コンピュータリテラシ＃3 – ネットワークと安全性

久野 靖 (電気通信大学)

2017.5.19

1　今回の目標

今回の目標は次の通りです。

- ネットワークにおける安全性の概念やそのための技術を理解する— 様々な活動の前にまず安全であることが必須の前提です。
- WWW、電子メールというネットワーク上の基本的な通信サービスを理解する — どちらも必ずお世話になりますが、その原理まで知っておくことが安全性や使いこなしの鍵です。
- ネットワークコミュニケーションの特性やその注意点を理解する— これらを意識しておかないと社会的トラブルになります。

なお、今回の内容に関連してですが、本学では全構成員に毎年 10 月 1 日までに「INFOSS 情報倫理 [速習版]」を受講しテストを 80 点以上で合格することを義務づけていますので必ず毎年受講してください。

2　情報セキュリティ

2.1　セキュリティと情報セキュリティ exam

セキュリティ(security, 安全性) とはその名前通り安全が保たれること、たとえば身体的危険や窃盗等の危険に逢わないことです。一般にはそのため、家屋に施錠したり、ガードマンを頼んだり、銀行に預けたりしますね。では**情報セキュリティ**(情報に関する安全性) はどうでしょう。たとえばあなたは、無線 LAN 経由で情報をやりとりする際、その内容が周囲の人に傍受され放題だと知っていましたか? [1] 多くの人は意識せずに無線 LAN 経由で重要な情報を送ったりしていますが、実際には情報セキュリティについて意識しておかないと、思わぬトラブルに遭遇する可能性があるのです。

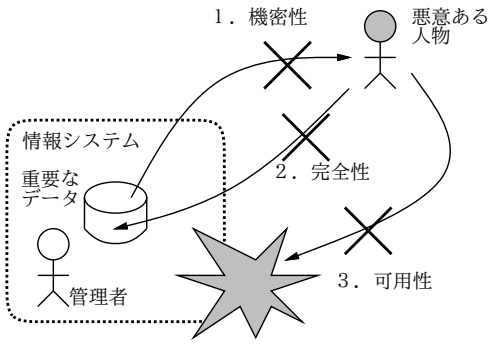

図 1: 情報セキュリティの 3 要素

一般に、情報セキュリティに関する目標としては、次の 3 つが挙げられます (図 1)。

[1] 暗号化通信を使っていれば別です。なお、暗号といっても無線 LAN の暗号化機能は除外して考えてください。同じ無線 LAN ステーションを使っている他人も、あなたと同じキーを使うので、それらの人には解読し放題ですから。

- **機密性** — 情報が、取得できるべきでない対象に漏洩しないこと。
- **完全性** — 情報が、完全な状態である、つまり改ざんされないこと。
- **可用性** — 情報が、必要なとき利用可能、つまりシステムがきちんと動作すること。

これらから、情報セキュリティの多くは人的問題、管理の問題だと分かります。たとえば機密性は、ある情報が「誰は」見てよく「誰は」だめか明確でなければ適正に保てません。完全性については、ある情報を権限を持つ人が業務の必要に応じて書き換えることは必要ですが、権限がない人が書き換えたり、権限がある人でも業務上の必要以外で書き換えるのは問題です。実際、企業等におけるセキュリティ侵害の大多数は外部の人ではなく内部の人によるという調査結果があります。

可用性については、いくらネットワーク経由の攻撃を防御しても、サーバ室に誰かが入ってきて電源を切ったり装置を破壊できたら無意味です。地震・火事など自然災害への対策も重要です。可用性に対する対策としては、システムを多重化してどちらかが壊れても運用を続行可能とする、重要なデータは定期的にバックアップして安全な場所に保管する、などがあります (そしてバックアップが流出すると機密性が損なわれます)。

今日、多くの情報システムや個人にとって情報セキュリティ侵害の大きな要因となっているものに、不正アクセスとマルウェアがあります。なお、これらの侵害においては、当事者の認識不足がきっかけである場合が少なからずあるので、そのようなきっかけにならないよう心してください。

不正アクセスとは、(1) 他人の ID/パスワードを使ったり、(2) ソフトの欠陥を衝くなどして、本来のアクセス制限を回避してシステムを利用することを言います。これに対する対策は、(1) についてはパスワードを適正に管理すること [2]、(2) についてはシステムソフトウェアのアップデートを適正に行うことが挙げられます。[3]

マルウェア (malware) とは、システム所有者の意図しないことを行うような有害ソフトウェア全般を指す言葉であり、**ウィルス** (プログラムや文書の中に自分自身を埋め込み、、ユーザがそのファイルを開くことで感染し広まるもの)、**ワーム** (ネットワーク経由で自分のコピーを他のホストに送り込むもの)、**トロイの木馬** (有用に見えるソフトウェアで、ユーザが騙されてそれを使うことで広まるもの)、などの種類があります。マルウェア対策としては、ウィルス対策ソフトウェア (既知のマルウェアを監視して排除/警告する) の使用と、他人から送られて来たりネットワーク上で入手したソフトウェアを安易に実行しないことが重要です。

2.2 暗号技術 exam

暗号 (cryptography) は昔から情報の機密性・完全性の維持に使われて来ました。その基本は秘密の通信に際し、もとのテキスト (**平文**) を一定の規則によりランダムに見えるような文字列 (**暗号文**) に変換 (**暗号化**) して送り、受け手はそれを逆に変換 (**復号**) して読む、というものです。通信途中の暗号文を誰かが盗み見ても暗号が解読できなければ秘密を知ることができませんし (機密性の保持)、暗号化の規則が分かっていなければ偽物のメッセージでだますこともできません (完全性の保持)。

