L-029

JXTA による暗号化マルチキャスト通信

Encrypted Multicast Transport in JXTA

田島 陽平† 小林 孝史‡
Yohei Tajima Takashi Kobayashi

1. はじめに

近年，コンピュータウィルスによる個人情報の流出や機密情報の漏洩などが増え，コンピュータにおけるセキュリティが非常に重要視されている．P2P モデルにおけるネットワーク通信においても例外ではなく，各ノードでの通信が行われるため，その時にクライアントサーバモデルと同様の問題が起こる可能性がある．また，ピュア型の P2P モデルの場合には，P2P ネットワークの管理が不可能なため，情報の流出の歯止めをかけることができず，さらなる被害をもたらす可能性がある．

本研究では，P2P モデルでのマルチキャスト通信の盗聴を防ぐ方法を模索するため，オープンソースである P2P フレームワーク JXTA [1]を用いて，公開鍵暗号による暗号化通信を実装しセキュリティを高める実験を行う．

JXTA におけるマルチキャスト通信時に，ユーザの送信するデータとは無関係に暗号化通信を行うことを提案する．マルチキャストによる通信は，LAN 内の PC から盗聴することが可能である．それゆえ常に暗号化通信を行うことが重要となる．よって，JXTA のアーキテクチャに変更を加えることで，マルチキャスト通信での暗号化を常時行うべきである．

2 節では，本研究において用いたマルチキャストによるファイル通信システムの仕組みを説明する．

3 節では，提案する暗号化通信について説明し，4 節でそれを用いた実験による評価を行う．

2. JXTA を利用したマルチキャストファイル転送システム

2.1 マルチキャストファイル転送

本研究では，JXTA でのマルチキャスト通信を行うため，JXTA ソフトウェアアーキテクチャで上位に位置する JXTA shell[2]で動作するファイル転送システムを開発した．JXTA でマルチキャスト通信を行うには Propagate Pipes[3]を作成してマルチキャストグループに属する必要があるが，JxtaMulticastSocket を用いることで，容易に Propagate Pipes を作成しマルチキャストグループに属することができる．そこで，JXTA shell 上で入力したコマンドにより JxtaMulticastSocket を利用したマルチキャストによるファイルの送受信を行うことができるコマンドを開発した．

2.2 システム仕様

本研究で開発したマルチキャストファイル転送システムの仕様は次の通りである．

(1) JXTA shell のコマンドにより，マルチキャスト通信を行うための Propagate Pipes を作成し，特定のマルチキャストグループに属する．
(2) それぞれのピアにおいて，マルチキャストグループに属している他のピアの ID リストを作成し，ピア間で非同期による共有を行う．
(3) 送信ピア側で送信するファイルを指定すると，ファイルを特定のサイズに分割し，そのパケットを送信ピアが属するマルチキャストグループに向けて送信する．
(4) 受信ピア側でパケットを受信した時，受信ピア内に同名のファイルがないかを確認する．もし同名のファイルがある場合には，更新時間の古いファイルを削除する．
(5) 受信ピア側は，パケットを正常に受信できたかをハッシュ値を用いることで確認する．
(6) 受信ピア側でパケットを正常に受信できた場合，パケットをキャッシュとして保存し，ファイルの再構成を行う．
(7) パケットを正常に受信できなかった場合や受信ピア側にパケットそのものが届かなかった場合には，受信ピアは送信ピア，もしくは受信ピアと同じマルチキャストグループに属し，ピア ID リストにあるその他のピアに対して再送要求を行う．

以上の方法により，再送要求による送信ピアの負荷を所属するマルチキャストグループのピアに対して分散させ，信頼性を確保したマルチキャストによるファイルの送受信をおこなうことができる．

3. 提案方式

2 節での JXTA Shell でのマルチキャストによる通信において利用した JxtaMulticastSocket クラスが記述されている JxtaMulticastSocket.java において，次のように変更を加える．

(1) 送信ピアでファイルの送信が行われる前に，秘密鍵・公開鍵のペアを作成する．
(2) 作成した秘密鍵を用いてデータの暗号化を行う．
(3) 対となる公開鍵を暗号化データに添付する．
(4) マルチキャストグループに向けてデータを送信する
(5) 受信側のピアにおいて，受信データから公開鍵を取り出す．
(6) 公開鍵を用いて，暗号化データを復号する．
(7) 鍵の取得を失敗した場合，再送要求を送信ピア，もしくはマルチキャストグループに属し，ピア ID リストにあるその他のピアに行う．

以上の変更により，ユーザの送信するデータとは無関係に暗号化通信が行うことができる．

4. 実験

4.1 実験 1

暗号化では，暗号化をするために相応の時間が必要であり，暗号強度と暗号化されるデータによって変化する．そこで最初の実験として，暗号強度と送信時間との関係を調べた．実験方法として，100KByte のランダムテキストデータを準備する．そして暗号強度が 512bit 以上で 6 種類の

