ISSN 0917-3668
2000. 8. 1

KUINSニュース

No. 33

京都大学学術情報ネットワーク機構
http://www.kuins.kyoto-u.ac.jp/

第 6 回京都大学高度情報化フォーラムを開催
撮影: 亀田能成 (総合情報メディアセンター)

目　　次

第 6 回京都大学高度情報化フォーラムを開催

第 6 回京都大学高度情報化フォーラムが，学術情報システム整備委員会技術専門委員会，大型計算機センター，総合情報メディアセンター，大学院情報学研究科の主催で，平成 12 年 7 月 3 日 (月) に附属図書館 3 階 AV ホールにおいて開催されました.

今回は,「インターネット社会におけるセキュリティ問題と対策」をテーマとし，本学全構成員を対象にインターネットセキュリティについて，学内における対策への意識を推進するために開催されたものです.

当日は，セキュリティ問題が現代社会に及ぼす影響，大学におけるセキュリティ対策の問題点に関する講演が行われ，具体的なセキュリティ対策の概要について紹介がありました．また，KUINS におけるセキュリティ対策の現状や各部局における問題とその対策について報告があり，最後に安全性を重要視した次世代キャンパスネットワーク (KUINS-III) の概要について紹介されました.

約百数十名の参加者を得て，活発な質疑応答や意見交換が行われ，インターネットのセキュリティ問題への高い関心とその対策の重要性を改めて認識させられました.

なお，シンポジウム当日の様子を次の URL より学内向けに公開しております.

http://lawn.imel.kyoto-u.ac.jp/SpecialEvent/2000/security-j.html

コンテンツの整備にご尽力いただいた総合情報メディアセンターの亀田先生らに感謝いたします.

スパムメール不正中継対策フィルタの設定について

学術情報ネットワーク機構事務室

平成 11 年 11 月に実施した「スパムメール不正中継対策に関するアンケート調査」の各部局からの回答に基づき，日程，設定内容等を調整の上，希望のあったサブネットについては 4 月中旬から 6 月中旬までに設定を終了しました.

この結果，スパムメール不正中継対策のフィルタを設定する前と比較して，本学のコンピュータがスパムメールを不正に中継させられるという事件の発生頻度は減少していますが，完全に防止できているわけではありません.

一部のサブネットからはフィルタ設定希望の回答がなかったため，いまだに対策フィルタの設定を行っておりません．このため今後もスパムメールの不正中継が発生する可能性があります．フィルタ未設定のサブネットで，フィルタ設定のご希望がある場合，本機構情報システム管理掛 (電子メールspamfilt@kuins.kyoto-u.ac.jp) までご相談願います.

なお，対策済みメールサーバであっても，OS やメールサーバ用ソフトウェア等の再インストールのさいに不正中継対策の設定が無効になっていて，スパムメールを不正中継してしまった例が何件か発生していますのでご注意願います.

KUMX の運用について

学術情報ネットワーク機構

本号の報告にありますとおり，KUINS では 2000 年 4 月よりスパムメール不正中継対策を順次実施しております．今後は，事情により対策が間に合わずフィルタリングが実施できない，あるいは不完全な設定のまま対策済と申請してしまったなどの原因で，不正中継の踏み台となってしまったホストに対しては，入口ルータにおいて学外からの通信を遮断する場合があります．この他にも，故障や OS の入れ替えによって申請済のメールサーバが利用できなくなるなどの理由で，サブドメイン宛のメールが届かなくなる事態も想定されます．

このように，登録されたメールサーバが利用不可能となる場合，現状の設定では，各サブドメインのユーザ宛に届けられるはずのメールが学外のサーバに滞留することになります．このこと自体はメール配送システムの正しい動作であると言えますが，未配メールの滞留によるディスク領域の圧迫など，学外のサーバ運用に多かれ少なかれ悪影響を及ぼしてしまいます．また，一定期間を超えると滞留したメールが送信者にエラーメールとして返送されてしまうなど，好ましくない状況が発生します．

このような状況に対応するため，KUINS では，学外からのメールを各サブドメインにバックアップ中継するメールサーバ (KUMX) の運用を開始いたします．DNS データベースにおいて，サブドメインの弱い MX を KUMX に向けることで，上位の MX が利用不能の場合に代ってメールを受け取り，(上位 MX の復旧後に) 配送することができます．

KUINS では，大型計算機センターと総合情報メディアセンターの協力を得て，複数台のスパム不正中継対策済サーバを KUMX として準備しました．管理の都合上，`kyoto-u.ac.jp` 直下のサブドメインに対する設定のみを対象としておりますが，KUMX の利用をご希望の場合には，`spamfilt@kuins.kyoto-u.ac.jp` 宛の電子メールで申請してください．

また，事情によっては，例外的にプライマリ MX を引き受けることもありますので，ご相談ください．ただし，不正中継対策済のメールサーバを早急に立ち上げていただくことを約束していただきます．

また，KUMX の運用管理にご協力いただける方 (部局) がありましたら，同様に上のアドレスまでお知らせ下さい．

サブネット単位でのフィルタリングについて

学術情報ネットワーク機構

2000 年 4 月から KUINS が実施しておりますスパムメール不正中継対策では，各サブネットの入口ルータにおいて，特定ホスト以外にむけた学外からの SMTP (25/tcp) 接続を遮断するフィルタを設置しております．

各サブネットにおけるセキュリティレベルを高めるため，SMTP 以外のプロトコルについてもフィルタの設置を希望される場合には，関連する利用者に周知の上，サブネット単位でおとりまとめいただき，`spamfilt@kuins.kyoto-u.ac.jp` までご相談ください．一般にフィルタリング規則が複雑になると，ルータの性能が低下しますのでご注意ください．

SSH (Secure SHell) の利用[1]

赤坂浩一，平野彰雄，沢田篤史 （大型計算機センター）

1. はじめに

ネットワークを利用して外部から計算機を利用する場合に telnet などを使ってログインしますが，この時，端末から入力された ID とパスワードはそのまま平文 (暗号化されない文字列) として，通信経路に流されています．その通信経路のすみからすみまでに目が行き届けば良いのですが，途中の経路に悪意を持った攻撃者が存在すれば，盗聴により ID やパスワードを知ることも可能です．rlogin, rsh を用いている場合はより深刻で，IP アドレスの偽造や DNS の改竄といったなりすまし攻撃に対して，パスワード無しでのログインすら許すことになりかねません．

本稿で紹介する SSH (Secure SHell) を利用することにより，通信データ (ID やパスワードを含めて) を暗号化して，安全に計算機を利用することが可能になります．SSH にはバージョン 1 と 2 の二種類があり，バージョン 2 の方にはより強度の高い暗号が用いられています．

