セキュリティーとプライバシー

セキュリティーと辞書で引くと

- 安全, 無事
  - public ～ 治安, 公安/in ～ 安全に, 無事に.
- 安心, 心丈夫
- (財政上の)安定, 保障:⇒social security.
- 〔危険･危害などに対する〕防衛(手段), 警備(態勢), 安全保障〔against, for〕

- (負債の支払いに対する)保証, 担保, 抵当(物件)、保証人、有価証券:government securities 政府発行の有価証券(e.g., 国債･公債)

セキュリティー

- 経済面での観察
  1. (とても) お金がかかる

     完全性が保証できない。。。。
  - お金を払いたがらない

    事故が起こらないと必要性を認識しない。

    ➔ 適度な事故が発生しないとお金を払わない。

- ということで、セキュリティーのビジネスが成立する社会 ＝ “適度に” 事故がある

セキュリティー

- 必要なこと / 実現すべきこと

守るべき物に関して、以下の2つを実現。

1. 破壊されない

   再生コストを必要とする事故

2. 盗難されない

   不正利用される

   a. 再生コストを必要する 事故

   b. 他人のコスト負担を必要とする事故

セキュリティー

- 例えば、デブ への生命保険
  - 困ること
    - 死亡 ➔ 収入の喪失
    - 疾病発生➔治療費

    (*) つまりは、収入の減少
  - 保障すべき物
    - お金
  - 最近保険会社がやっていること
    - デブにならないようにする方法を奨める

      i.e., 予防策の伝授 ➔ 保険利用の確率を下げる。

セキュリティー

- 可能な対策
  1. (修復)コストを保障/補償 (保険, insurance)
  2. 事故を未然に防ぐ (Pro-active)
  3. 事故発生時の対応策
     a. 事前策 (Pro-active)
     b. 事後策 (Re-active)

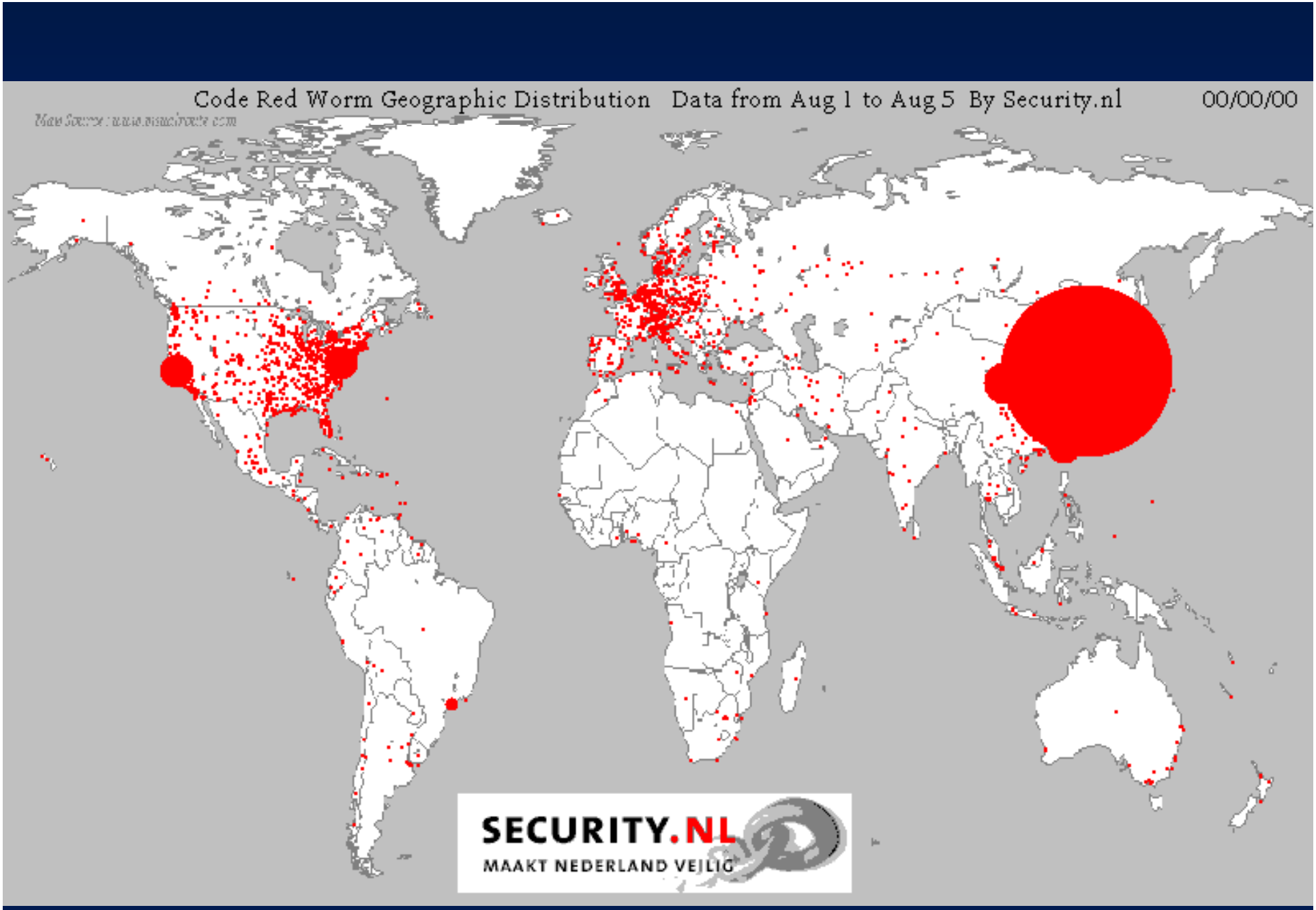
- 注意すべき点
  1. 完全性は要求不可能
     (*) 事前、事後の両面において
  2. “完全な” 社会ではセキュリティービジネス自体が成立しない

いろいろなモデルがある。。。。

- SLAMERの例からのレッスン
  - 先進国(e.g., 北米/欧米/日本)
    - 合法ライセンスが90%
  - 発展途上国(e.g., 韓国)
    - 違法ライセンスが90%

      (*) Windows Update Patchなし。。。。。
  - 後進国 (e.g., 中国)
    - 徹底した違法ライセンス

      (*) Windows Update Patch込みの違法コピー


Code Red Worm Geographic Distribution Data from Aug 1 to Aug 5 By Security.nl
00/00/00
SECURITY.NL
MAAKT NEDERLAND VEILIG

いろいろなモデルがある。。。。

- オープンソース至上主義
  - すべての ソフトウェアは、オープンソースでなければならない。
  - ブラックボックス VS オープンソース

    (Microsoft)　　(Linux)
  - ブラックボックスがないとセキュリティーは実現不可能
  - ブラックボックスだと、手が出せない
  - オープンソースは、誰でも設計図を持ってしまう。

分かること。。。。

- 中途半端は、一番 怪我をする。
- 中庸は、デジタル社会では、罪悪。

ところで、コストはどうなる？

- 手段の共通化共有化が容易
  - 陽: 守りの強さを常時同レベルに維持可能
  - 陰: 攻めるための情報
- 鍵 の複製コストが低下
  - 陽: いつでも、鍵は変えられる
  - 陰: 鍵は すぐに 複製され流通可能

攻撃者は誰?

- 組織内部：大半
  - 社員は信用できない。
- 組織外部：少数
- 保守的人種 (組織の内外を問わず)
  - 新しいもの(e.g.,技術)には危険/リスクはつきもの。。。。。でも、現状を維持したがる。

プライバシー

プライバシー

- (他人からの干渉を受けない個人の)私生活
- 秘密, 内密
- 隠遁(いんとん)
- Private の名詞形

かなり、主観的な面が強い。。。。。
➔ 結局は、セクハラと 類似している

プライバシー

- デジタル技術の効果
  - 複製
  - 流通
  - 共有
  - 加工

プライバシー

- 結局は、セクハラ、アカハラ、ドクハラ と似ている。
- 同じ事象でも、問題がない場合と、問題になる場合とがある。
- 基準も変化していく
- 必ず 騒ぎ出す ひとがいる。
  - 最初から話をするべし！

プライバシー

- 対策は、責任の所在を明らかにすること
- 基本的には、個人で守るしかない。
- しかし、個人情報に 合った サービスを 顧客はありがたがる。
  - ➔ 情報の 2次流通、2次利用に関するコンセンサスの必要性

プライバシー

- 結局は、誰が(Who)、何の目的で(Why)、どのように(How & Where)、いつ(When) 個人の情報を利用するか。
  - ➔ なんだ、5W 1H じゃん。
- デジタルもアナログも基本的には同じ。デジタルは、情報の流通速度を大きくする。

プライバシー

- ところが、、、、、
  - 情報理論:

    秘密の情報(＝貴重な情報) ほど、高い価値がある。
  - プライバシー

    公然の事実（＝普通の情報）になると、プライバシーではなくなる

前半のまとめ

- セキュリティー
- プライバシ

- 過剰反応は無意味
- “適度”でないとビジネスにはならない
  - 極端では、ビジネス上の意味を持たない。

ネットワークセキュリティー

実世界の“価値ある”情報がネットワーク上に存在し交換流通されている。

→ 情報の保護、利用者の認証、不正アクセスの防止
サービスの継続

保護対象；

(1) データ
  (a) 機密性(他人に知られたくない情報)
  (b) 保全性(他人に変更されたくない情報)
  (c) 可用性(管理を必要とする情報)
(2) 資源 (CPU、メモリなど)
(3) 評判
(4) サービス

インターネットシステムの構造と精神

- One for All, All for One
  - みんなへの貢献 ←→ みんなからの貢献
- 善人ばかりのネットワークだった。。。。。
  - 悪人(Intentionally)が増えてきた
  - おばかさん(Accidentally) が増えてきた
- 隣人を信じなさい
  - 騙されやすい 通信プロトコル
    (aka Quality向上がやりやすい, e.g., Proxy, Redirection)