ところでこの暗号化や復号は昔は人間がやっていて大変でしたが、今では当然コンピュータでやります。だったら、暗号化や復号をおこなうプログラムを秘密にしておくのでしょうか (悪い人に復号プログラムが渡ると秘密が解読されますし、暗号化プログラムが渡ると偽物の秘密通信が来るかも知れません)。実際にはプログラムを秘密にするというのは難しいので、プログラムは誰もが共通に使うことにして、そのプログラムに**鍵** (key) と呼ばれる情報 (実際には 128 ビットとか 512 ビットとかの長さの 0/1 の列です) を与えて暗号化/復号をおこないます。この鍵を秘密にしておくことで、悪い人に解読されないようにするわけです。

上記が暗号の基本でしたが、その方式には大きく分けて、次の 2 種類があります。

- **対称鍵暗号**ないし**共通鍵暗号**— 暗号化と復号に同じ鍵を用いる。

[2] 推測されやすいパスワードを避ける、複数サイトに同じパスワードを使わない、紙に書いて放置しないなど

[3] システムには多くの欠陥が含まれていて絶えず新しいものが発見されているので、その対策を施したものに入れ替えておかないと、その発見された欠陥を利用して不正アクセスを受ける危険があります。

- **公開鍵暗号** — 2 つの鍵の対を使い、片方で暗号化、他方で復号を行う。

歴史的には対称鍵暗号の方が古くからありますが、これには「鍵をどうやって安全に伝達するか」という問題があります。ネットワークで送る? 送っているのを盗聴されたら後の暗号化通信も解読されてしまいます。USB メモリなどに入れて手渡たせば安全ですが、面倒です。

そこで考案されたのが公開鍵暗号です。この方式では、鍵を**秘密鍵**と**公開鍵**の対で生成し、公開鍵は皆に知らせます。秘密の通信をしたい人は、公開鍵を使って通信を暗号化して送り、受け手は自分だけが持つ秘密鍵で解読します。他人は秘密鍵を知らないので、暗号を解読できません (図 2)。

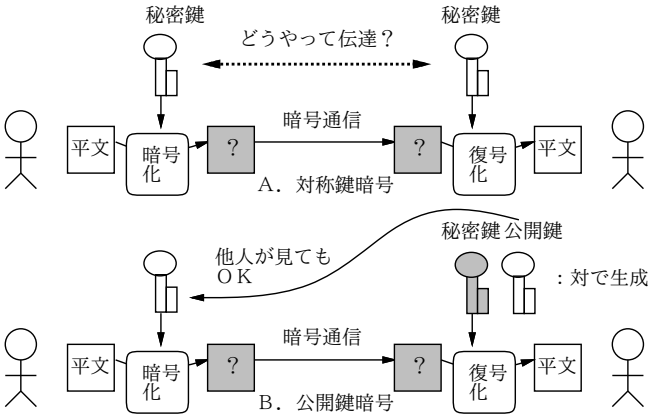

図 2: 対称鍵暗号と公開鍵暗号

また、公開鍵暗号の別の用途として、持ち主が持つ秘密鍵によって情報に印をつけ、対応する公開鍵を用いてそれが OK であることを検証する (偽物の秘密鍵だと NG となる) ことにより、情報を発信したのが確かに秘密鍵を持つ本人だということを証明する、というものもあります。こちらは書類にサインするのと似ていることから、**電子署名** (digital signature) と言います。

どんな暗号であっても、十分な時間を掛ければ…すべての鍵の可能性を「総当たり」で試せば、破れます。ただしそれには、すごく長い時間が掛かります。512 ビットの鍵であれば 2^{512} 通りの場合があり、これはすごく大きな数なので、全部試すよりも前に太陽が燃え尽きてしまうくらい時間が掛かるから安全だ、と考えるわけです。もちろん、鍵のビット数が増えるほど、試す場合の数は多くなりますから、より安全になります (その代わり処理は遅くなります)。

2.3 PKI と証明書の連鎖

先に公開鍵を皆に知らせると簡単に言いましたが、途中で誰かがその公開鍵を別のものにすりかえたりすると、秘密のつもりの通信がそのすりかえた人に解読されてしまいます。ですから、公開鍵が確かに本ものかを確認することが必要ですが、それを通信する相手ごとにやるのは大変そうです。

そこで **PKI**(Public Key Infrastructure) という枠組みが用意されました。これは、**CA**(Certification Authority) という組織が他の組織や個人の公開鍵を証明書により「正しいと保証」するという制度で、保証してもらいたい側は CA に自分の出自を示す書類を送るなどして確認してもらいます (もちろん、CA も商売なのでお金が掛かります)。

では、CA 自体の正しさは…それは、その CA を保証する「親 CA」があり、そのまた親…とつながって行き、最後は「根元」の CA(**ルート CA**) にたどりつきます。ルート CA はそう沢山はないので、あらかじめ確認しておくなどの手段が取れます。なお、CA の仕事は「当該組織が存在していること」を確認し保証することまでであり、その組織の業務内容や社会的信頼性を調査して保証するわけではない点は注意しておいてください。

WWW では普通の (暗号化しない) サイトが「`http://`」で始まるアドレスなのに対し、ネットショップや Web メールなどログインを要するサイトは「`https://`」を使用します。これが暗号化されたデータ転送を用いる **TLS**(ないしその旧版である **SSL**) プロトコルを使うサイトで、通信に公開鍵暗号を