このたび KUINS では，京都大学における非営利目的利用に限定した SSH バージョン 2(以下 SSH2) のサイトライセンスを取得し，学内向けにダウンロードを可能としました．そこで本稿では，これらのインストール方法と簡単な利用方法について解説します．利用にさいしてはライセンス条項をよくお読みのうえ，違反することのないようにご注意ください[2]．

2. SSH とは

SSH は，ネットワークに接続された二つのホスト間に安全な通信経路を提供するプログラムです．強力な認証機能で IP アドレスの偽装などを防ぎ，通信データを暗号化することで内容が盗聴されないようにできます．SSH はクライアントサーバ形式のプログラムですので，利用するにはリモート・ローカルの双方に SSH を導入する必要があります．

SSH では，ホストとユーザの認証に DSA (バージョン 1 では RSA) の認証プロトコルを利用します．DSA の認証は公開鍵暗号方式で，「公開鍵」と「秘密鍵」の二つの鍵を使います．「公開鍵」は復号することが困難なので，通信経路にそのまま流しても大丈夫です．だたし，暗号化・復号化に要する処理が複雑で，処理に時間がかかるという問題があります．

この問題を解決するため，SSH では公開鍵暗号方式と共通鍵暗号方式を組み合せて用います．安全な通信セッションを始めるにあたっては，ホストとユーザの認証が不可欠ですが，そこには上述のように DSA を利用します．そこで信頼のおけるアクセスであることが確認されれば，その後の通信では使い捨ての「共通鍵」を生成し，それを利用して暗号化・復号化が行われます．

---

[1] 本稿は，京都大学大型計算機センター広報, Vol. 32, No. 5, pp. 226–237 (1999.10) の解説記事に加筆修正を行ったものです．

[2] KUINS で取得したライセンスは，京都大学構成員の非営利目的利用に限られています．たとえ京都大学の教職員や学生であっても副業やアルバイト目的に用いることはできません．また，学外への二次配布は行わないでください．

共通鍵による暗号方式では暗号化・復号化に要する処理が比較的簡単であるため，オーバヘッドを最小限にすることができます．この共通鍵が盗聴されると通信の内容も暴露されてしまうという危険がありますが，セッション毎に使い捨ての鍵を用い，それを公開鍵暗号方式で交換することで，その可能性を低くしています．

3. UNIX での利用方法

SSH は多くの UNIX ベースの OS で動作が確認されています．ここでは，SSH2 のインストール方法と SSH2 クライアントの利用方法について解説します．

3.1 ソースコードの入手とインストール

SSH2 のソースコードと京都大学向けのライセンスファイルは，次の Web ページから入手することができます．このページには学内からのみアクセスできます．

```
http://www.kuins.kyoto-u.ac.jp/download/
```

本稿執筆時点の 2000 年 7 月現在，バージョン 2.2.0 一式が置いてあります．今後もバージョンアップがあれば最新のものを置くように準備しております．

ソースコード一式 (ファイル名 ssh-2.2.0.tar.gz) をダウンロードし，適当なディレクトリで展開します．

```
% gzip -dc ssh-2.2.0.tar.gz | tar -xvf -
```

ssh-2.2.0/のディレクトリが作られますので，そこに移動し，README ファイルを読みます．

```
% cd ssh-2.2.0
% less README
```

またライセンスの条項は LICENSE に記述されていますので，必ず目を通しておいてください．

```
% less LICENSE
```

つづいてコンパイルします．gcc などが必要な場合はあらかじめインストールしておきます．

```
% ./configure
% make
% su
# make install
```

これで SSH2 のサーバ，クライアント，オンラインマニュアル一式が/usr/local/{sbin/,bin/,man/}の下にインストールされ，同時に次のホスト鍵と設定ファイルが /etc/ssh2/ の下に生成されます．

- ssh2_config — SSH2 クライアントの設定ファイル
- sshd2_config — SSH2 サーバの設定ファイル
- hostkey — ホスト鍵 (秘密鍵)

- hostkey.pub — ホスト鍵 (公開鍵)
- ssh_dummy_shell.out — ダミーシェルの出力メッセージ

以上でインストールは完了です．お使いの計算機で SSH2 サーバを起動しない場合は，次項のサーバ設定を行う必要はありません．

3.2 サーバの設定

サーバの設定は，/etc/ssh2/sshd2_config ファイルで行います．インストール時に生成されたファイルのままでも通常の運用は可能ですが，次の項目については運用上変更する必要が生じるかもしれません．

- SSH1 サーバとの互換性
  Ssh1Compatibility を yes にし，Sshd1Path に sshd1 のパス名を設定します．
- DSA ホスト認証 (.rhosts，hosts.equiv の利用)
  AllowedAuthentication に hostbased を加えます．

その他のパラメータの詳細に関してはオンラインマニュアルsshd2(8)，sshd2_config(5) を参照してください．

設定ファイルの変更が終わったら，SSH2 サーバ (sshd) を立ち上げます．OS の起動時に立ち上がるようにするためには，/etc/rc* あたりのファイルを変更する必要が生じるでしょう．ご利用の OS に合わせて設定してください．

3.3 クライアントの設定

SSH2 クライアントの設定は，/etc/ssh2/ssh2_config に定義されています．ここでは，計算機で利用する SSH2 クライアントに共通の定義を書いておきます．サーバの設定と同様，インストール時に生成された内容のままで通常の運用は可能です．

ここに書かれた設定内容は，各ユーザがホームディレクトリ (~/.ssh2/ssh2_config) で上書きすることができます．各パラメータの詳細に関しては，ssh(1)，ssh2_config(5)，sshd(8) のマニュアルを参照してください．

3.4 クライアントの利用

ここまではシステム管理者向けの設定方法の解説でしたが，ここからやっと一般ユーザ向けの利用方法の解説に入ります．

SSH2 がインストールされたシステムでは，クライアントプログラムとしてslogin，ssh，scp，sftp の等コマンドが使えるようになります．これらのコマンドはそれぞれ，rlogin，rsh，rcp，ftp の代わりに使用することができます．

これらのコマンドを使ってお使いの計算機 (ホスト名: local) からリモートの計算機 (ホスト名: remote) にアクセスするための準備のに必要な作業は次の通りです．