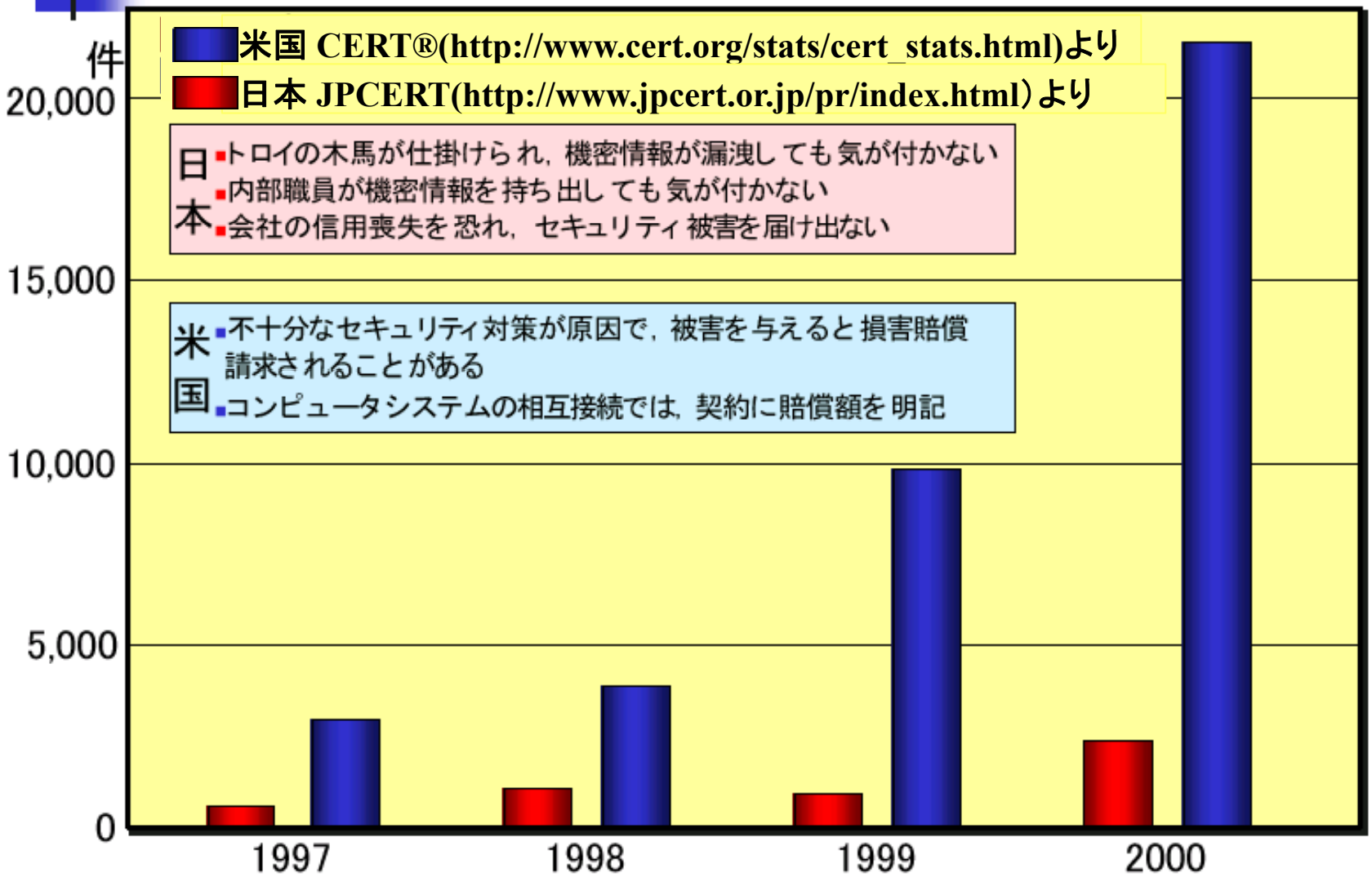
次世代インターネットのQoS

- Quality of Service (QoS) の定義
  - いわゆる Quality of Service
    - End-to-Endに提供される信号品質
      (e.g., Latency、RTT、帯域幅、エラー、紛失)
    - End-to-Endへの挑戦
      (e.g., CDN、chache、Proxy)
  - 信頼性/堅牢性としての QoS
    - Compliant Users
    - Malicious Users

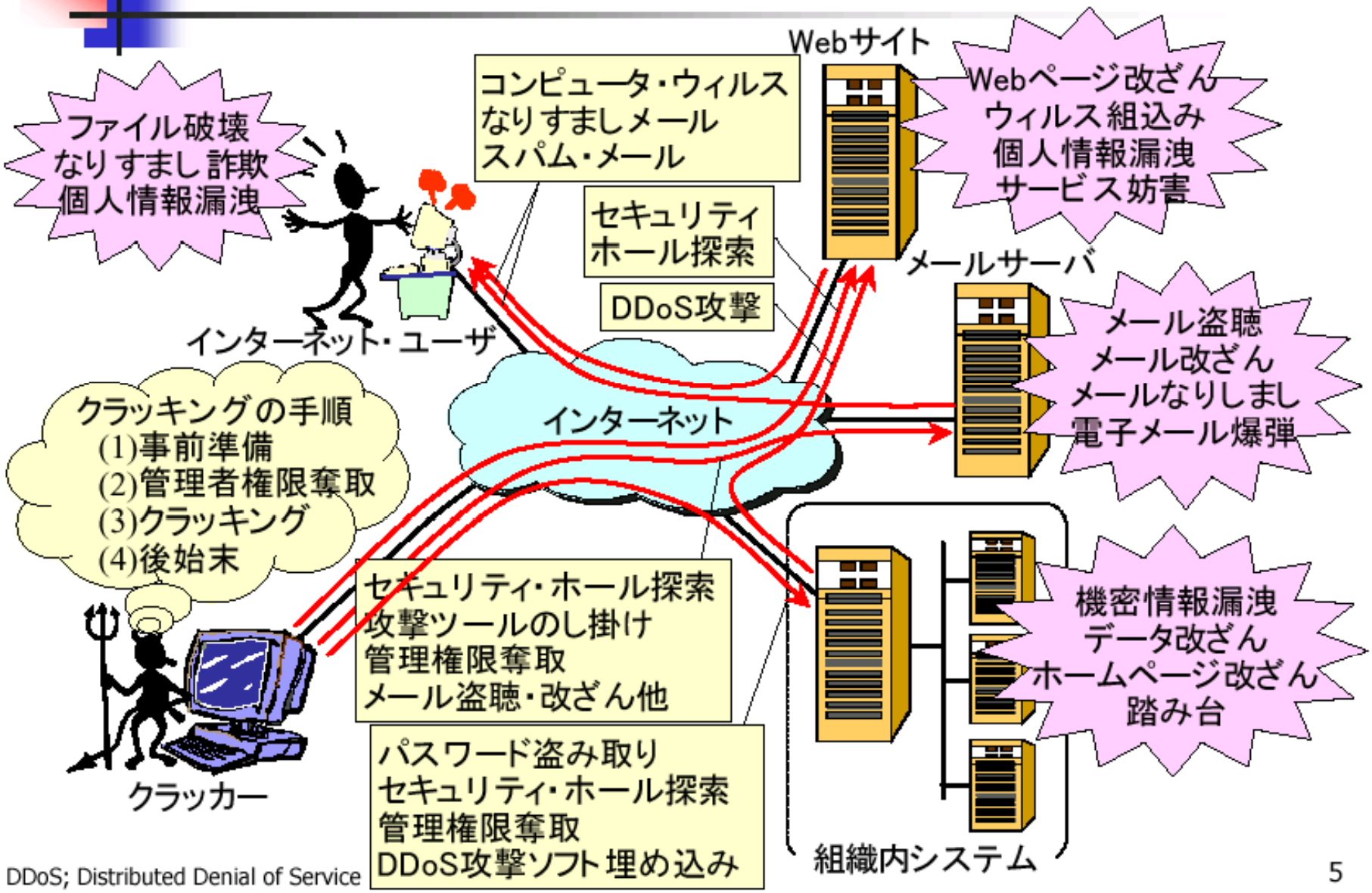
不正アクセス行為とクラッカー

- 不正アクセス行為/クラッキング行為
  - 計算機やインターネットの知識を悪用して、許可されていない計算機に侵入しデータを改竄したりサービスの継続を妨害したりすること
- クラッカー
  - 上記のような行為を行う人
- ハッカー
  - コンピュータやネットワークの動作や構造を深く理解することに喜びを覚える人々。