使用し、PKI を用いたチェックをおこないます。そのため、代表的なルート CA について、ブラウザを配布する時に最初からその証明書情報がブラウザ内に組み込まれています。ブラウザで `https://` のページを開いた時は、このページの証明書情報からルート CA までの連鎖をたどれるかどうかをブラウザが自動的にチェックしています (図 3)。大切なのは、単に「暗号化が使われている」(通信内容が傍受されない) ことだけが重要なのではなく、「通信相手が本当に意図した相手なのか」(偽サイトにつながされていないか) も同じくらい重要だということです。

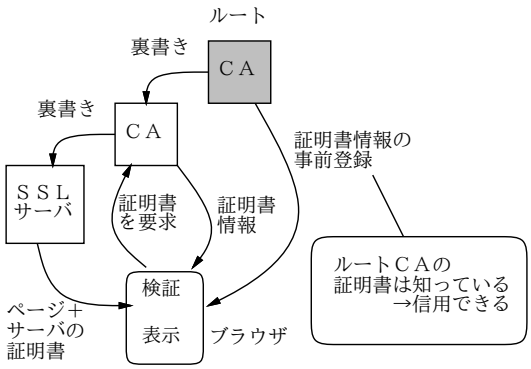

図 3: PKI と CA の連鎖

しかし、お金の節約その他 (?) の理由で、この連鎖に加わらずに `https:`のページを動かすサイトもあります。このようなサイトの証明書を (自分で勝手に正しいと言っているニュアンスで)「**オレオレ証明書**」と呼びます。その種のサイトを表示させると、ブラウザは「ルート CA への連鎖がたどれない」という警告を出し、次の選択肢を提示します (原則として (2) を選びましょう)。

(1) このサイトを恒久的に信用する。

(2) 表示をとりやめる。

最後に、普段のシステムへのログインなどパスワードに基づく**個人認証**も暗号技術が基本になっていることを注意しておきます (指紋や静脈パターンなどの**生体認証技術**は別です)。

演習 1 ブラウザで適当な `https:` サイトを開き、鍵マーククリックなどの方法で「証明書」の情報を表示させなさい (Firefox では「鍵マーク」→「>」→「more info(詳細を表示)」→ (別窓) →「View Certificate(証明書を表示)」で見られます)。表示できたら、以下のテーマから 1 つ以上やってみなさい。

a. 自分が開いている `https:`サイトからルート CA までの CA の連鎖を確認してみなさい (Firefox では「Details」タブに切替えると見られます)。できれば、複数のさまざまな国の `https:`サイトについて、連鎖がどうなっているかを整理してみられるとよいです。

b. ブラウザの「証明書管理」画面を開き、そのブラウザにどのようなルート CA の証明書が標準で入れられているか見てみなさい (Firefox では「メニュー」→「Preferences(設定)」→「Advanced(詳細)」→「Certificates(証明書)」→「View Certificaties(証明書を表示)」で見られます)。できればその中のいくつかについて、ネットで検索してどのような企業か調べてみるとなおよいでしょう。

c. 例えば `https://kotonet.com/mail/inquiry1.html` などのオレオレ証明書サイトをブラウザで開き、警告の様子を調べなさい。

   また、`https://www.e-gov.go.jp` などは単なるオレオレ証明書ではないですが、こちらも調べてみてください。

3　ネットワークサービスとその構成 exam

私達がネットワークを使うのは、メッセージを送るなど、何か「役に立つ機能」を使うためです。ネットワーク上でこのような機能を提供するもののことを**ネットワークサービス**と呼びます。

ネットワークサービスはどんな形で提供されているのでしょう? 通信を行うためには、互いに通信する 2 つのプログラムが「あちら」(リモート側/ネットワークの向こう側) と「こちら」(ローカル側/自分の手もと) で動く必要があります。「こちら」は自分のマシンだから自由になりますが、「あちら」はどうしましょうか? この問題に対する 1 つの解は、プログラムを次の 2 種類に分けることです (サーバが「あちら」、クライアントが「こちら」になります)。

- サーバ — サービスを提供するマシンで常時稼働していて、サービス要求があるまで待ち、要求があったらサービスを提供する。
- クライアント — サーバに要求を出して、そのサービスを受ける。

これを**クライアントサーバ方式**と呼びます (図 4)。この方式では、上述の通信の問題が自然な形で解決できますし、サービス提供のために必要な資源 (データ等) はサーバのところで一括管理できるので、各種サービスの実現が比較的簡単に行えます。このため、クライアントサーバ方式はネットワークの初期から今日に至るまでネットワークサービスの構成方式として広く使われています。

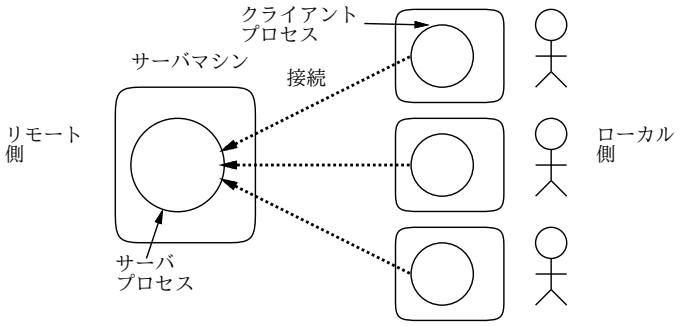

図 4: クライアントサーバ方式

クライアントはサービスを利用するユーザの数だけ動きます。ユーザはサービスを使うときにクライアントを起動し、そのプログラムがサーバと接続しサービスを受けます。つまりクライアトサーバ方式では、クライアントが各ユーザのマシンで動き、サーバはサービスを提供するための専用マシンで動く形になります。