1. まずは，ユーザであるあなた自身を認証するための鍵を生成しなければなりません．鍵

の生成には，local においてssh-keygen というコマンドを実行します．

```
local% ssh-keygen
Generating 1024-bit dsa key pair
   1 oOo.oOo.o
Key generated.
1024-bit dsa, username@local, Wed Jul 19 2000 02:27:43
Passphrase :
Again      :
Private key saved to /home/username/.ssh2/id_dsa_1024_a
Public key saved to /home/username/.ssh2/id_dsa_1024_a.pub
```

ssh-keygen は，新たな鍵に対するパスフレーズを入力するよう要求します．二回同じフレーズを入力したら，ホームディレクトリ直下に.ssh2/ というディレクトリができ，その下に鍵を記録したファイルが生成されます．id_dsa_1024_a が秘密鍵，id_dsa_1024_a.pub が公開鍵です．秘密鍵とパスフレーズは決して他人に知らせてはいけません．

2. 次に local の ~/.ssh2/ ディレクトリにidentification というファイルを作ります．

   ```
   local% cd ~/.ssh2
   local% echo "IdKey id_dsa_1024_a" > identification
   ```

   これにより，ホスト local のユーザであるあなた自身を示す鍵が手順 1 で作った id_dsa_1024_a に収められていることが宣言されます．

3. remote でも手順 1 と同様の作業を行います．入力するパスフレーズは local のものと異っていても構いません．local，remote の相互でアクセスし合う場合などは，remote で手順 2 も行う必要があります．この作業が盗聴されたら元も子もありませんので，安全な環境で行ってください[3]．

4. local で作成した (手順 1) 公開鍵 (id_dsa_1024_a.pub) を remote の~/.ssh2/ に適当な名前 (例えば local.pub) でコピーします．

   ```
   local% scp ~/.ssh2/id_dsa_1024_a.pub remote:~/.ssh2/local.pub
   ```

5. remote の ~/.ssh2/ にauthorization というファイルを作ります．

   ```
   remote% cd ~/.ssh2
   remote% echo "Key local.pub" >> authorization
   ```

   これにより，~/.ssh2/local.pub というファイルが，local というホストのユーザであるあなたの公開鍵であることが宣言されます．

これでやっと local から remote へ SSH2 を使ってログインすることができるようになりましたので，さっそく試してみます．local で ssh，slogin などのコマンドを実行すると，

[3] ログインに ssh (slogin) を使えば，DSA ユーザ認証を設定していない状況でも一定の安全性は確保されます．

```
local% slogin remote
Host key not found from database.
Key fingerprint:
....
Are you sure you want to continue connecting (yes/no)?
```

のようにホストの鍵を登録して良いかを尋ねてきます (初回接続時のみ). ここで, yes と答えると, remote というホストの DSA 公開鍵が ~/.ssh2/hostkeys/ の下のファイルに保存されます. 次回接続するさいには, ここで登録された鍵がチェックされ, もし入れ替わっていれば警告メッセージが出力されます. これにより, DNS が書き換えられて悪意のあるホストへ接続されるような事態を防ぐことができるようになっています.

つづいて,

```
Passphrase for key "/home/username/.ssh2/id_dsa_1024_a" with
comment "1024-bit dsa, username@local, Wed Jul 19 2000 02:27:43":
```

のように尋ねてきますので, 手順 1 で入力した local 上でのパスフレーズ[4]を入力します. 認証に成功すれば, めでたくシェルのプロンプトが表示され, セッションが開始されます.

3.5 パスフレーズ入力の省略

パスフレーズを毎回入力する手間を省くためには, ssh-agent というプログラムを使用します. このプログラムは, プロセスのメモリ空間にユーザの秘密鍵を保持しておき, 必要時に送り出してくれます.

ssh-agent コマンドを次のように実行すると, バック・グラウンドで動作し, SSH2_AUTH_SOCK と SSH2_AGENT_PID の環境変数に値を設定します.

```
local% eval `ssh-agent`
Agent pid 12345
```

お使いのシェルが csh ベースのものであれば, ssh-agent -c で起動します.

次に, ssh-add コマンドを使って, ユーザの秘密鍵を登録します.

```
local% ssh-add
Adding identity: /home/username/.ssh2/id_dsa_1024_a.pub
Need passphrase for /home/username/.ssh2/id_dsa_1024_a ...
Enter passphrase:
```

この状態で SSH2 のコマンドを実行すれば, パスフレーズの問い合わせなく認証が行われます.

なお登録されている秘密鍵は, ssh-add コマンドの -l オプションで確認することができます. ssh-agent に登録した秘密鍵を (ssh-agent のメモリから) 消去するには, ssh-add コマンドの -D オプションを用います.

[4] ここで手順 3 での remote のパスフレーズを入れたくなるかも知れませんが, remote の SSH2 サーバが認証するのは,「接続を求めてきた local というホストのユーザであるあなたがホンモノかどうか」です.

3.6 パスフレーズの変更

キーに設定したパスフレーズを変更するには，次のように秘密鍵のファイルを指定して `ssh-keygen -e` を用います.

```
local% ssh-keygen -e ~/.ssh2/id_dsa_1024_a
Passphrase needed for key "1024-bit dsa,..."
Passphrase :
```

まずはこのようにパスフレーズを尋ねられますので，現在 (変更前) のものを入力した後，コメントやパスフレーズの変更を行い，変更後の秘密鍵をファイルにセーブします.

なお，パスフレーズを変更しても,「公開鍵」をリモート計算機に送り直す必要はありません.

3.7 安全な X コネクションの確立

X Window の端末エミュレータで SSH2 クライアントを用いてリモートホストにログインすると，接続先では `DISPLAY` 環境変数が，例えば`remote:10.0` のような値に設定されます. ここに設定された`remote:10.0` のサーバは仮想的なもので，ここに X コネクションを張ると，SSH2 コネクションを経由して `local` の X サーバに接続してくれます. ただし，暗号化・復号化に負荷がかかり，多少動作が遅くなります.

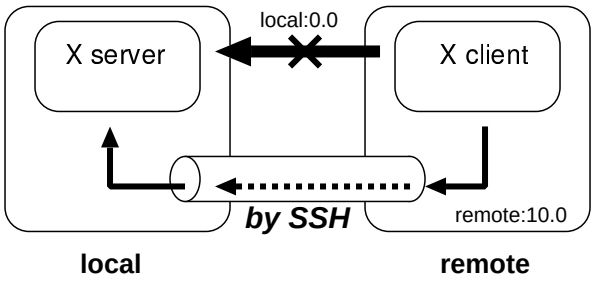

図 1: 安全な X コネクション

3.8 ポートフォワーディング

SSH2 は，rlogin や rcp だけでなく，図 2 のように任意のポートを使用した通信を安全なものとすることができます.