不正アクセス被害届け出件数

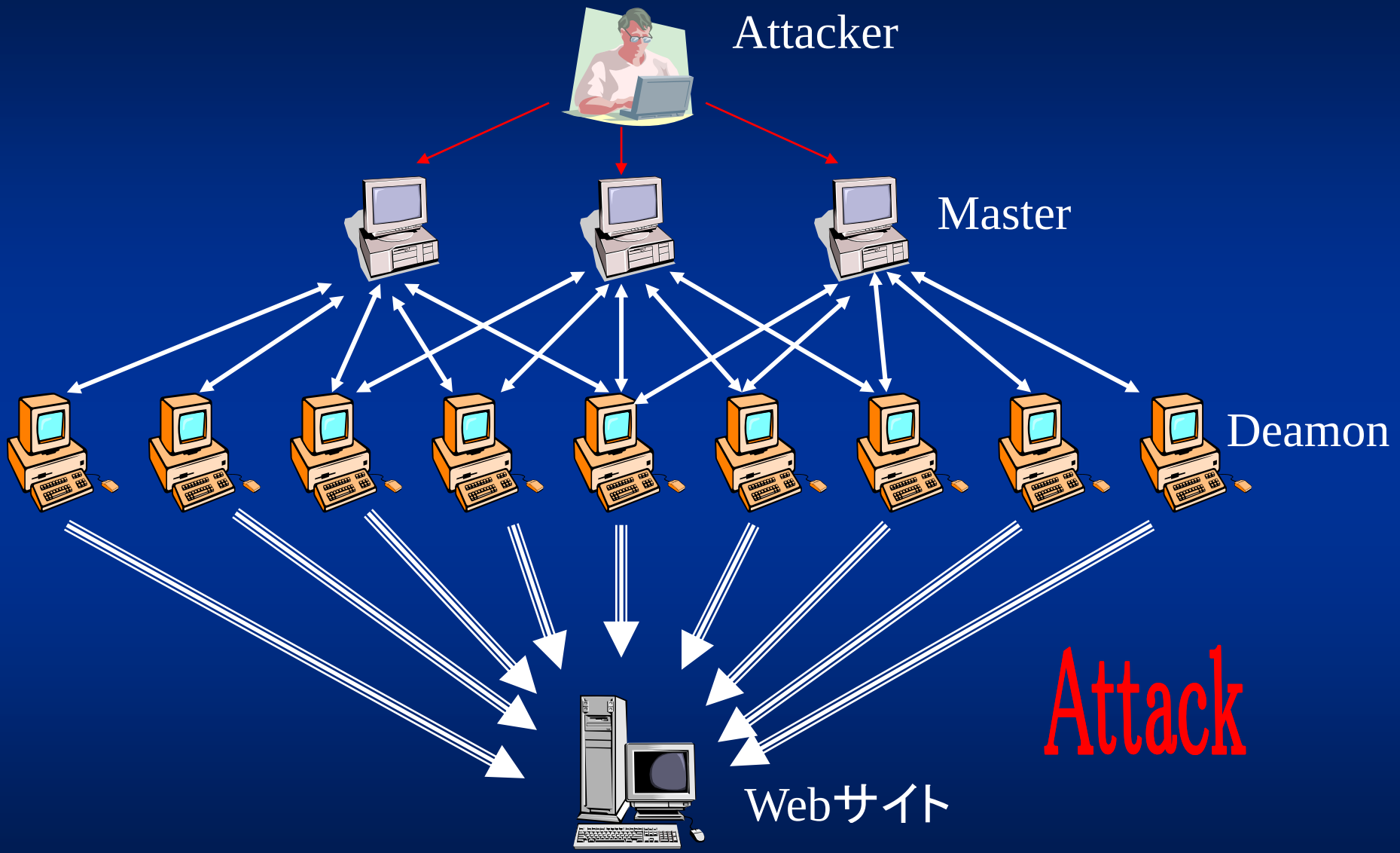


不正アクセス行為の手口と悪事

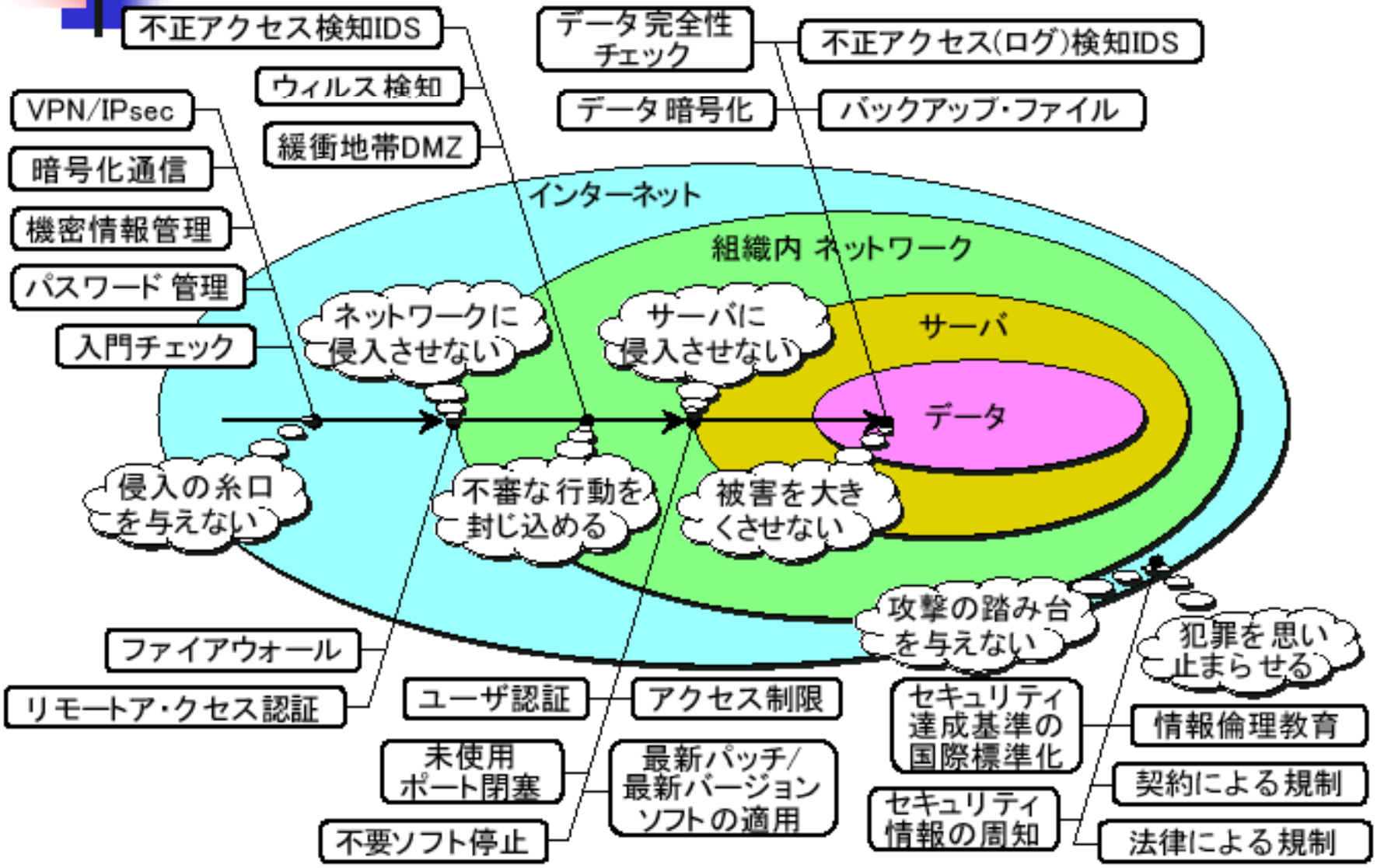


DDoS; Distributed Denial of Service

DDoS 攻撃の仕組み



セキュリティ防衛策

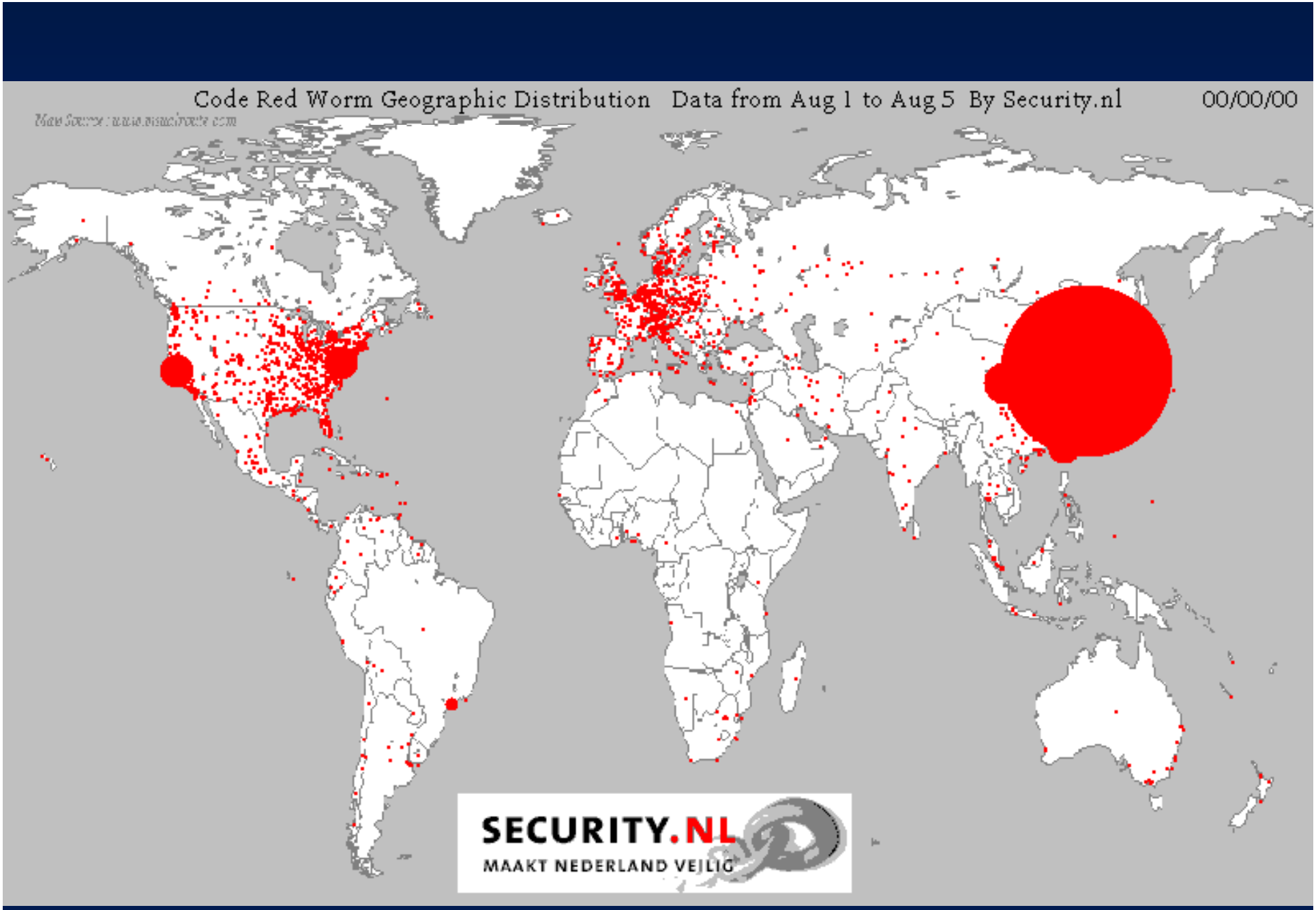


IDS; Intrusion Detection System, DMZ; DeMiliarized Zone

他のプログラム依存/自己増殖型の不正プログラム

最近のウィルス/ワームとその対策

- コードレッドⅡ(‘01.7)，ニムダ(’01.9)
  （ワーム／ウィルス）
  - 感染力強く(複数の感染ルート)，変幻自在
  - パターンファイル流布前に，感染が拡大
  - インターネットからの攻撃防衛だけでは不備
    例) 感染したモバイルPCから社内ネットワークに侵入
- 対策のポイント
  - 早期発見：異常トラフィック，不信メール/ファイル・・・・
  - 正確な情報の早期入手(アンチウィルスベンダーから)
  - 感染経路遮断：ケーブル引抜，プロトコルの制限・・・・
  - 対応態勢(通常/警戒/非常事態)の整備
  - 多くはMS系ソフトを攻撃対象⇒UNIX系マシン？


Code Red Worm Geographic Distribution Data from Aug 1 to Aug 5 By Security.nl
00/00/00
SECURITY.NL
MAAKT NEDERLAND VEILIG

ネットワークセキュリティー

1. セキュリティーポリシーを決める
2. ユーザーの認証(パスワード管理)
3. ファイルの保護
4. Secureなホストへの アクセスの方法
5. アクセス制御(xinetd，TCP wrappers)
6. 暗号化 (IPsec)
7. Firewall (e.g., socks)
8. 電子メールシステム

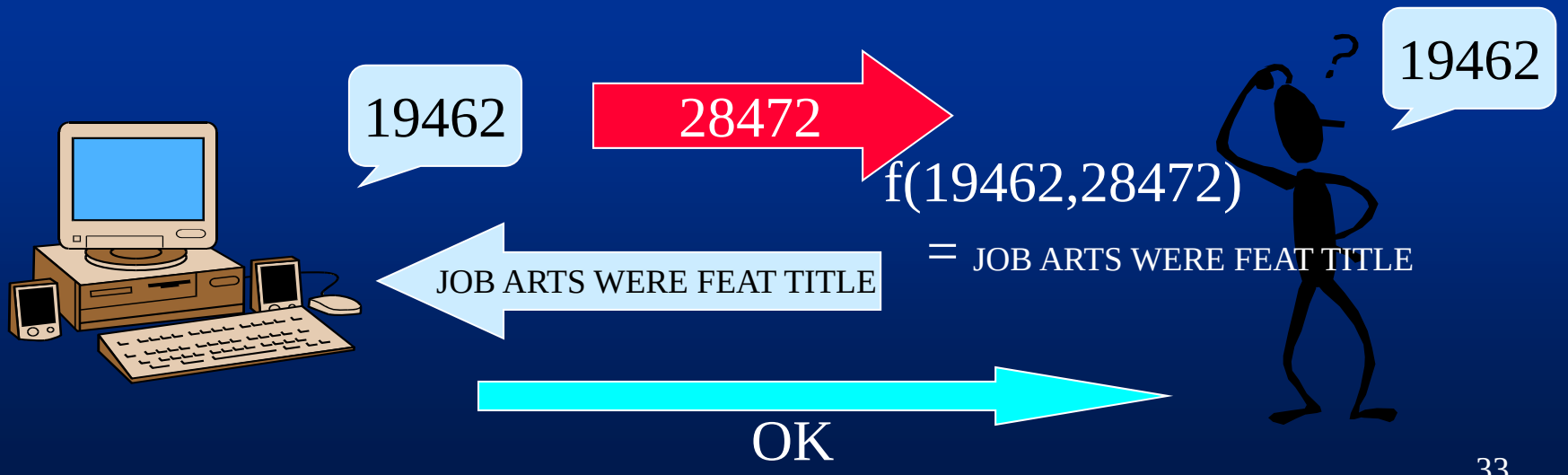
ユーザの認証(アカウント管理)