ところで、クライアントサーバ方式ではない構成のネットワークサービスもいくつかあります。これらは**ピアツーピア方式** (**P2P**) と呼ばれ、特定のサーバはなく、各プログラムが対等な立場で通信します。このようなシステムで大規模なものの代表例としては、Napster、Winny、Skype などが挙げられます。これらはいずれも草の根的な通信やファイル転送を目的としているため、どこか 1 箇所に皆がアクセスするよりも、互いにやり取りをしたい組をうまく設定できさえすれば、それぞれが直接やり取りすることで負荷の集中を避けられるのです。[4]

4　World Wide Web

4.1　WWW とリンク exam

今日、最も広く利用されているネットワークサービスは **WWW**(World Wide Web ないし Web) です。皆様もそのクライアントである **Web ブラウザ**を使わない日はないと思います。しかし、その見慣れた画面の裏側の仕組みを考えたことのある人は、多くないかも知れません。

Web の基本的な枠組みは**ハイパーテキスト**です。ハイパーテキストとは次のようなものです。

[4]中央サーバが無いので不法コピーなどが発見されにくいという点もあります。Winny は暗号化と組み合わせてこの部分を売りものにしたソフトですが、その秘密主義がさまざまな不幸をもたらしました。

- 計算機の画面上にテキストや画像などの内容 (コンテンツ) が表示されている。
- コンテンツの中には、他のコンテンツやその特定箇所を「指し示して」いる箇所が埋め込まれている。これを**リンク**という。
- リンクの箇所を何らかの方法で選択すると、画面はそのリンク先の内容に切り替わる。
- このようなリンクで互いに結び合わされたコンテンツの集まりは、リンクを自由にたどりながら読み進んでいくことができる。

ハイパーテキストの概念そのものは WWW よりずっと前から存在しましたが、それをネットワークサービスの形に組み立てたのが WWW なのです。

WWW ではネットワーク上にある様々なモノ (資源ないしコンテンツ) を指す手段として **URL**(Uniform Resource Locator) と呼ばれる形式を使用しています。URL は一般に次の形をとります。

スキーム:アドレス

スキーム中で最も一般的なのは、Web サーバからコンテンツを取り寄せるプロトコル **HTTP**(HyperText Transfer Protocol) を使う場合で、次の形を取ります。[5]

`http://`ホスト指定`/`ディレクトリ`/.../`ディレクトリ`/`ファイル

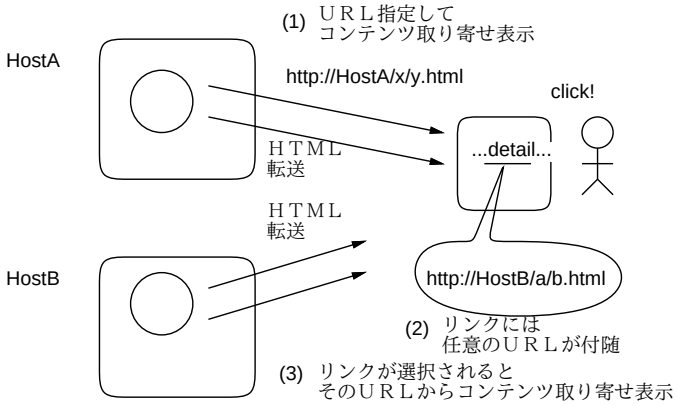

図 5: リンクの仕組み

なぜここで URL について長々と説明しているのかというと、URL こそが WWW の「肝」だからです。WWW でコンテンツを伝送しているのはごく簡単なプロトコルであり、ブラウザがやっていることは基本的には次の 2 点だけです。

a. 指定された URL からコンテンツを取り寄せる。

b. 取り寄せたコンテンツを適切な形で画面に表示する。

コンテンツを取り寄せたとき、それが **HTML**(HyperText Markup Language) 形式であれば、中にリンクが含まれていることがあります (そこに別コンテンツを指す URL 情報が含まれています)。そしてユーザがリンクを選択すると、ブラウザはリンク先のコンテンツを取り寄せて表示し (上記 a+b の動作)、結果的に「別のページへ飛ぶ」わけです (図 5)。これだけで済むのは、「リンク先」が URL の形で表され、世界中のどのサーバのどのコンテンツでも自由に指し示せるから、なのです。

[5]既に説明したように、サーバとブラウザの間で暗号化通信を行うために TLS/SSL を使うこともでき、その場合はスキームとして `https:`の代わりに `https:`を使います。

4.2 Web アプリケーション exam

前の節では WWW をサーバからコンテンツを取って来るだけのシステムとして説明しましたが、実際にはそうではありませんね。私たちは Web ブラウザを使ってネットショップの買物をしたり掲示板などで互いにコミュニケーションしたりします。それはどのような仕組みによるのでしょうか。

実は HTML によって記述されるページの中にはボタン、入力欄などの GUI 部品を含めることができます。そして、ブラウザ上でユーザが入力欄などにテキストを記入したり選択メニューで選択肢を設定したあと、「送信」ボタンを押すことでこれらの情報を Web サーバに送ることができるのです。