例えば，`local` の 8023 番ポートを `remote` の 23 番 (telnet) ポートへ接続するには，次のようにします.

```
local% ssh -f -L 8023:remote:23 remote
```

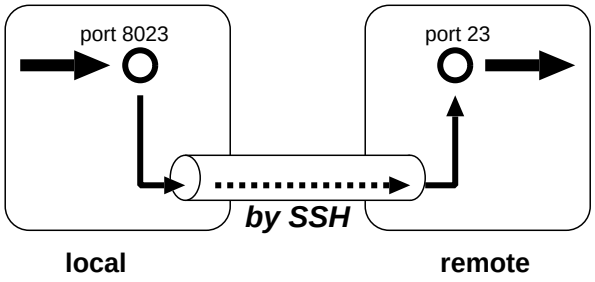

図 2: ポートフォワーディング

ここで，パスフレーズを入力して認証が完了すると，ssh のプロセスがバックグラウンドで動作して，ポートの振り替えをしてくれます．この状態で`local` の 8023 番ポートにアクセスすると`remote` の telnet に接続することができます．

また，次のように `remote` 側のポートをフォワードすることもできます．

```
% ssh -f -R 8023:local:23 remote
```

この場合，`remote` の 8023 番ポートにアクセスすると`local` の telnet に接続されます．

ポートフォワーディングは，telnet に限らず SMTP や POP にも用いることができます．

4. PC での利用方法 (SSHWin 2.2.0)

SSHWin は，Windows95/98,NT,2000 で利用できる SSH2 のクライアントソフトウェアです．パーソナルコンピュータ (以下，PC) から安全な通信経路でリモートの計算機 (以下，WS) を利用するためには，お使いの PC にインストールしておく必要があります．

なお SSHWin は SSH2 プロトコルのみサポートしていますので，WS に SSH1 のサーバしかインストールされていない場合は接続できません．必ず SSH2 のサーバをインストールしてお使い下さい．

4.1 バイナリの入手とインストール

SSHWin についても京都大学で非商用ライセンスを取得しています．次の URL からリンクを辿り，該当のファイル (`SSHWin-2.2.0.exe`) をお使いの PC の適当なフォルダに保存してください．

```
http://www.kuins.kyoto-u.ac.jp/download/
```

インストールは簡単です．ダウンロードしたファイル (`SSHWin-2.2.0.exe`) をダブルクリックするとインストーラが起動しますので，そのまま画面にしたがって行います．

4.2 SSHWin の利用

インストール時に何も変更しない場合，インストールが完了後に PC のデスクトップに図 3 のようなアイコンが作成されます.

図 3: SSHWin のアイコン

SSH Secure Shell Client は端末エミュレータで，**SSH Secure File Transfer Client** はファイル転送用のクライアントです．これらを使って，WS に安全な経路でログインしたり，ファイルを転送することができます.

SSHWin をインストールすると，そのままでもリモートの WS のアカウントとパスワードを利用して安全にログインすることができますが，UNIX クライアントの場合と同様の準備を行うと，パスフレーズを使ったより安全な接続が可能となります．その手順は次の通りです.

1. ユーザ鍵ファイルの作成デスクトップの**SSH Secure Shell Client** を起動し，図 4 のように設定のアイコンをクリックします.

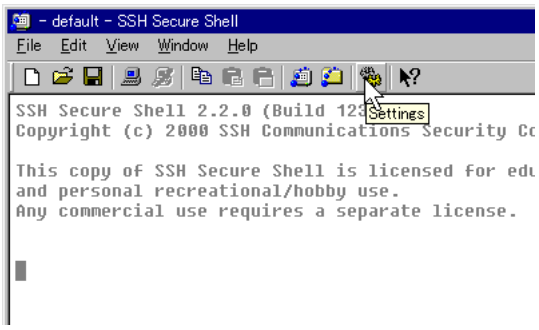

図 4: 設定

図 5 の設定ウィンドウから,「**User Key**」を選択し，`Genarate New Keypair..` をクリックします.

ユーザ鍵作成のウィンドウが表示されますので，指示に従い進みます．鍵の長さは，デフォルトで1024 となっており，通常はこのままで良いでしょう.

ユーザ鍵の生成が終わると，図 6 のように **ファイル名・コメント・パスフレーズ**を入力するためのウィンドウが表示されます.

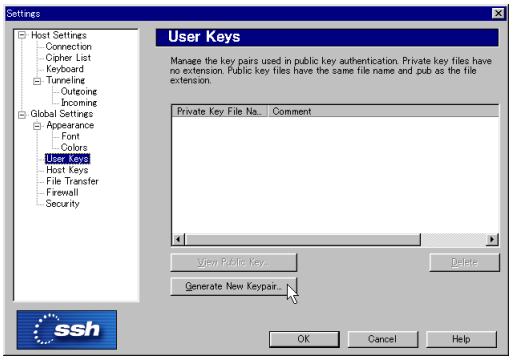

図 5: User Key の作成

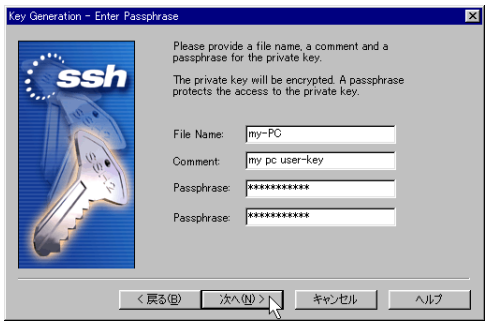

図 6: User Key の作成 つづき

ファイル名は適当な名前でかまいません．ここでは，my-PC としました．コメントも適当でかまいません．パスフレーズは間違えないように二度入力します．短いパスフレーズでは意味がありませんが，長すぎるのも毎回の入力が大変なので適当な長さとしてください．

これでユーザ鍵ファイルが作成できました．作成したユーザ鍵ファイルは，SSHWin をインストールしたフォルダ配下に保存されています．

ユーザ鍵ファイルには，秘密鍵 (my-PC) と公開鍵 (my-PC.pub) の二つがあります．秘密鍵とパスフレーズは他人に漏らさないように気をつけましょう．

2. 公開鍵のインストール
   UNIX クライアントの準備に示した手順 4, 5 と同じことを行います．具体的には，先ほど作った公開鍵 (my-PC.pub) をログイン先の~/.ssh2 ディレクトリに置き，authorization というファイルに Key my-PC.pub と記述 (追加) します．

