $\mathcal{U}_i$ はサーバ（AS）との間で，登録プロトコル 3.3 を実行し，データベースを初期化する．
(2) クライアントは，サインインによってサーバ（NS）へユーザ認証を行い，Step 2-1 に従って鍵交換のための情報を提供する．
(3) サーバ（NS）は，この時点で，他のオンラインのユーザの各々へ $\mathcal{U}_i$ がオンラインになったことを通知し，Step 2-2 を実行して鍵を交換する．
(4) メッセージは，共有された鍵で暗号化され，サーバ（SS）を経由して交換される．復号化は各クライアントで行われる．

図 3 に，開発したクライアントの実行画面を示す．本クライアントには，受信メッセージや他のユーザの

表 7 平均処理時間
Table 7 Average processing time.

エンティティ	プラットフォーム	平均 [ms]	標準偏差
クライアント 1（イニシエータ）	Windows 2000, Pentium 4, 1.7 GHz	156.0	20.5
サーバ	Linux RedHat 6.2, Pentium III, 730 MHz	725.8	49.6
クライアント 2	Windows 2000, Pentium 4, 1.7 GHz	25.1	15.2

オンライン状況を示す受信用ウィンドウ（上），送信する暗号化メッセージを入力するウィンドウ（下），および，送信やリセットを実行するボタンが用意されている．図上部の SAVE，LOAD のボタンは，交換したメッセージのファイルへの保存と読み込みを実行する．この実行例では，ユーザ 16 が今オンラインになり，“Hello How are you?” というメッセージが届いたところを示している．

4.4 実装による評価

実装システムの暗号化および通信によるオーバヘッドを次のように測定した．**表 7** に示される仕様のクライアント 2 台，サーバ 1 台を用いて，クライアントサーバ間をパケットの平均往復伝達時間 17.7 ms のネットワーク環境で結び，10 回鍵交換を実行してその平均時間とその標準偏差を求めた．クライアント 2 を先にオンラインにしておき，クライアント 1 が後からサインインして鍵確立までの処理時間を求めている．

イニシエータ（クライアント 1）は，乱数を作って g^r をサーバへ送り，その後で y_2^r から鍵を計算しなくてはならない．一方のクライアント 2 は，この後者の仕事だけでよく，実際の処理時間でも 6 倍程度の差が生じていることが示され，このことを裏付けている．サーバでの処理時間が比較的長いが，このほとんどは，オンライン状態の他のユーザの検索とソケット通信のコストであり，実際ほとんど遅延のない LAN 環境ではこの時間は短縮された．メンバリストのメンバ数が増えても，クライアント 1 の処理時間はほとんど変わらず，サーバの処理時間のみに影響する．

以上の考察により，暗号化による処理時間の増大はクライアントあたり 1 秒未満であり，十分実用的な利用性を得ることが示された．

5. おわりに

中継するサーバがメッセージをいったん復号化することなく，複数クライアント間でメッセージを暗号通信することを特徴とするセキュアインスタントメッセージングプロトコルを提案し，既存のグループ鍵共有プロトコルとの比較評価を示した．インスタントメッセージ固有の，非同期でメンバが動的に変化する同報通信の課題に対して，イニシエータのコストを小さくするプロトコルが有効であった．提案プロトコルが正しく動作することを，試験実装システムの上で検証した．実装したクライアントシステムは，標準的な環境のもと，1 秒以内で鍵確立を実行可能であり，目標であったリアルタイムでの暗号通信を十分に実現することを確認した．

今後の課題には，リスト管理や利用者登録のためのプログラムを充実させて，システムの完成度を上げること，現在は独自プロトコルによる通信を標準化プロトコルに置き換えて，既存のコンシュマーインスタントメッセージングサービスとの互換性実現すること，サーバ集中型ではなく P2P などの分散型モデルでのセキュアな IM プロトコルを研究することなどがある．以上の課題をクリアしたうえで，より大規模な環境での実運用を行い，現実のネットワーク環境での利用の中がどのような問題を生じさせるか明らかにしていきたいと考える．