1. 想像しにくいパスワードの使用
   -すべて小文字、すべて大文字とか、、、、
2. パスワードファイル(/etc/passwd)の保護
   - Shadow Password File ; /etc/master.passwd
3. 使い捨てパスワード(OTP; One Time Password)
   - Challenge & Response 型 with 共有鍵
4. ssh ; Secure Shell
   - 暗号化された Remote Login with 公開鍵+秘密鍵
5. RADIUS (Remote Authentication Dial In User Services)
   - ダイアルアップユーザに対する認証処理

使い捨てパスワード；OTP

- 毎回異なるパスワードを利用する。
- Challenge & Response 型の認証
- 同一の鍵をサーバーとクライアント間で共有する。
- 例えば、

  ftp://ftp.nrl.navy.mil/pub/security/opie/opie-2.3.tar.gz
- Windowsのクライアントソフトもある。



安全なRemote Shell
- ssh ； secure shell -

- 公開鍵暗号方式；公開鍵＋秘密鍵
- フィンランド ヘルシンキ大学で開発
- RSA、IDEA、DES 方式による暗号化
- 例えば； http://www.cs.hut.fi/ssh
- Windows版クライアントもあります(Teraterm)
- sshd (secure shell daemon)
- ssh (rlogin)、scp (rcp)、ssh-keygen
- 鍵情報ファイル

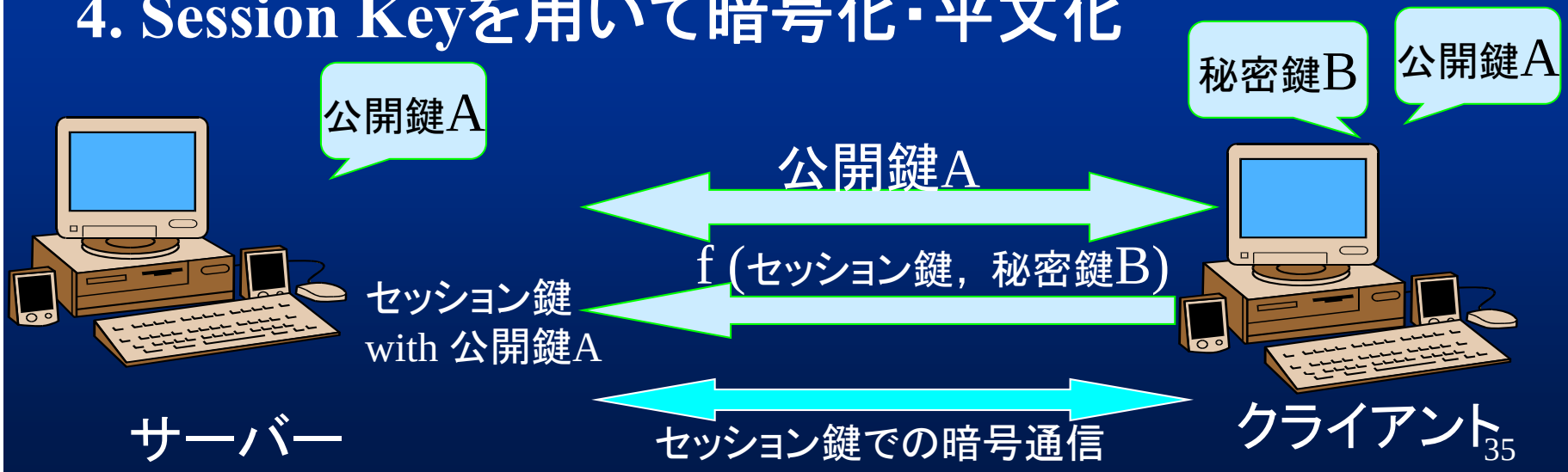
  .ssh/identity

  .ssh/identity.pub (client) => .ssh/authorized_keys (server)

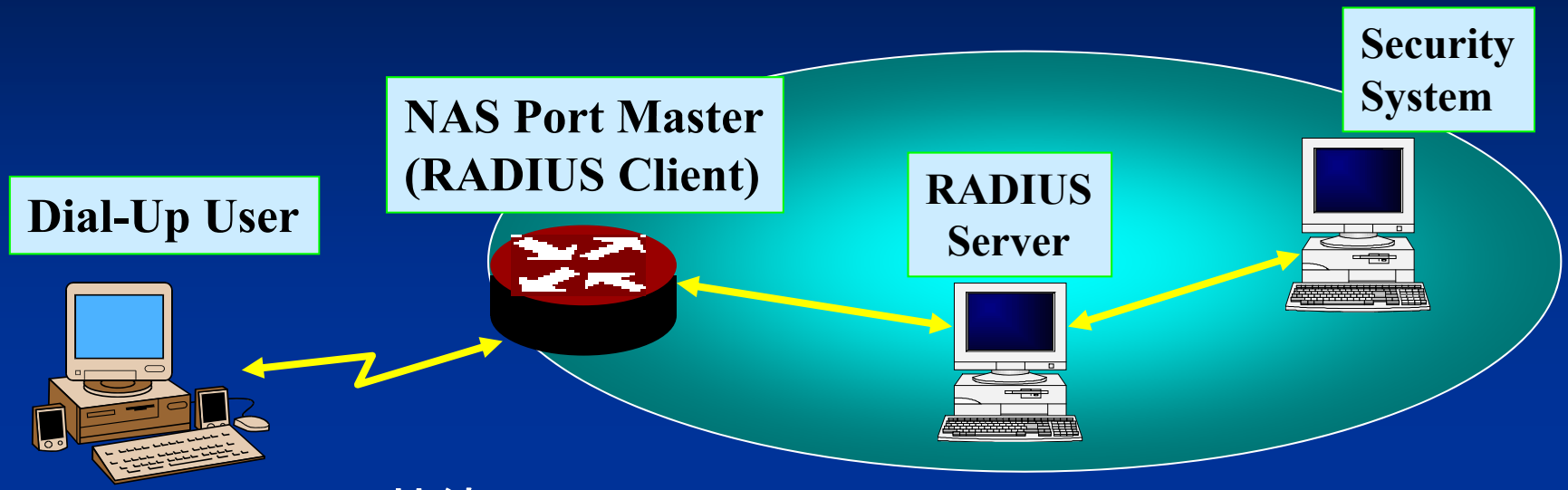
  .ssh/ssh_known_hosts, /etc/ssh_known_hosts

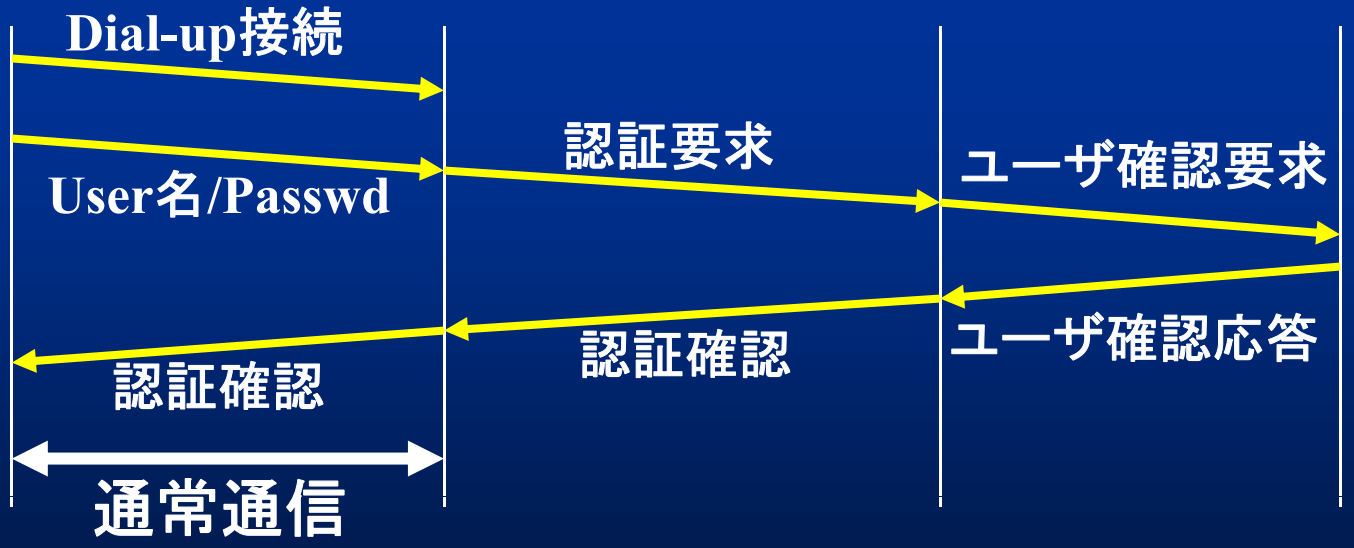
安全なRemote Shell
- ssh ； secure shell -

1. 公開鍵(Public key)の交換とチェック
   /etc/ssh_known_hosts、..ssh/known_hosts
2. クライアントは秘密鍵を使って Session Keyを送る
   (*) Session keyは毎回変わる。
3. サーバは公開鍵を使って Session Key を平文化する
4. Session Keyを用いて暗号化･平文化



RADIUSの動作


Dial-Up User
NAS Port Master
(RADIUS Client)
RADIUS
Server
Security
System


Dial-up接続
User名/Passwd
認証要求
ユーザ確認要求
ユーザ確認応答
認証確認
認証確認
通常通信

ネットワークセキュリティー

1. セキュリティーポリシーを決める
2. ユーザーの認証(パスワード管理)
→ 3. ファイルの保護
   (i) ファイルアクセス
   (ii) Freeソフトのインストール
4. Secureなホストへの アクセスの方法
5. アクセス制御(xinetd TCP wrappers)
6. 暗号化 (IPsec)
7. Firewall

ファイルアクセス権

3つのレベルのファイルアクセス権

(1) Global, Group(SGID), User(SUID)

(2) Readable, Writable

・root でのアクセスを極力防止する。

例；

ユーザ hiroshi の .login が Group (elab) でwritable。

/etcが Group Ownerにとってwritable (root&sysem)

→ elab グループの誰かにloginされると、hiroshi の .loginに次のコマンドを潜り込ませることが可能。

```
rm -f /etc/passwd
cp /tmp/data526  /etc/passwd
```