これで準備は完了です．早速ログインしてみましょう．SSHWin を起動して，図 7 のように接続のアイコンをクリックします．

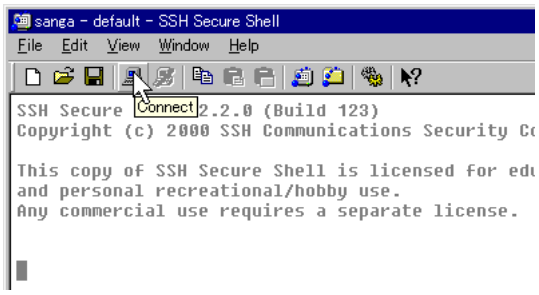

図 7: WS にログイン

図 8 のウィンドウで，ログインする WS のホスト名とユーザ名を指定して，OK をクリックします．パスフレーズを使ってログインするので，ここにパスワードは記入しません．

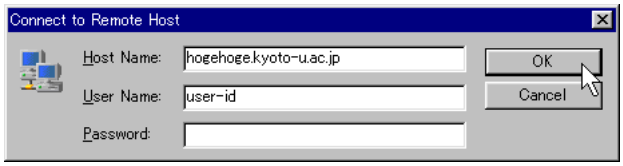

図 8: ログインのウィンドウ

初めて接続する WS の場合にはその旨のメッセージが表示されますので，はい をクリックして次にすすむと，図 9 のようにパスフレーズを入力するウィンドウが表示されます．設定したパスフレーズを記入して OK をクリックします．

図 9: パスフレーズの入力

認証が終わるとめでたくログインが完了します．

4.3 ポートフォワーディング

SSHWin の端末エミュレータでは，残念ながら日本語の表示ができません．どうしても日本語を使う必要がある場合は，ポートフォワーディング機能を使って，別の端末エミュレータを使います．

なお，ポートフォワーディング機能を使うことにより，telnet の他にも POP や SMTP も安全な通信経路で利用することができます．

ポートフォワーディングを設定するには，図 4 のように設定のアイコンをクリック，設定ウィンドウを立ち上げ，図 10 のように「**Outgoing**」を選択，`Add..`をクリックします．

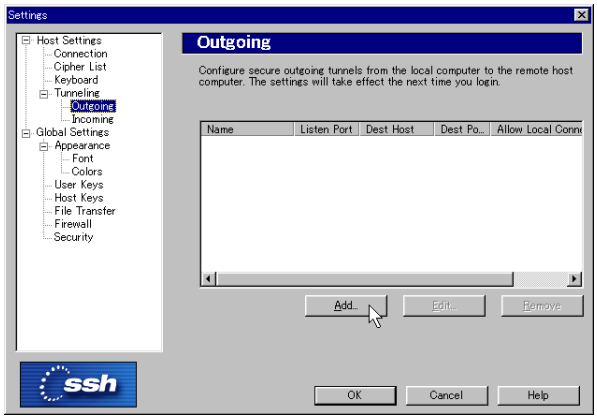

図 10: ポートフォワーディングの設定

図 11 のようなウィンドウが表示されますので，ここでポートフォワーディングするホスト名やポート番号を指定します．

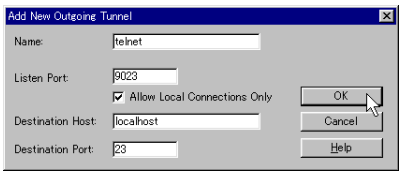

図 11: ポートフォワーディングの設定 つづき

Name:の欄には適当な文字を記入しておきます．**Listen Port:**には PC 側のポート番号を指定します．ここでは，9023 としました．**Destination Host:**には，接続先の WS のホスト名を指定します．ここでは，`localhost` となっていますが，これは SSHWin で接続したホスト自身を表しています．**Destination Port:**には接続先の WS のポート番号を指定します．ここでは 23 とし，`telnet` ポートへの接続を指定します．

`OK`をクリックし，ウィンドウを閉じます．ポートフォワーディングの設定は以上です．設定ウィンドウも閉じましょう．

4.4 ポートフォワーディングによるログイン

ポートフォワーディング機能を利用するためには，必ずいったん SSHWin でログインを完了しておく必要があります．この接続がなされている間，ポートフォワーディング機能が有

効となります.

まず，図 8 および図 9 の要領で WS に接続します．次に，適当な端末エミュレータを起動します．ここでは，Windows に標準で用意されている TELNET を使っています.

ツールバーの **接続** をクリックし，メニューから**リモートシステム**を選択すると，図 12 のウィンドウが表示されます.

図 12: ポートフォワーディングによるログイン

ホスト名:には `localhost` を指定し，**ポート:**に先ほどポートフォワーディングのために設定した9023 を指定，接続 をクリックすると，図 13 のように，SSHWin で接続した同じ WS のログインプロンプトが現れますので，普通にログインしてください.

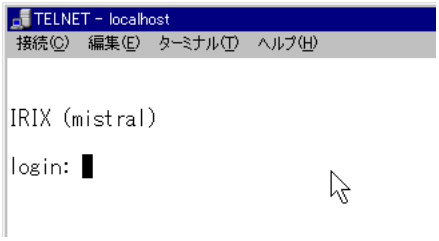

図 13: ポートフォワーディングによるログイン つづき

4.5 ポートフォワーディングによるメール送受信

ポートフォワーディングは，POP や SMTP のポートにも利用することができます.

たとえば，POP サーバのホスト名が `pop.kyoto-u.ac.jp` で，SMTP サーバのホスト名が `smtp.kyoto-u.ac.jp` である場合，図 14 のように設定します.

この後，SSHWin で接続したうえでメーラを利用すれば，メールの送受信を安全な経路で行うことができます．ここで，メーラ側の設定をそれぞれローカル (PC 側) のポート (ここでは POP が 9110，SMTP が 9025) を変更する必要があります．ただし，ポート番号を指定できないメーラをお使いの場合は，ポートフォワーディングの利用ができません.