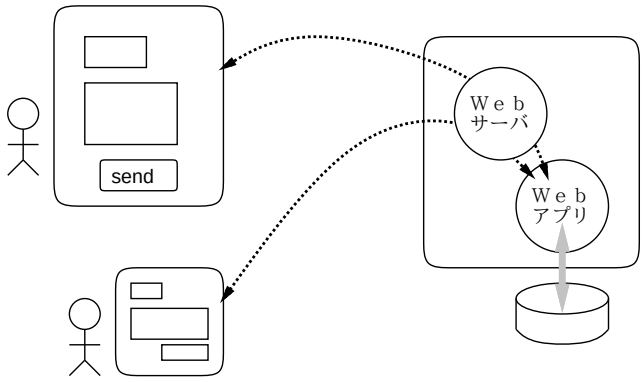

図 6: Web アプリケーション

この送信されたデータは、Web サーバと連携して動いているプログラムによって受け取られ、このプログラムがさまざまな処理を行います。そしてその結果は再び HTML としてブラウザに送られ、ブラウザの画面で処理結果を見ることができます。これがネットショップや掲示板などのサイトを使っている時に起こっていることなのです。[6] このような、Web サーバと Web ブラウザの上で動くプログラムのことを **Web アプリケーション**と言います (図 6)。

Web アプリケーションは PC 上にブラウザさえあればそれ以上特別なプログラムを入れなくても使えるので、WWW の普及とともに広く使われるようになりました。また、スマートフォンやタブレットなどでもブラウザが動くため、これらのモバイル機器からも使えるという利点も生まれました。このため、今日では非常に多くの Web アプリケーションが使われています。[7]

5 電子メール

5.1 電子メールサービスの構成 exam

電子メール (e-mail) はネットワークの初期からの情報交換サービスです。電子メールは遠隔地との情報交換を基本的に「サーバどうしで」行う設計になっていて、ユーザは「自分の手元の」サーバと通信してメッセージを投入したり取り出したりします (図 7)。ユーザがメッセージの読み書きを行う際に用いるクライアントソフトのことを**メールリーダ**ないし **MUA**(Mail User Agent) 呼びます。これと対比してメールサーバは **MTA**(Mail Transfer Agent) とも呼びます。

電子メールでは、MUA からメッセージを送信すると、メッセージは手元の MTA がまず受け取り、そこから相手側の MTA に送られます (場合によっては複数の MTA が順次中継します)。これらのデータ伝達には **SMTP**(Simple Mail Transfer Protocol) と呼ばれるプロトコルが使われます。

相手側の MTA のありかは次のようにして知ります。まずメールアドレスは「`someone@example.com`」のような形を取り、「`@`」の後ろ側はドメイン名 (この場合は `example.com`) になっています。そこで

[6] 実際には処理結果が返されたときにページ全体が切り替わるのでなく、部分的に変化するようになっている場合もあります。この場合は最初の HTML の中に連携用のプログラムが含まれていて、このプログラムが処理結果を受け取って画面の一部を変更するなどの処理をします。

[7] スマートフォンなどでは画面が小さいため、ブラウザでは使いづらい場合もあります。そのため、とくに人気がありユーザ数の多いサービスでは、ブラウザ版に加えて Android や iOS 用に専用のアプリを用意しているものも増えています。

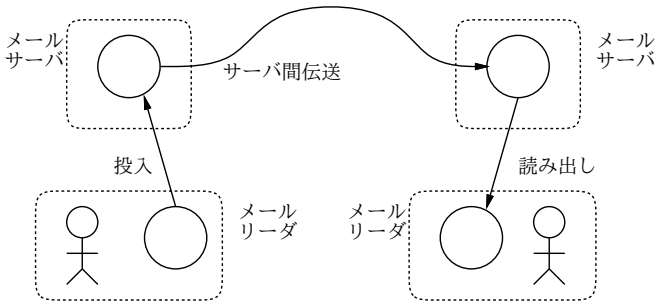

図 7: メールの伝送経路

DNS で検索することで、宛先 MTA の IP アドレスが得られます。[8]IP アドレス分かれば、そのホストの MTA に接続して「someone さん宛ですよ」といってメッセージを送れます。

メールがメールサーバに到着した後、ユーザはサーバから自分宛のメッセージを取り出して読みます。その際ユーザがサーバとやりとりするのに、大きく分けて 2 通りの方法があります。

- ユーザが手元のマシンにメッセージを格納し管理する — MTA から MUA にメッセージを取り寄せる際に **POP**(Post Office Protocol) を使う。
- メッセージは常時サーバ内に格納しておき、ユーザは読みたいメッセージだけ取り寄せて眺める — MTA と手元の MUA の間での通信に **IMAP**(Internet Message Access Protocol) を使う。

POP ではメッセージが手元のマシンにあってすぐ探せ、サーバの通信が受信/送信時の短時間で済みます。一方、あるマシンで読んだメールは手元のマシンに移動してしまうので、後で別のマシンから読むことはできません。IMAP はそのような不便がない代わりに、ネットワーク経由でサーバとつながっていないとメッセージを見ることができません。