フリーソフトウェアの導入

フリーソフトの導入を行うときには、次のようなポイントに注意してインストール作業を行う方が良い。

(1) 特権ユーザ(e.g., root)としてはテストを行わない。
(2) ソースコードをコンパイルして使用する。
　　＝ バイナリコードは使用しない。
(3) Archiveをunpackするときは、内容を確かめてから（`tar tf file`）
(4)テンポラリーなディレクトリーでの unpack
(5)file コマンドでファイルの内容のチェック
　　（バイナリファイルは注意）
(6)ソースコードを眺める

フリーソフトウェアの導入

（7）makeファイルを眺める。make －n
　　変なディレクトリーをアクセスしていないか
（8）string コマンドでObjectファイルをチェック
（9）可能な限りlocalファイルでテストする
（10）make －n install でmakeプロセスをチェック
（11）SUID、SGIDコマンドがあったら、要注意

ネットワークセキュリティー

1. セキュリティーポリシーを決める
2. ユーザーの認証(パスワード管理)
3. ファイルの保護
→ 4. Secureなホストへの アクセスの方法
   - Secure Shell； `ssh`
5. アクセス制御(xinetd，TCP wrappers)
6. 暗号化 (IPsec)
7. Firewall

ネットワークセキュリティー

1. セキュリティーポリシーを決める
2. ユーザーの認証(パスワード管理)
3. ファイルの保護
4. Secureなホストへの アクセスの方法
→ 5. アクセス制御(xinetd，TCP wrappers)
   (i) 不要なサービスをシャットダウン
   (ii) ログをとるプログラム(tcpd)
   (iii) ホスト・サービスのアクセス制御
6. 暗号化 (IPsec)
7. Firewall

アクセス制御

1. 必要なサービスのみにしてしまう。
   ＝ 不要なサービスはシャットダウンする。
   ＝＞ /etc/inetd. conf の不要な行をコメントアウト

   （1）共通；conmsat，finger，ntalk，
   （2）クライアント; pop，imap
   （3）ssh導入時；login，exec，shell，ftp

アクセス制御

2. *tcp_wrapper*

http://csrc/nist.gov/tools/tools.htm

(1) */etc/inetd.conf*

Before;

```
#service socket protocol wait?  User  program           arguments
ftp      stream  tcp    nowait  root /usr/sbin/ftpd       ftpd
telnet   stream  tcp    nowait  root /usr/sbin/telnetd    telnetd
shell    stream  tcp    nowait  root /usr/sbin/rshd       rshd
login    stream  tcp    nowait  root /usr/sbin/logind     logind
```

After;

```
#service socket protocol wait?  User  program           arguments
ftp      stream  tcp    nowait  root /usr/sbin/tcps       ftpd
telnet   stream  tcp    nowait  root /usr/sbin/tcpd       telnetd
shell    stream  tcp    nowait  root /usr/sbin/tcpd       rshd
login    stream  tcp    nowait  root /usr/sbin/tcpdd      logind
```

(2) inetd のreread

```
# kill -HUP pid-of-inetd-process
```

アクセス制御

3. ホスト・サービスのアクセス制御

(1) /etc/hosts.allow

```
fingerd       : ophelia hamlet laertes
rshd,rlogind: LOCAL EXCEPT hamlet
telnetd,ftpd: LOCAL, .expcons.com, 192.1.4
```

(2) /etc/host.deny

```
tftpd : ALL : (/usr/sbin/safe_finger -l @%h |
                /usr/sbin/mail -s %d-%h root) &
ALL   : ALL
```

アクセス制御

1. セキュリティーポリシーを決める
2. ユーザーの認証(パスワード管理)
3. ファイルの保護
4. Secureなホストへの アクセスの方法
5. アクセス制御(xinetd，TCP wrappers)
→ 6. 暗号化 (IPsec)
   (1) 暗号メール
   (2) トランスポート層での暗号化
   (3) IPsec
7. Firewall

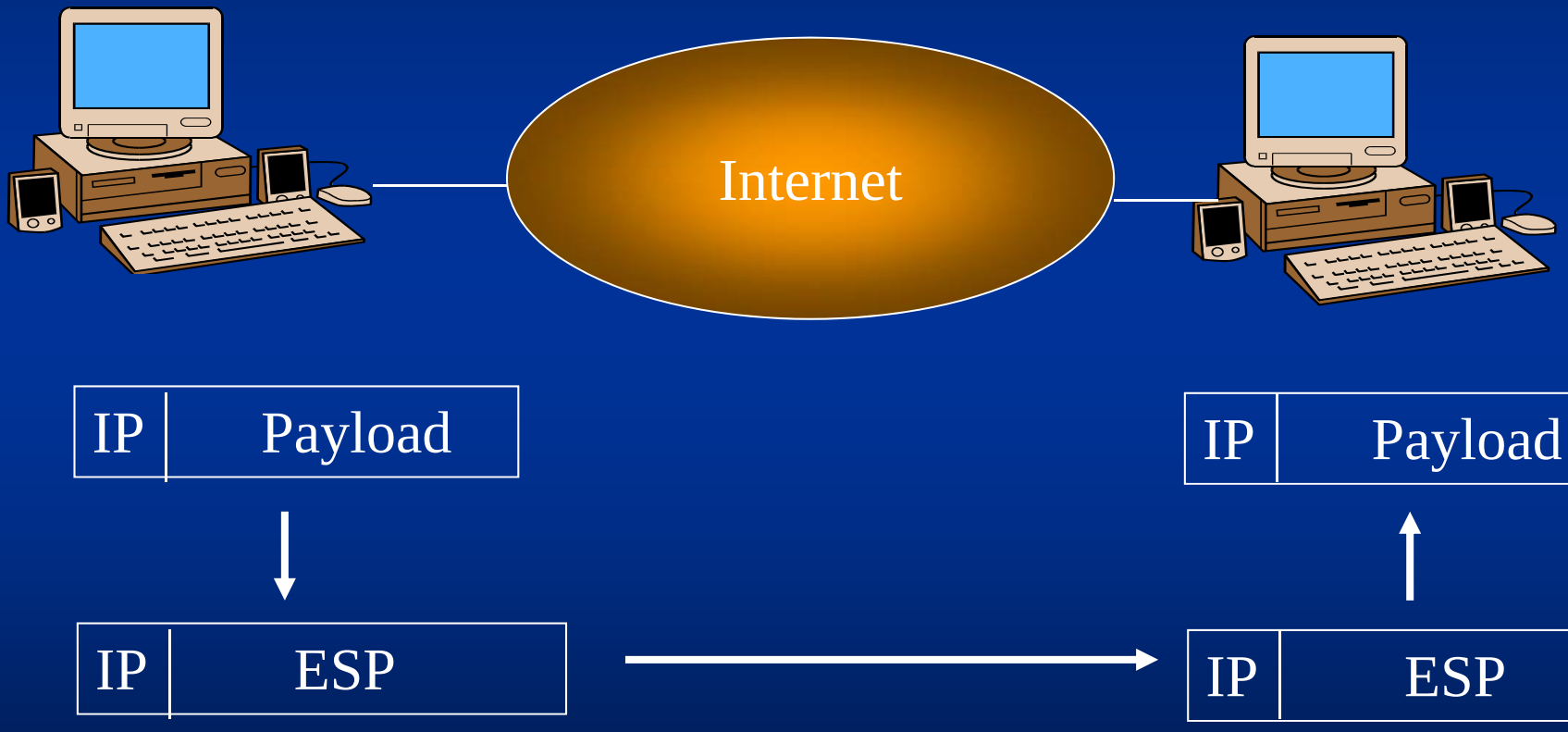
暗号化・IPsec

1.暗号メール
- PEM, MOSS
- S/MIME
- PGP (Pretty Good Privacy)

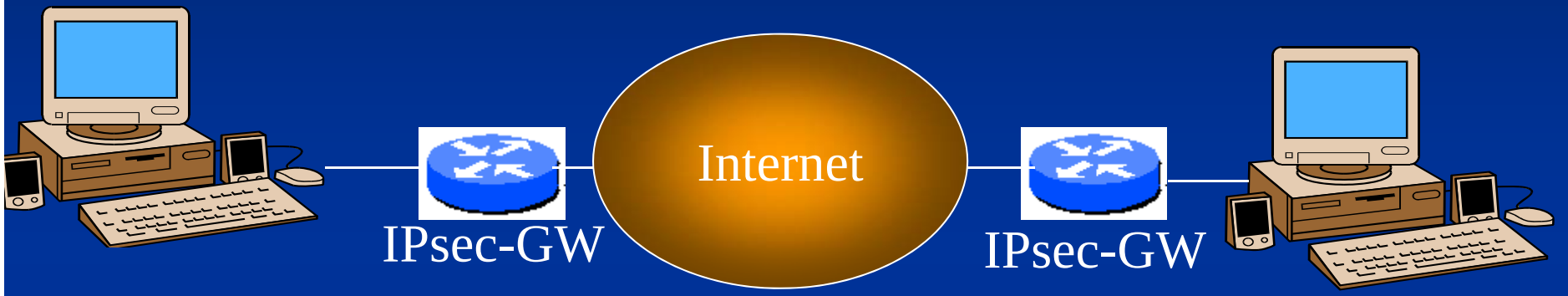
2. トランスポート層での暗号化
- SOCKS (http://www.socks.nec.com/)