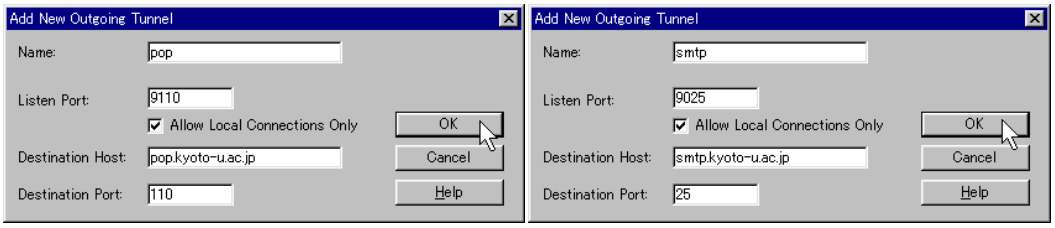

図 14: POP や SMTP のポートフォワーディング

5. おわりに

本稿では，SSH2 について簡単にその設定法・利用法を解説しました．出張先や自宅の学外ネットワークから学内にログインする場合には，できるだけ SSH2 クライアントを利用することをお奨めします．

より詳しい利用法に関しては，ソースコードの中に解説がありますので，参考にしてください．また，繰返しになりますが，KUINS のページからダウンロードしたソフトウェアに関しては，ライセンス条項を遵守いただくようお願いします．

不正アクセスに関する連絡先について

学術情報ネットワーク機構事務室

平成 12 年 2 月 13 日より「不正アクセス行為の禁止等に関する法律」が施行されました[1]．京都大学における，不正アクセス行為にかかる相談先および連絡先は，

学術情報ネットワーク機構事務室情報システム管理掛 (075)753-7841
（大型計算機センター等ネットワーク掛 (075)753-7432）

となっておりますので，ここにお知らせします．

[1] 法律の条文と骨子等は警察庁のホームページ (**http://www.npa.go.jp/**) に公開されています．

SMTP AUTH 顛末記

関口隆昭 （大学院情報学研究科）

1. はじめに

SMTP はインターネットにおける代表的なメール転送プロトコルです．しかし SMTP にはクライアントを認証する手段が用意されていなかったため，適切な設定を行っていないと，いわゆる SPAM メールの中継や「なりすまし」メールの原因となり，ある日突然，他サイト等から苦情が来ることになってしまいます．

そんな SMTP にも，RFC 2554 に定義された「AUTH」拡張によって，クライアントを認証する手段が提供されました．この夏，私達の研究室では先生方を始めとして外国へ出張する人が多いため，SPAM メールの中継を拒否しつつ研究室外からでも研究室のメールサーバを使用してメールを送信できるように，この SMTP AUTH による認証システム付きのメールサーバを導入することになりました．

2. インストール

今回インストールしたソフトウェアは以下のとおりです．

- Berkeley DB 3.1.14 (`http://www.sleepycat.com/`)
- Kerberos 4 1.0.1 (`http://www.pdc.kth.se/kth-krb/`)
- Cyrus SASL 1.5.21 (`http://asg.web.cmu.edu/sasl/`)
- Sendmail 8.10.2 (`http://www.sendmail.org/`)

sendmail は，Cyrus imapd 用に開発された認証用ライブラリ SASL (Simple Authentication and Security Layer) を用いることによって，バージョン 8.10 から SMTP の AUTH 拡張に対応しています．以下にそれぞれのインストールの顛末を簡単に書きますが，特に記録をとっていたわけではないので，かなりうろ覚えの記憶を頼りに書いていることをご了承ください．なお，インストール対象のマシンは Sun Ultra 60 + Solaris 7 です．

Berkley DB 3.1.14

まずは Berkley DB (Database) が必要だとのことです．このデータベースライブラリ自体は，すでに別マシンの /usr/local に入れてあったのですが，こちらはパッケージから入れたものであり多少バージョンが古かったので，改めて入れなおすことにしました．

```
% tar xvfz db-3.1.14.tar.gz
% cd db-3.1.14/build_unix/
% env CFLAGS=-O2 ../dist/configure --prefix=/usr
% make
# make install
```

インストール自体は何事もなく完了したのですが，SASL の構築 (後述) の時になってはじめて，実はシェアードライブラリ (libdb.so) のほうが必要であったこと，そして普通に make しただけでは作ってくれていないことが判明しました．そのため make をやり直すことになりました．

```
% make distclean
% env CFLAGS=-O2 ../dist/configure --prefix=/usr --enable-dynamic
% make
# make install
```

Kerberos 4 1.0.1

Berkley DB のインストールが終わった時点で，SASL のインストールを試みました．SASL の配布ファイルに含まれていたドキュメントによれば，Kerberos 4 はオプショナルなものであると書かれていたからです．ところが，いざ SASL の configure を走らせてみると，今度はライブラリ libdes.so が無いと警告が出ました．そんなものは最初から入れていないので，この際 Kerberos 4 もインストールしておくことにしました．

```
% tar xvfz krb4-1.0.1.tar.gz
% cd krb4-1.0.1
% env CFLAGS=-O2 ./configure
% make
# make install
```

インストールは問題無く終わったのですが，やはりシェアードライブラリが出来ていなかったので，また make をやり直しました．

```
% make distclean
% env CFLAGS=-O2 ./configure --enable-shared
% make
```

しかし，シェアードライブラリを作らせようとすると，どうしてもエラーが出て make が途中で終了してしまいます．インストール対象のマシンに既に入っている別のライブラリが原因でエラーが出ているみたいでしたが，原因を探るのがめんどくさかったのと，目的のライブラリの構築までは出来ていたこともあり，Makefile を見つつ手動でライブラリをコピーして，めでたく一件落着としました．

```
# cp libdes.so /usr/athena/lib
# cp libkrb.so /usr/athena/lib
# ...
```

Cyrus SASL 1.5.21

以上で，やっと SASL の構築に入ることができました．ところが configure が，やはり libkrb.so が無いと警告を出してきます．ログファイルを見たところ，nsl ライブラリ内の関数を参照できないのが原因で libkrb.so のテストプログラムがリンクに失敗しているみたいでしたので，明示的に nsl ライブラリを指定して configure を再実行しました．

```
% tar xvfz cyrus-sasl-1.5.21.tar.gz
% cd cyrus-sasl-1.5.21
% env CFLAGS=-O2 LIBS=-lnsl ./configure --prefix=/usr \
--with-des=/usr/athena --enable-krb4=no \
--enable-gssapi=no --enable-login=yes
% make
# make install
```

LOGIN 認証 (`--enable-login`) は，デフォルトでは使用不可になっていますが，Microsoft Outlook Express が LOGIN 認証を使うらしいので，使用可能にしておきました．

Sendmail 8.10.2

最後が sendmail のインストールです．昔やった時と何だか make の方法が変わっていて一瞬戸惑いましたが（昔は makesendmail とかいうスクリプトを走らせていたような気がします），ここは何の問題も起きませんでした．

ソースを展開して，まずは `devtools/Site/site.config.m4` を書きます．

```
APPENDDEF(‘confLIBS’, ‘-ldes -lgdbm’)
APPENDDEF(‘confLIBDIRS’, ‘-L/usr/athena/lib -R/usr/athena/lib’)
APPENDDEF(‘confENVDEF’, ‘-DSASL’)
APPENDDEF(‘conf_sendmail_LIBS’, ‘-lsasl’)
```