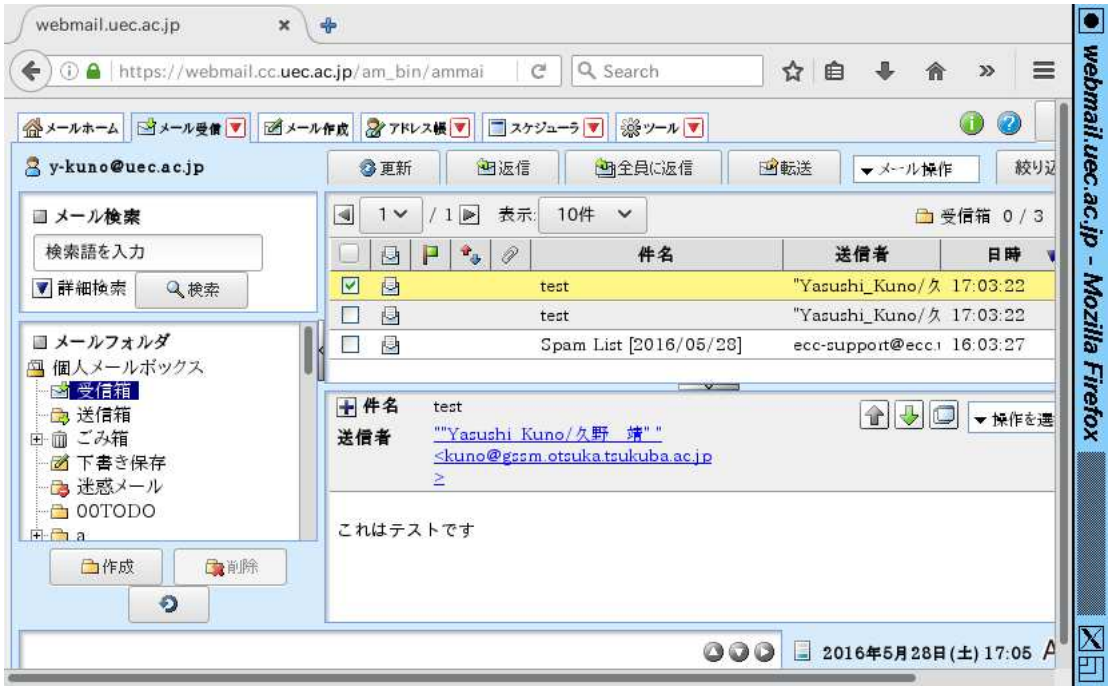

図 8: Web メールの画面

MUA は従来は PC やスマホ上で動くアプリケーションが主流でしたが、最近は Web アプリが普及してきたため、Web アプリとして動作する MUA を使うことが増えてきました。これを **Web メール**

[8]ただし、メール伝送の時は DNS 上で「MX(Mail Exchange)」レコードと呼ばれるメール専用種別のデータを検索します。これは、メールアドレスの場合、あるアドレス (つまりメールサーバ) が停止していたら代替のサーバに送るなど、メール固有の扱いをする場合があるためです。

と呼びます。この科目でも、電子メールの読み書きには http://webmail.cc.uec.ac.jp で利用できる ActiveMail という MUA を標準として使用します (図 8)。[9]

5.2 メールメッセージの形式 exam

個々のメールメッセージには、送信者 (From:)、件名 (Subject:)、日付 (Date:) などの情報があり、そして本文があります。通常 MUA でメッセージを見ているときはこれらは別々に見えていますが、実際にインターネット上でメッセージが送られているときは **RFC822** と呼ばれる形式でこれらの情報が一緒になって送られます (図 9)。

RFC822 形式メッセージの冒頭には**メッセージヘッダ**と呼ばれる部分があり、ここに上述のもを含めた各種付加管理情報が格納されています。その後ろにメッセージ本文がありますが、ヘッダと本文の間は 1 行の空白行 (長さ 0 の行) で区切られています。

ヘッダの情報を見ると、メッセージがいつどこで投入され、どのように中継されてきたかが分かります。メールヘッダの一例を見てみましょう。これはあくまでも例なので本文が 1 行しかありません

```
Return-Path: <kuno@gssm.otsuka.tsukuba.ac.jp>
X-Original-To: y-kuno@uec.ac.jp
Delivered-To: ka002689@uec.ac.jp
Received: from localhost (localhost [127.0.0.1])
    by mx-east.uec.ac.jp (Postfix) with SMTP id 49A144E348E
    for <y-kuno@uec.ac.jp>; Sun, 29 May 2016 12:26:59 +0900 (JST)
Received: from utogwpl.gssm.otsuka.tsukuba.ac.jp (utogwpl...[210.154.96.162])
    by mx-east.uec.ac.jp (Postfix) with SMTP
    for <y-kuno@uec.ac.jp>; Sun, 29 May 2016 12:26:58 +0900 (JST)
Received: (qmail 17906 invoked from network); 29 May 2016 03:26:58 -0000
Received: from OneOfLocalMachines (HELO gssm.otsuka.tsukuba.ac.jp) (127.0.0.1)
    by localhost with SMTP; 29 May 2016 03:26:58 -0000
From: "Yasushi_Kuno" <kuno@gssm.otsuka.tsukuba.ac.jp>
To: y-kuno@uec.ac.jp
Subject: test
Mime-Version: 1.0
Content-Type: text/plain;charset="ISO-2022-JP"
Date: Sun, 29 May 2016 12:26:58 +0900
Sender: kuno@gssm.otsuka.tsukuba.ac.jp
Message-Id: <20160529032659.49A144E348E@mx-east.uec.ac.jp>
                ←この 1 行がヘッダと本文の区切り
これはテストです
```

図 9: SMTP によりやりとりされる RFC822 形式メッセージ

が、実際にはもちろん本文の方に肝心の用件が書かれています。

ヘッダは「フィールド名: 値」の形の行が並んだもので、Subject:や From:などの情報もすべてヘッダフィールドとして伝達されています。ただし、誰が送信者かの情報はこれとは別に SMTP プロトコルでも伝達しています。この情報を**エンベロープ From** と呼び、最後に Return-Path:ヘッダに格納されます。また、Received:ヘッダを見ると、メッセージがどのような経路を通って中継されてきたかを追跡できます (途中のサーバがここに嘘を書き込んでいない限り)。

上の例のメッセージは久野が筑波大内から電通大の自分のアドレスにあてて送信したものです。この場合は Received:が 4 つありますが、受信相手が「127.0.0.1」のものは同一マシン内での受渡しなので、中継しているメールサーバは 2 つです。