3. IPsec
- 認証ヘッダ(AH: Authentication Header)
- 暗号ペイロード(Encapsulating Security Payload)
- 鍵交換プロトコル(Internet Key Exchange)

暗号化・IPsec
- トランスポートモード -



暗号化・IPsec
- トンネルポートモード -



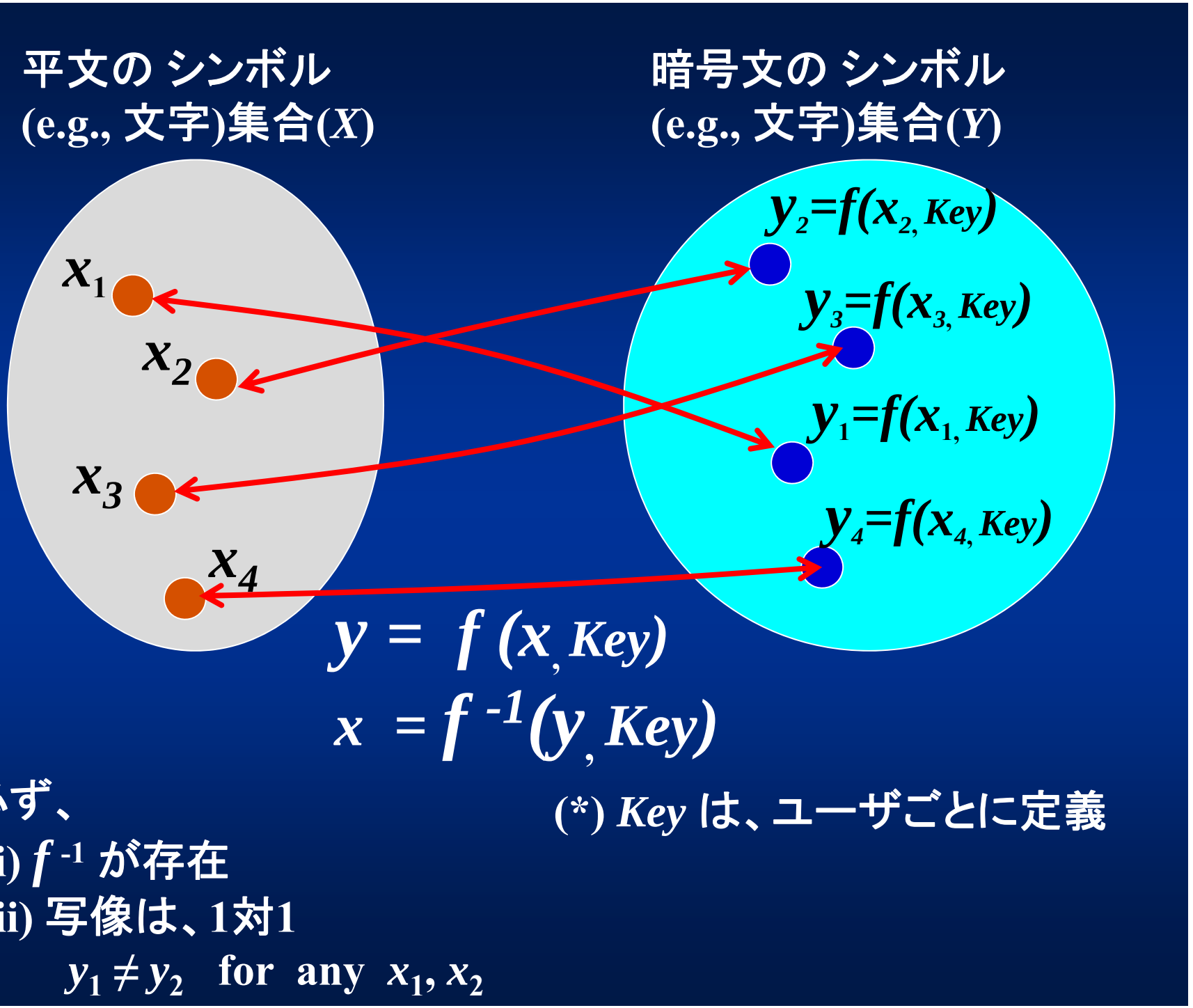
暗号化・認証アルゴリズム

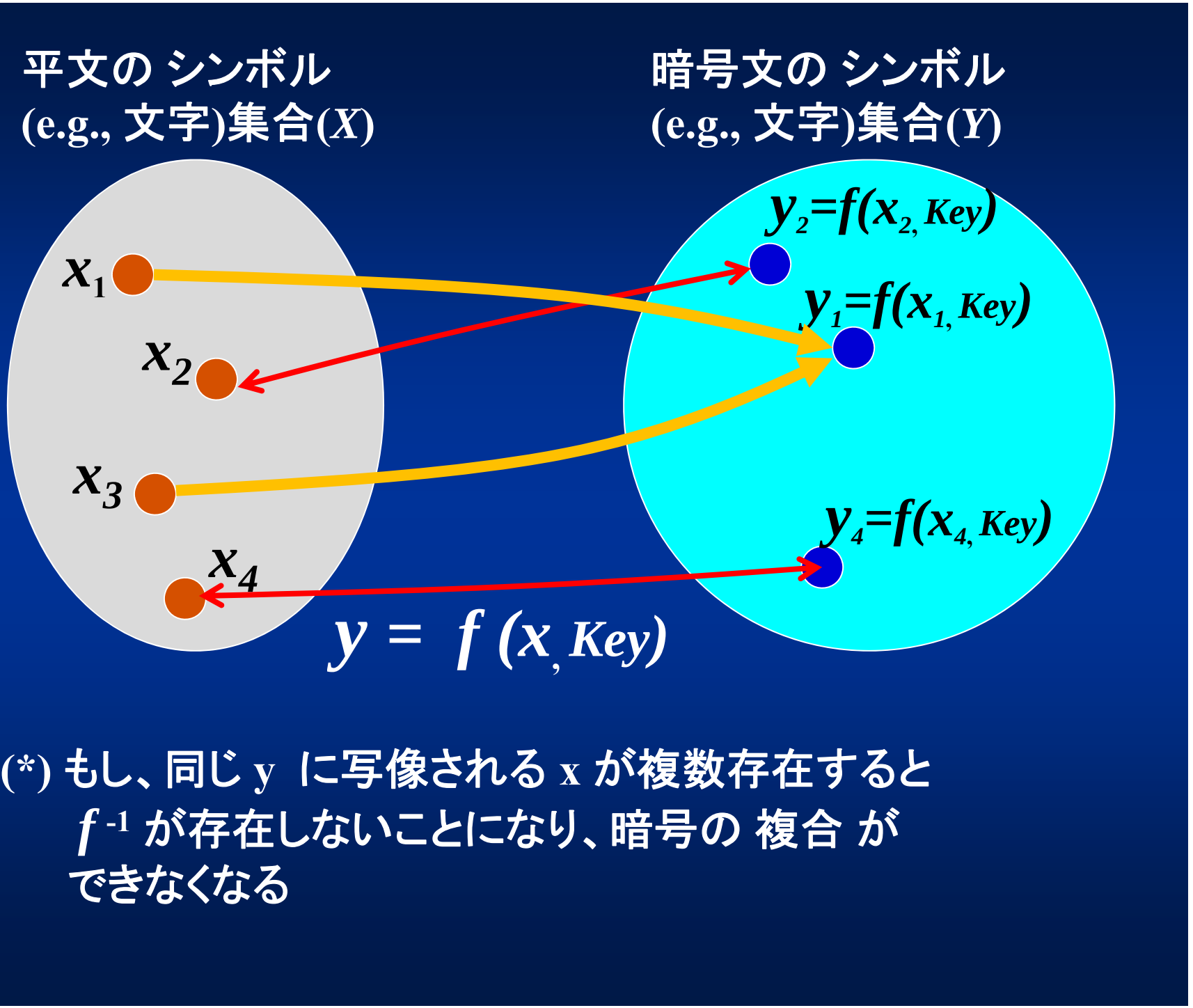
[1] 平文の認証 (一般には、N bits(入力) → m bits(出力), N>m)
[2] 平文の暗号化
(3) 暗号化
① DES(Data Encryption Standard) ; 秘密鍵方式
② RSA(Rivest, Shamir, Adleman) ; 公開鍵方式

暗号化は何をしているのか。

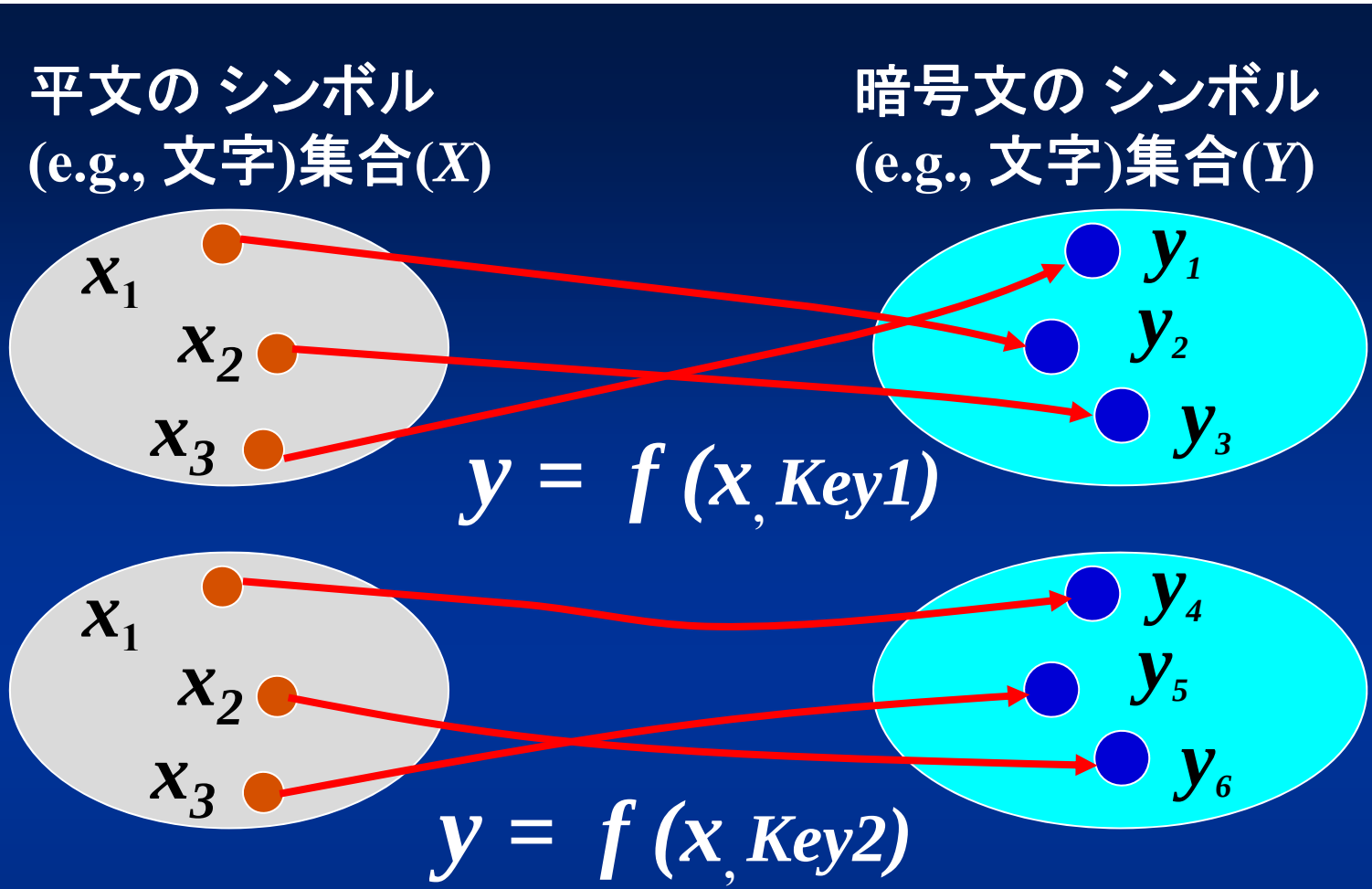
- {平文}文字の空間 と {暗号}文字の空間 との間の 写像{F(文字,パラメータ)} の計算を行っている。 {暗号}文を、たくさん 眺めると, 文字 と パラメータ が 見えてくる。
- この {たくさん} が、十分に大きければ よしとする。
- {暗号}文の空間 も、写像関数F もほぼ、無限に存在する。
  - 写像関数 F の 逆関数 F^{-1} が存在すること
  - 必ず、1対1 の写像となっていること。