その後 make，そしてインストールをしました．

```
% sh Build
# sh Build install
```

3．設定と動作確認

インストール完了後，まず SASL のパスワードデータベースを作成しました．データベースの作成とパスワードの設定はコマンド saslpasswd で行います．例えばあるユーザーのパスワードを設定するときは，サーバ上でスーパーユーザーになり，

```
# saslpasswd ユーザー名
```

と打ち，その後パスワードを入力します．スーパーユーザーにならないとパスワードを設定できないのでは個々のユーザーにとって不便であるので，今後改良する必要があると思われます．

その後，作成したパスワードデータベースを sendmail から利用できるように，`/usr/lib/sasl/Sendmail.conf` というファイルを作成して，次の一行を書いておきました．

```
pwcheck_method: sasldb
```

最後に `sendmail.cf` の作成を行いました．今までは CF を使用していたのですが，どうやら SMTP AUTH 対応はしていないようなので，sendmail 付属のツールで作成することになりました．m4 という新たな言語（大昔からあるらしいですが，少なくとも私は初めて使いました）に手間取りましたが，sendmail 付属の README を読みつつ何とか設定を行いました．SMTP AUTH に関係があるのは次の 2 つのマクロです．

```
define(‘confAUTH_MECHANISMS’, ‘LOGIN DIGEST-MD5 CRAM-MD5’)dnl
TRUST_AUTH_MECH(‘LOGIN DIGEST-MD5 CRAM-MD5’)dnl
```

「confAUTH_MECHANISMS」は AUTH コマンドで利用可能な認証方式であり，その中でも「TRUST_AUTH_MECH」に定義された認証に成功すればメールの中継が許可される，と理解したのですが，これは間違っているかもしれません．とりあえず出来上がった sendmail.cf を読むと，デフォルトでは中継は拒否するが，認証されたクライアントに対しては中継を許可するようです．

MUA 側の設定は以下のようになります．

- アカウント: ユーザー名@ドメイン名
- パスワード: saslpasswd で設定したもの

ユーザー名の後に「@ドメイン名」が必要となることになかなか気が付かず苦労しました．ドメイン名の部分は，デフォルトではサーバのホスト名 (FQDN ではない) になります．これは，サーバ側のどこかの設定で変更できると思うのですが，まだよくわかっていません．

Outlook Express バージョン 5 での設定を例に挙げると,「ツール」メニューから「アカウント」を選択し,「メール」でサーバを選択してから「プロパティ」を押し,「サーバー」タブの「このサーバーは認証が必要 (V)」チェックボックスを ON にして,「設定 (E)」ボタンを押して出てきたダイアログボックスで上記のものを入力します．ただ，このダイアログボックス上の「セキュリティで保護されたパスワード認証でログオンする (S)」というチェックボックスは機能しません．どうやら，ここで使われる認証機構は sendmail では対応していないもののようです．チェックしないと LOGIN 認証が使用されます．

必要な作業は終わり，後は動作確認です．Outlook Express を用いて行ってみたのですが，案の定うまく行きませんでした（原因は，LOGIN 認証を使用可能なように SASL を構築していなかったことと，アカウント名に「@ドメイン」が必要なことになかなか気が付かなかったからです）．LOGIN 認証以外の認証方式に関しては，研究室の先生にお願いして動作確認を行ってもらったところ，うまく動いているようでした．

4. まとめ

今回はメール中継問題に対する 1 つの対策として，Sendmail 8.10 と SMTP AUTH のインストール及び設定を行いました．ドキュメントがいまだ少ないこともあり時間がかかりましたが，問題無く動作するところまでは確認できました．

SMTP AUTH は，Outlook Express，Netscape Messenger，Eudora Pro 等，多数の MUA が対応しているようですが，MUA によって使用できる認証方式に違いがあります．特に，事実上ユーザー数が最も多いと思われる Outlook Express では，セキュリティレベルが比較的低い LOGIN 認証しか使用できません．SMTP AUTH の利用が主流になるかどうかは，各 MUA の今後の対応次第であると思われます．

MTA 間の認証等，まだよくわかっていない点が多いのですが，読まれた方の参考に少しでもなれば幸いです．

学外からのメールの送信について

岡部寿男 （大学院情報学研究科）

KUINS では，度重なる SPAM メール踏み台事件に対処するため，ユーザによる SPAM 対策設定の徹底と対策済サーバの届け出，KUINS による対策済みサーバ以外のホストへはメールを届かなくするフィルタリングの組み合わせによる対策を開始しています (前々号記事[1]参照)．しかし，このような対策強化により，学外のネットワーク環境から京大内のホストを SMTP サーバとして指定してメールを出すことがそのままでは不可能となっています．そこで，このような場合にはどのような方法を取ればよいのかをメモとして整理してみました．主としてノートパソコンなどを，自宅などのダイヤルアップ PPP 環境で接続したり，出張先でネットワークを借りて接続させてもらったりしてメールを送る場合を想定しています．

（方法１） パソコン上の sendmail を利用する

最近ですと Linux などの PC UNIX をお使いの方も多いと思います．その場合には，sendmail.cf を正しく設定すれば，メールを京大内のサーバを一律に経由して送信するのではなく，DNS を参照して本来の受け取り先を調べてそこに送信するようにできます．但し，一通のメールであっても配送先が複数のサイトに分かれている場合にはそれぞれに対して別々に送信するため，配信に掛かる時間は長くなります．またそのうちどれかのサイトがトラブルで停止している場合には，タイムアウトまで待つことになるだけでなく，そこへの配送は一定時間待った後に再度行うことになります．このため，ダイヤルアップ PPP 環境で使っている場合には電話代や再接続の手間などで不利になります．

（方法２） SMTP AUTH 拡張の利用

最近になって，メール配送のためのプロトコルである SMTP (Simple Mail Transfer Protocol) でユーザ認証や通信路暗号化を行うための拡張が標準化され (RFC2554, RFC2555)，サーバ (MTA; Mail Transfer Agent) やクライアント (MUA; Mail User Agent) もこれに対応するようになって来ました．たとえばサーバでは

- sendmail 8.10 以降
- Microsoft Exchange Server

などが対応済みですし，クライアントとしては

- Microsoft Outlook Express 5
- Microsoft Outlook 2000
- Netscape Messenger
- Eudora Pro
- Winbiff (V2.31beta3 以降)