† 関西大学大学院 総合情報学研究科 知識情報学専攻
‡ 関西大学 総合情報学部

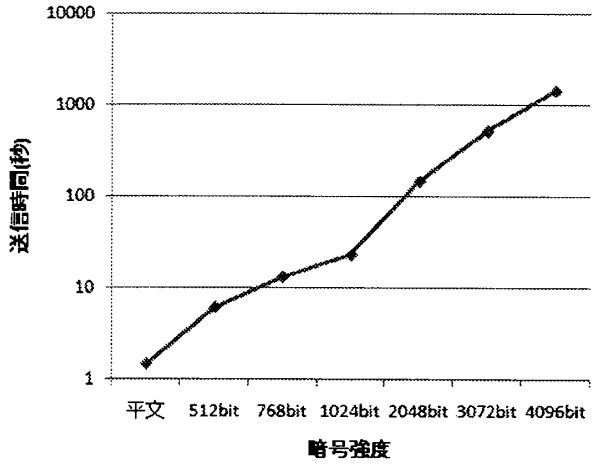

図 1　暗号強度と送信時間との関係

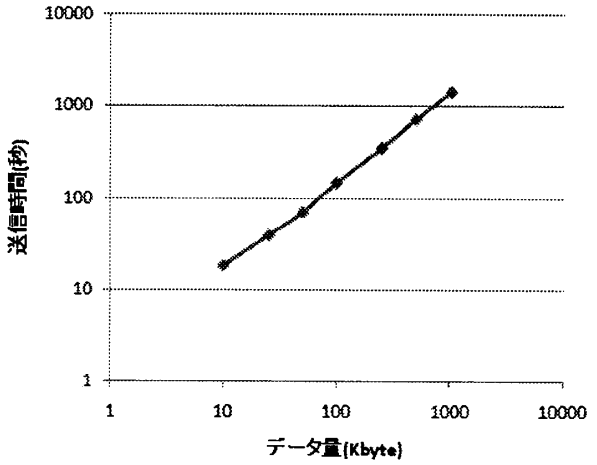

図 2 送信データの長さと送信時間との関係

異なる暗号化を用いたマルチキャスト通信を行い，通信が完了するまでの時間を 50 回計測して，それぞれの平均時間を算出した．

なお，比較検証を行うために，同じ 100Kbyte のランダムテキストデータを暗号化せずにマルチキャスト通信を行い，暗号化通信時と同様に，通信が完了するまでの時間を 50 回計測し，その平均時間の算出を行った．

図 1 は，実験 1 によって求められた送信時間の平均値と暗号強度との関係を元に作成したグラフである．

平文での平均送信時間はおよそ 1.5 秒だったものの，暗号強度を 512bit とした時では約 4 倍かかった．さらに 1024bit では約 15 倍，4096bit では送信に約 965 倍もの時間を必要とした．

4.2 実験 2

次に，暗号化されたデータの長さと送信時間との関係を調べるため．暗号強度を一定にして実験を行った．実験方法は，送信データとして 10Kbyte から 1Mbyte までの 7 種類のデータを用意し，暗号強度を 2048bit に設定した暗号化通信が完了するまでの時間を 50 回計測し，その平均時間の算出を行った．

図 2 は，実験 2 の結果から送信データの長さと送信時間との関係を示したグラフである．10Kbyte のデータの送信時間は約 18 秒であったが，100Kbyte では約 147 秒，1Mbyte のデータを送信するには約 23 分必要だった．なお，送信データの長さと送信時間との相関係数を求めると 0.999963 であった．このことからデータの送信時間とは送信するデータの長さは高い正の相関があることがわかる．

5. 考察

実験 1 から，暗号化通信にかかる送信時間は暗号強度に依存しているということがわかるが，2005 年 11 月 2 日に行われた RSA Security Inc. 主催の『RSA Challenge 解読コンテスト』において，RSA-640(640bit) の解読が行われたため，現在 RSA の暗号強度は 1024bit 以上が必要であるとされている．また，2010 年までには 1024bit の暗号が解読されるという予測がされている．以上のことを考えると，暗号化には 2048bit での暗号化が必要であり，平文での通信と比べて約 15 倍以上の大幅な通信時間の遅延は，安全な通信を行うために必要である．

実験 2 では，暗号化に用いる暗号強度が等しいならば，暗号化が通信の遅延する要因とは成りうることがないことがわかる．また，暗号化を行った場合には暗号化されたデータの長さが暗号化される前のデータの長さによって増減するが，暗号化に伴うデータの増減によって，通信時間が変化する要因とならないことがわかる．

6. おわりに

本研究では JxtaMulticastSocket による IP マルチキャストによる通信実験を行った．マルチキャストには IP マルチキャスト以外にアプリケーションレベルマルチキャスト(ALM)があるが，JXTA では IP マルチキャストによる通信方法のみが実装されている．そのため，今回の実験の様な小規模のプライベートネットワーク内でのマルチキャスト通信では問題ないが，マルチキャストルーティングが必要となる大規模ネットワークでは，JXTA を実装しているピア以外に対しての手間と設備コストといった負担がかかるという問題が挙げられる．ALM ではアプリケーション層等の上位層にマルチキャストの機能を実装しているため，JXTA 単独でのマルチキャスト通信を行うことができる．そのためピア間での最短経路による通信や，スケーラビリティの向上，信頼性・輻輳性の向上を図ることができる．今後は JXTA での ALM によるファイル転送システムの実装を行うことを考えている．

参考文献

[1] JXTA, http://www.jxta.org/, 2001.

[2] Brendon J Wilson, “JXTA のすべて－P2P Java プログラミング”, 日経 BP 社, 2003.

[3] Joseph D. Gradecki, “Mastering JXTA: Building Java Peer-to-Peer Applications”, John Wiley & Sons Inc, 2002.