(*) 認証では、この条件が少し甘くできる。


平文の シンボル
(e.g., 文字)集合(X)
暗号文の シンボル
(e.g., 文字)集合(Y)
x1
x2
x3
x4
y2=f(x2, Key)
y3=f(x3, Key)
y1=f(x1, Key)
y4=f(x4, Key)
y = f (x, Key)
x = f -1(y, Key)
(*) 必ず、
(i) f -1 が存在
(ii) 写像は、1対1
y1 ≠ y2 for any x1, x2
(*) Key は、ユーザごとに定義



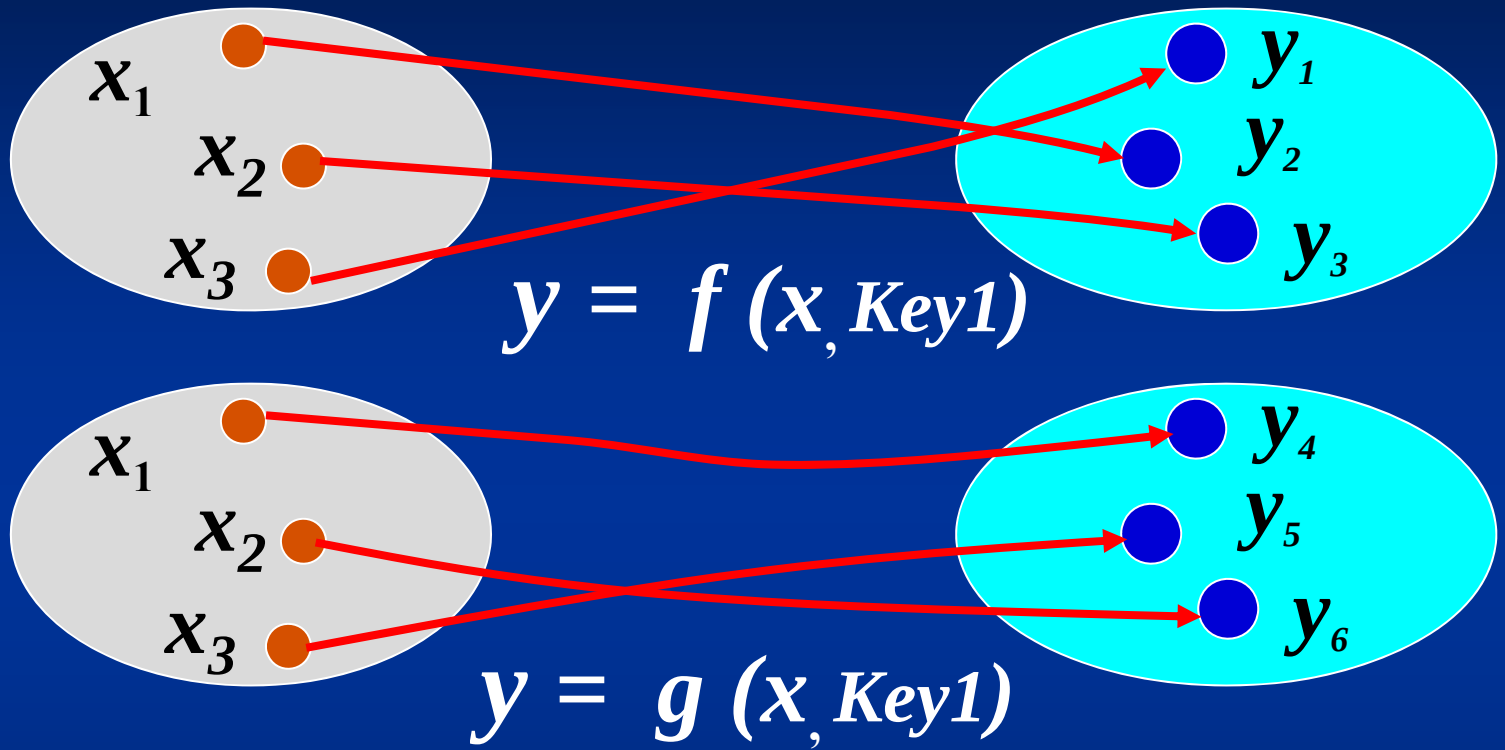
(*) もし、同じ y に写像される x が複数存在すると
f^{-1} が存在しないことになり、暗号の 複合 が
できなくなる



(*) 同じ写像関数(暗号化関数) f を用いていても、
関数 f のパラメータである Key が異なれば、
写像値(暗号文のシンボル)も異なる。

平文の シンボル
(e.g., 文字)集合(X)

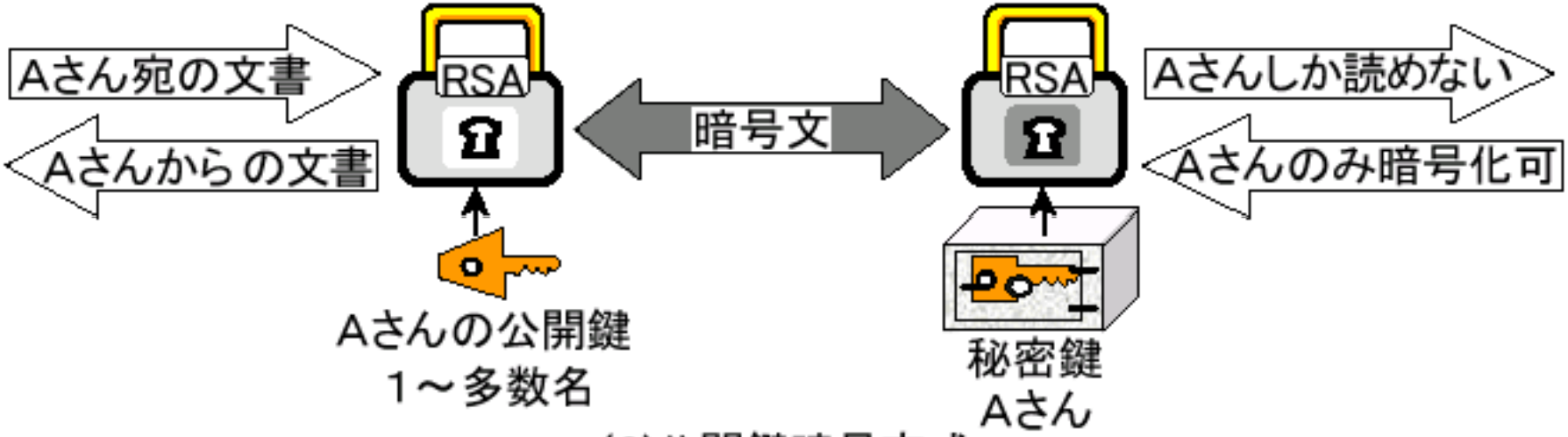
暗号文の シンボル
(e.g., 文字)集合(Y)