[1] http://www.kuins.kyoto-u.ac.jp/news/31/spam.html

- Wanderlust 1.1.1 + SLIM[2]
- Semi Gnus + SLIM
- Mew (1.95b35 以降)

などが対応しているそうです．sendmail 8.10 のインストール方法と Outlook Express 5 での設定方法については別記事を参照ください．

なお，RFC2555 ではさまざまな認証方式を用いることができるようになっており，sendmail 8.10 ではこのうち LOGIN，CRAM-MD5，DIGEST-MD5 などに対応しています．一方 Outlook Express では LOGIN 認証方式と Microsoft 独自の方式のみに対応しており，共通で用いることができるのは LOGIN 認証方式のみということになります．ところが LOGIN 認証方式ではアカウントとパスワードが暗号化されないままネットワークを流れますので，盗聴の可能性を排除することができません．**LOGIN 認証方式を用いる場合には，必ず通常のログインや POP ／ IMAP で用いるパスワードとは別のものを設定するようにしてください．**

（方法３） 各種の VPN (Virtual Private Network) 技術の利用

遠隔のネットワークから自分の普段いるネットワーク環境へ安全にアクセスするために，各種の VPN (Virtual Private Network) 技術が用いられています．例としては SOCKS，SSH，Microsoft PPTP (point-to-point tunneling protocol) 等があります．これらを用いることで，論理的には京大内のホストとして学内の SMTP サーバにアクセスできるようになります．

SSH の設定法については本号の別記事をご覧下さい．SSH を用いた SMTP のポートフォーワーディングの設定については，東京大学情報基盤センターの利用者向けドキュメント[3]などが参考になるでしょう．

（方法４） POP before SMTP の利用

国内商用プロバイダなどでは，ローミングサービスなどそのプロバイダ外からのアクセスにおいて POP before SMTP 方式を採用することが多くなっています．これは，POP による認証付のアクセスがあった同じ IP アドレスからの SMTP のアクセスを一定時間許そうというもので，厳密に言えば抜け道を作っていることになりますが，実用上はこれで十分と考えられます．POP と SMTP に対応しているクライアントであればどれでも使えることが利点です．但し，多くのクライアントは通常 SMTP による送信の後 POP による受信を行うため，操作を多少工夫してやる必要があることもあります．「POP before SMTP」というキーワードで検索エンジンで探すと，プロバイダが提供している各種クライアントに対する具体的な操作の工夫の仕方がたくさん引っかかるはずです．

サーバ側では POP サーバの吐くログを拾って一定時間アクセス権を与えるための仕掛けを組み込むことが必要です．実装には簡便なものからちゃんとしたものまでいろいろあるよ

[2] ftp://opaopa.org/pub/elisp/
[3] http://www.ecc.u-tokyo.ac.jp/secure_access/

うです．ドキュメント類も含めちゃんとしていると思われるものとして，DRAC (Dynamic Relay Authorization Control)[4] をあげておきます．

（方法５） CF の ROAM_* を用いる方法

以上に述べた方法に比べればセキュリティ的には危うくお勧めできませんが，クライアント側の対応が全く不要な方法として CF[5] の ROAM_* の機能を利用する方法があります．これは，ROAM_HOST_IPADDR に指定された IP アドレスまたは ROAM_HOST_DOMAIN に指定されたドメインに属するホストから，エンベロープの発信者アドレスが ROAM_USERS に指定されたものに含まれる場合にのみ，遠隔からのメールを中継することを許可するものです．設定の詳細については CF に添付のドキュメントをご覧下さい．

但し，エンベロープの発信者は誰でも偽造できてしまうこと，そのようなメールの送り方をしていることが送られたメールのメールヘッダを見ればすぐにばれてしまうことから，よほど慎重に設定しないとメールを送るたびにセキュリティホールの存在を宣伝しているのに近い状態になりかねません．また最近の SPAM メール送信用ツールの中には，Web ページから切り出したメールアドレスを用いて SPAM が中継されるかどうかのチェックに自動的に使う悪賢いものもあるようです．ROAM_HOST_DOMAIN に商用プロバイダを設定する場合は，ユーザ管理をしっかり行っていて万一のことがあったときのこちらからの調査要求にある程度誠意を持って対応してくれるところであるかを確認してから行ってください．

お知らせ

KUINS ニュースへの寄稿を歓迎します．詳細は

kuins-news@kuins.kyoto-u.ac.jp

または下記までお問い合わせください．

問い合わせ先

学術情報ネットワーク機構情報システム管理掛 ((075) 753-7841)
（大型計算機センター等ネットワーク掛 ((075) 753-7432)）

[4] http://mail.cc.umanitoba.ca/drac/
[5] ftp://ftp.kyoto.wide.ad.jp/mail/CF/

KUINS 会議日誌

平成 11 年 12 月 15 日～平成 12 年 7 月 31 日

学術情報システム整備委員会

平成 12 年 2 月 3 日（第 24 回）
- 委員異動の報告
- KUINS 機構定員化の概算要求について

平成 12 年 3 月 15 日（第 25 回）
- 技術専門委員会からの報告
- 学術情報専門委員会からの報告
- KUINS 機構定員化の概算要求について
- キャンパス情報ネットワーク設備第 3 期整備計画（案）について
- キャンパス情報ネットワーク設備第 3 期整備計画（案）について
- 学術情報ネットワーク機構運営経費要求（案）について
- 学術情報ネットワーク機構維持管理経費要求について

学術情報システム整備委員会技術専門委員会

平成 12 年 2 月 29 日（第 48 回）
- 平成 13 年度概算要求について
- その他

学術情報システム整備委員会学術情報専門委員会

平成 12 年 3 月 21 日（第 1 回）
- 情報倫理について
- その他

学術情報ネットワーク機構運営会議

平成 12 年 3 月 29 日
- 機構運営会議の構成員の交代について
- 研究開発部門の担当教官の補充について
- ＫＵＩＮＳ機構の定員化について

学術情報ネットワーク機構課長等連絡会議

平成 12 年 3 月 29 日
- ネットワーク機構運営会議について
- その他

KUINS ネットグループ連絡会議

平成 12 年 5 月 2 日（第 87 回）
- 接続端末数について
- 接続状況報告
- KUINS 障害報告
- KUINS-I 用ルータの保守について
- KUINS-II 部局購入の ATM 機器の保守について

平成 12 年 6 月 14 日（第 88 回）
- 接続端末数について
- 接続状況報告
- KUINS 障害報告
- スパムメール不正中継対策フィルタ設定実施状況について
- SSH (Secure Shell) の non-commercial site license の取得について
- KUMX の運用について