[9] Web メールのサービスを提供しているマシン (Web メールサービスの観点からはサーバ、電子メールサービスの観点からは MUA なのでクライアント) はネットワークに常時つながっていますから、Web メールの MUA では MTA からの受け取りは IMAP を使うのが普通です。

なお、RFC822 はあくまでもこのような単純な文字の並びから成るメッセージを格納する規格なので、音や画像など、また場合によっては日本語の長い行など、この規格に収まらないデータを送る場合はそのデータをまずビット単位のデータと考えてから英字の列に変換 (**符号化**) して送ることになります。このような多様なデータを送る際の約束は **MIME**(Multipurpose Internet Mail Extension) と呼ばれる規格で定められています。

演習 2 ブラウザで `http://webmail.cc.uec.ac.jp` を開き、ログインしなさい。「メール作成」を選び、宛先「自分の `ID@uec.ac.jp`」に簡単なメールを送りなさい。次に隣に座っている人の ID を尋ね、同様に送りなさい。送れたら、以下の演習を 1 つ以上やってみなさい。

a. 「メール受信」を選択し、送ったメールが読めることを確認する。さらに本文表示領域の上の「操作を選択」から「ソース表示」を選び、RFC822 形式メッセージ全体を表示する。どのようなメッセージヘッダがあるか、そこにどんな情報が含まれているかを検討する。

b. 普段自分が使っている (電通大以外の) アカウントから「自分の `ID@uec.ac.jp`」に簡単なメールを送りなさい (または知合いに送ってもらうのでもよい)。送られたメッセージについて上と同様にメッセージ全体を表示し、上とはどのような違いがあるか、どのような経路でメッセージが送られて来ているかを中心に検討しなさい。

c. 今度は件名が日本語だったり、短い/長い日本語の行が入っていたり、画像などの添付されたメールを同様に送りなさい。送られたメッセージについて上と同様にメッセージ全体を表示し、上とはどのような違いがあるか、データがどのような形で送られて来ているかを中心に検討しなさい。

電子メールで注意するべきことは、「内容は (暗号を使わない限り) 覗き見られたり改ざんされる可能性が常にある」点です。前者は、世界中のメールサーバは基本的に RFC822 形式のメッセージを中継するので、サーバの管理者が中身を見るのは容易だからです。後者も、サーバは RFC822 形式に合致したメッセージであれば単純にそれを受け入れて宛先に配送するので、「中身が本もの」という保証は何も無いことによります。ですから、そのメッセージに嘘がないかどうかの判断規準は基本的に「知っている相手とのやりとりで」「その相手がいつも言っていることと矛盾しないことを言っているか」に尽きるわけです。

6 ネット上のコミュニケーション

6.1 ネット上のコミュニケーションが持つ性質

ここまでで電子メールや Web アプリケーションについてひととおり見て来ました。過去においてはネット上のコミュニケーションは電子メールや**メーリングリスト** [10] が主流でしたが、今日では Web アプリケーションから発達してきた掲示板 (2 ちゃんねる等)、SNS(Facebook, Twitter, Instagram、…)、メッセージ交換サービスの比重が高まって来ています。そして、これらの上で毎日膨大な量のトラブルが発生し、多くの人が嫌な目にあったり時間を無駄にしています。このような問題に巻き込まれないようにすることも、この科目で学んで頂きたい重要な内容です。

ここではまず、ネット上のコミュニケーション手段一般について考えてみます。ネット上のコミュニケーション手段は一般に次のような性質を持つと言えるでしょう。

- 文字が中心である — 通話や動画もありますが、やはり文字によるメッセージ中心になります。そして文字だとつい「言葉足らず」になりやすく、その結果誤解や食い違いを生みがちです。
- 手段ごとに特性が異なり、それによる影響がある — たとえば電子メールに基づくコミュニティでは 1 つのメッセージを全員が読むまでに時間が掛かるので「ゆっくり」議論をするような慣

[10] メーリングリストサーバのメールアドレスにメッセージを送ると全参加者にそのメッセージが転送される仕組みで、これによって全員による情報交換が可能になります。

習ができます。逆にチャットや Twitter のように短いメッセージを前提とした場では頻繁なやりとりが行われ「その場かぎり」の話題が中心となります。

- コミュニティごとに「常識」が異なる — たとえば巨大掲示板サイトの技術的なテーマを題材としたあるスレッドでは、「初心者です」という書き出しは良くないとされていました。それは「初心者を免罪符に自分で調べることなく質問してくる」人がいたためですが、このように一般には問題ない行動でも場によっては問題とされることがしばしばあります。[11]
- 炎上が起きる — ネット上には他人を攻撃することを趣味にする人が多数いて、問題行動や問題発言を見つけるとその場に集まって来て非難の書き込みを集中させたりします。
- 匿名による勘違い — ネット上の場では「自分が誰だか明かさずに発言できる」ものも多くあり、自分が誰だか知られないという安心感から問題行動や問題発言がよく見られます。[12]
- 忘れてもらえない — 日常ではなにか失言をしてもその場にいた人の記憶が薄れるまで待てばあまり問題にならないのですが、ネット上の情報はコピーが簡単なため「記録」されてしまいやすく、そうなるといつでも蒸し返されてくる恐れがあります。[13]

6.2 電子メールに関する留意事項

電子メールは今日では高校生以下にはあまり使われていませんが、逆に大学や社会では広く使われており、社会に参加する上で不可欠となっています。そこで知らないままに「LINE や SNS ののりで」メールを使って失敗する例が多く見られます。次のことを注意しましょう。