謝辞 本研究の遂行にあたって多大な協力をいただいた東海大学工学部電気工学科秋山曜佑氏，松木佑太氏に感謝する．

参考文献

1) Nielsen//NetRatings. インスタントメッセンジャー調査レポート. http://www.netratings.co.jp/im.html（2002 年 9 月参照）.
2) Diffie, W. and Hellman, M.: New Directions in Cryptography, *IEEE Trans. Inf. Theory*, Vol.22, No.6, pp.644–654 (1976).
3) 2002 年インターネット白書，Impress (2002).
4) Marsan, C.D.: Secure instant messaging software coming for corporate users, *Netwok World Fusion* (2002), available at http://www.nwfusion.com/news/2002/128936_01-14-2002.html（2002 年 10 月参照）.
5) Steiner, M., Tsudik, G. and Waidner, M.: Diffie-Hellman Key Distribution Extended to Group Communication, *Proc. CCS'96*, pp.31–37 (1996).
6) Ingemarsson, I., Tang, D. and Wong, C.: A Conference Key Distribution System, *IEEE Trans. Inf. Theory*, Vol.28, No.5, pp.714–720 (1982).

7) Needham, R.M. and Schroeder, M.D.: Using encryption for authentication in large networks of computers, *Comm. ACM*, Vol.21, No.12, pp.993–999 (1978).
8) ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inf. Theory*, Vol.31, No.4, pp.469–472 (1985).
9) 菊池，多田，中西：インスタントメッセンジャーにおける鍵配送プロトコルの一提案，2002 年暗号と情報セキュリティシンポジウム（SCIS 2002），Vol.II, pp.961–965 (2002).
10) 菊池，秋山，多田，中西：セキュアインスタントメッセージングシステムの実装，コンピュータセキュリティシンポジウム 2002（CSS2002）論文集，pp.401–405 (2002).
11) Herbsleb, J., Atkins, D., Boyer, D., Handel, M. and Finhold, T.: Introducing Instant Messaging and Chat in the Workplace, *Proc. SIGCHI Conference on Human factors in computing systems: Changing our world, changing ourselves* (April 2002).
12) Saint-Andre, P.: Jabber Technology Overview (Mar. 2001), available at http://docs.jabber.org/general/html/overview.html.
13) IMUnified Webpag, http://www.imunified.org/（2001 年 12 月参照）.
14) Movva, R. and Lai, W.: MSN Messenger Service 1.0 Protocol, Internet Draft (Aug. 1999). draft-movva-msn-messenger-protocol-00.txt
15) Day, M., Rosenberg, J. and Sugano, H.: A Model for Presence and Instant Messaging, Internet RFC 2778 (2000).
16) Day, M., Aggarwal, S., Mohr, G. and Vincent, J.: Instant Messaging/Presence Protocol Requirements, Internet RFC 2779 (2000).

(平成 14 年 11 月 29 日受付)
(平成 15 年 6 月 3 日採録)

菊池　浩明（正会員）

1988 年明治大学工学部電子通信工学科卒業．1990 年同大学院博士前期課程修了．1990 年（株）富士通研究所入社．1994 年東海大学工学部電気工学科助手．1995 年同専任講師．1999 年同助教授，1997 年カーネギーメロン大学計算機科学学部客員研究員．2000 年東海大学電子情報学部情報メディア学科助教授，現在に至る．博士（工学）．ファジィ論理，多値論理，ネットワークセキュリティに興味を持つ．1990 年日本ファジィ学会奨励賞，1993 年情報処理学会奨励賞，1996 年 SCIS 論文賞．電子情報通信学会，日本ファジィ学会，IEEE，ACM 各会員．

多田美奈子（学生会員）

2002 年東海大学工学部電気工学科卒業．現在，同大学院工学研究科博士前期課程在学中．暗号セキュリティの研究に従事．電子情報通信学会会員．

中西祥八郎

1967 年東海大学工学部電気工学科卒業．1969 年同大学院博士前期課程修了．同年同大学工学部電気工学科助手．1971 年同専任講師，札幌校舎勤務，1973 年同湘南校舎勤務．1985 年同助教授．1991 年同教授，2000 年同電子情報学部情報科学科教授，現在に至る．工学博士．日本ファジィ学会，電気学会，計測自動制御学会，システム制御情報学会，日本神経回路学会，日本経営工学会，IEEE，IFSA 各会員．