- 内容を表す件名 (Subject:) を記入する — 忙しい人は日に何十もメールを受け取るので、MUA に表示される件名で内容を把握できないと大変迷惑になります。[14]
- 必ず名乗り丁寧な言葉づかいをする — 出だしで「○○大学×年の△△です。」のように自分が誰かを知らせるようにします。こうすることで、本題の予測がつけやすくなり誤解を防げます。本題も丁寧な言葉づかいで簡潔に用件を述べるようにします。最後は「よろしくお願いします。」などでしめくくればよいです。メールは目上の相手に出す機会が多くなるため、常に丁寧な言葉づかいを使う習慣をつけた方が結局得です。
- 添付ファイルは注意して使う — メールは大量データの扱いには向いていないため、大きなデータ (写真や圧縮アーカイブなど) を添付することは避けるべきです。また受け取る際も知らない相手からの添付ファイルはマルウェアの可能性が高いので開いてはいけません (ということは、自分の添付ファイルもそう扱われる可能性があることを理解するべきなわけです)。

6.3 SNS などのコミュニケーションサービスに関する留意事項

Twitter などに代表されるコミュニケーションサービスは高校生から大人まで広く使われていますが、とくに大学生や若者について大きな問題がしばしば起きています。次のことを注意しましょう。

- 投稿内容は仲間内だけでは済まない — ふだん何も問題がない内容をやりとりしている間は、そのメッセージに興味を持つのはあなたの仲間だけなので、仲間しか読まないと勘違いしがちです。しかし実際には他人でも読める場合が多く、何も反応がなくても「あの人は実はこんなことを言ってる」と影ながら思われている可能性があります。

[11] このため、ネット上の場に参加する前にまず 1 か月くらいは「ROM る」(書き込まずにやりとりを読んで勉強する) ことがよいとされます。

[12] しかし一旦炎上が起きると、「発言者が誰だか特定する技能に長けた人」がやってきてあちこちの情報をもとに実名や住所などを明らかにしてそれを勝手に公開することが多く起きています。

[13] 米国で自分で SNS に貼った「恋人との性的な写真」が学校に見付かった保育専門学校の女子学生が退学になった例があります。

[14] 内容を表す件名としてダメなものは「こんにちは」のように情報の無いものです。よい例は「○月○日の××の授業内容の質問」のように本題を表すようなものです。

- SNS の「友人のみ」は保証されないと思うべき — SNS ではメッセージの見られる範囲を「友人のみ」「友人の友人」などに限定できますが、それを見て広めたいと思った「友人」がコピーすればいくらでも広められます。ですから、いちど投稿したらその内容は全世界から見られるものだと考えておくべきです。[15]
- 起きる可能性のある悪いことは必ず起きると思え — たとえばあなたがネットに悪ふざけの写真を流そうとしたとして、その前に必ず「これを親が/学校が/将来の就職希望先が見たら」と想像するべきです。実際にプライベートな悪ふざけ写真のために希望する職業の道をあきらめざるを得なくなった若者は多くいます。
- 「炎上」は大きなダメージ — 「問題発言」「問題写真」などを投稿したとたん、それを見つけて炎上させる人が一斉にやって来ますし、匿名のはずでも「名寄せ」などの技術であなたが誰かを明らかにし、まとめサイトに載せたりします。ネットに投稿するときには、常にその危険を想像した上で「問題ない」と思うことだけにとどめるべきです。

演習 3 ネット上で次のような問題事例を 1 つ以上探して報告しなさい。探した事例についてレポートを見る人が検証できるように URL を明記し、起こったことを分かりやすく整理して示すこと。また、「何が原因で悪い結末に至ったか」をきちんと考察すること。

a. 2 ちゃんねるなどの掲示板サイトで、最初は楽しく対話ができていたのに、途中から喧嘩になって後味悪い結末となった事例。
b. Twitter や SNS などのサイトで、仲間内だけだと思って気軽に書いた内容が想定しない相手に見られていて不幸な結末となった事例。
c. SNS やブログなどでの発言に対して、非難が集中して炎上や本人特定などに至った事例。

本日の課題 3A

本日の課題は「演習 1」「演習 2」「演習 3」に含まれる小問 (合計で 9 個) の中から 1 つ以上を選択し、結果をレポートとして報告して頂くことです。LMS の「レポート # 3」の入力欄に直接入力してください (別途作成したものをコピーペーストで貼っても構いません)。以下の内容がこの順に含まれるようにしてください。

- 冒頭に題名「コンピュータリテラシレポート # 3」、学籍番号、氏名、提出日付を書く。(グループでやったものはグループのメンバー全員の氏名も別途書く。)
- 課題の再掲を書く (どんな課題であるかをレポートを読む人が分かる程度に要約する)。
- レポート本体の内容 (やったこととその結果) を書く。
- 考察 (課題をやった結果自分が新たに分かったことや考えたこと) を書く。
- 以下のアンケートに対する回答。

Q1. セキュリティ、WWW、電子メール、ネットコミュニケーションの難しさについてどれくらい知っていましたか。新たに知ったことで面白かったことは何ですか。
Q2. 電子メールの仕組みについてどのように思いましたか。
Q3. リフレクション (今回の課題で分かったこと)・感想・要望をどうぞ。

なお、課題はグループでやって構いません。その場合も、(メンバー全員の氏名を明記した上で) レポートは必ず各自で執筆してください。レポート文面が同一 (コピー) と認められた場合は同一であると認めた全員について点数にペナルティを科すことがあります。

[15] 「ネットに投稿する前に、そのメッセージを紙に書いてあなたの家の玄関に貼ってもよいか想像してみよう。そうしてもよいと思うなら、投稿してもよい」という指針があります。