(*) 仮に、暗号化に必要な 鍵 (key) が同じでも、
異なる写像関数(暗号化関数) を用いることも
可能。

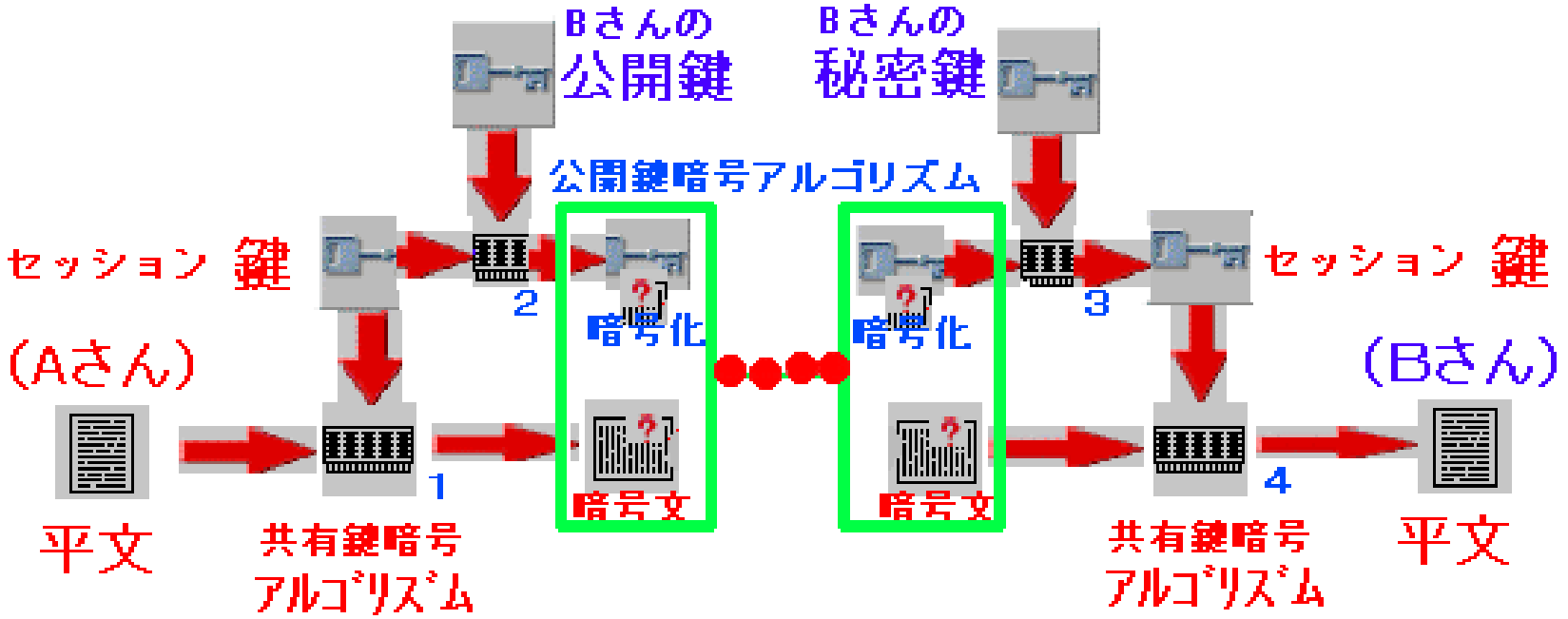
暗号方式の種類

(1)共通鍵暗号方式



(2)公開鍵暗号方式

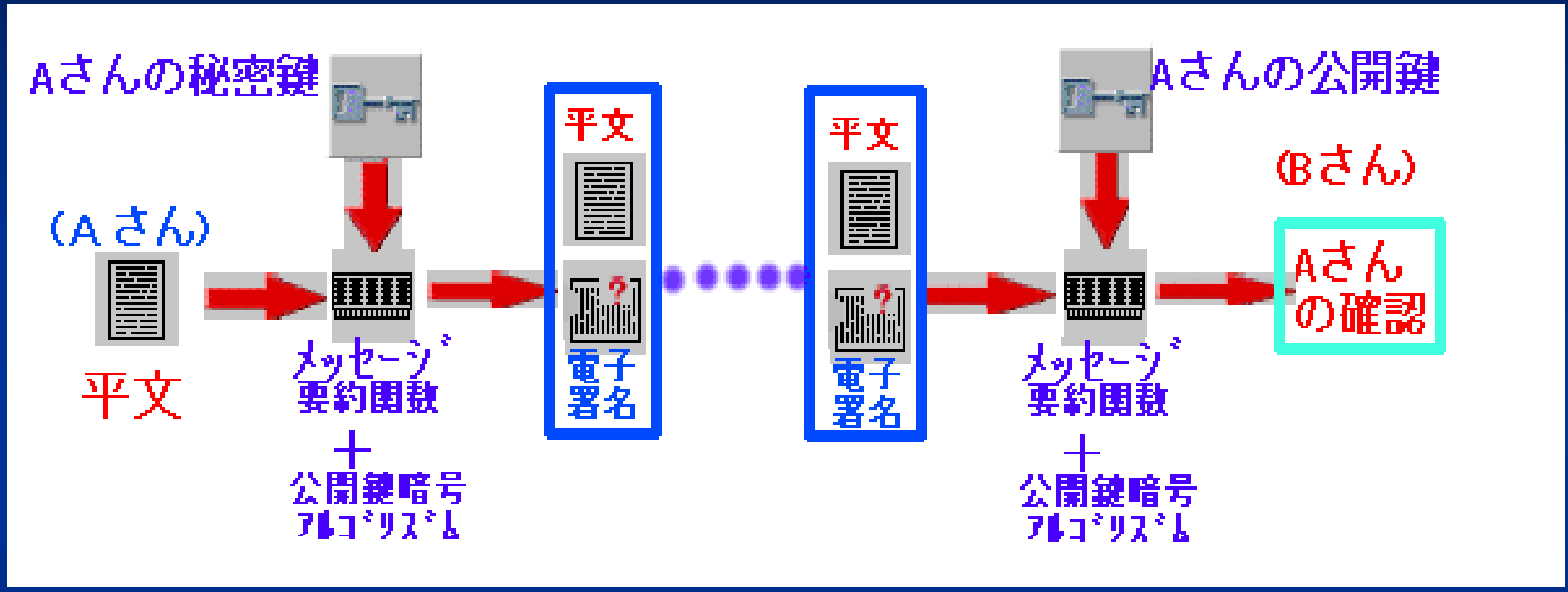
公開鍵暗号方式



例; ssh (Secured Shelll)

出典: http://www.psn.or.jp/TROUBLE/security.html

電子署名方式



- **MD5 (128 bits)**
- **SHA (160 bits)**

出典: http://www.psn.or.jp/TROUBLE/security.html

PGP署名メール

[署名したいメール]

７月３０日午前１０時に天満橋で会いましょう。

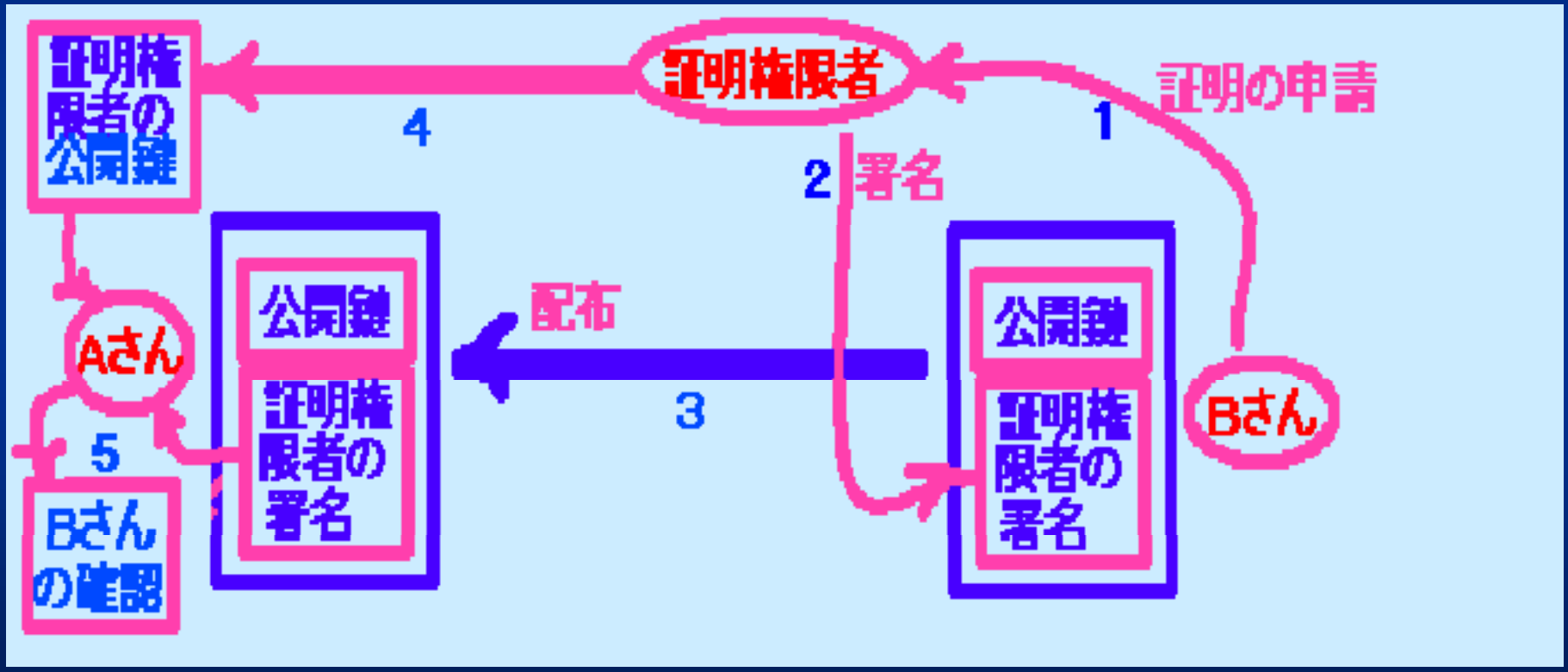
[署名されたメール]

```
-----BEGIN PGP SIGNED MESSAGE-----

７月３０日午前１０時に天満橋で会いましょう。

-----BEGIN PGP SIGNATURE-----
Version: 2.6.3ia
Charset: noconv

iQCVAwUBMey8R6UtC+xzFETZAQEnUAP+N30di02slY+rRYa2gBJ2u2ImWofjeyks
1AkvsN9errDk4N/VcFmc3d6F4heDkiy87u3XAVoulz2orb9xZ3qFveoEZp3QLLa6
Pkzs6/N1nmJZFZFlf1M8yUR5WZTbyaVHQmC1AuSZhJsM8+8S/+IbpXVPJJ68M4JE
cDYBT86eekM=
=UE6f
-----END PGP SIGNATURE-----
```

出典: http://www.psn.or.jp/TROUBLE/security.html

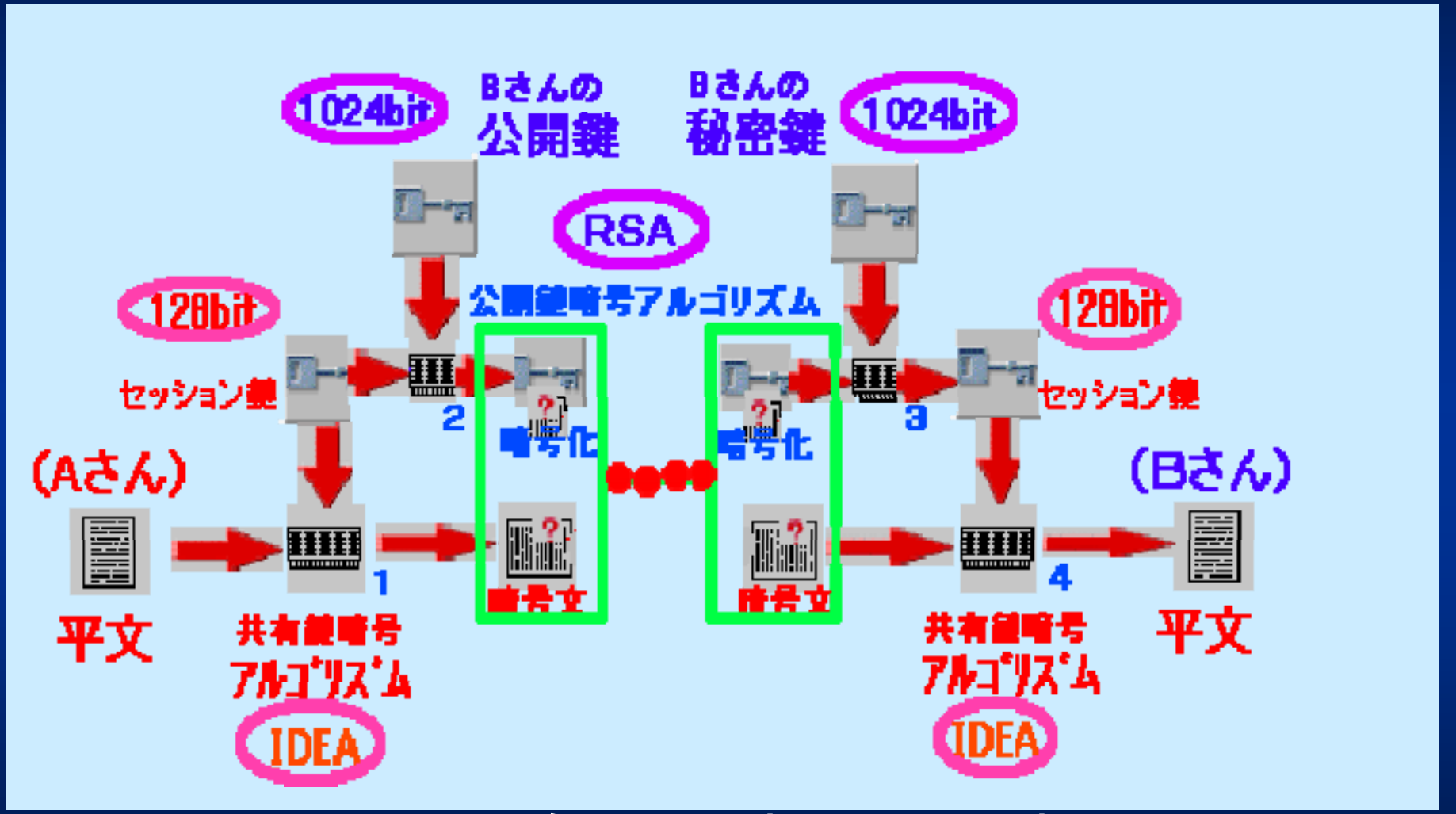
鍵の証明と配布

・証明を発行する機関・組織を利用
　例; Netscape Webサーバ 暗号化情報交換



出典: http://www.psn.or.jp/TROUBLE/security.html

PGP暗号化メール



公開鍵暗号化方式(暗号化方式；RSA+IDEA)

出典: http://www.psn.or.jp/TROUBLE/security.html

PGP暗号化メール

[暗号化したいメール]

７月３０日午前１０時に天満橋で会いましょう。

[暗号化されたメール]

```
-----BEGIN PGP MESSAGE-----
Version: 2.6.3ia
hIwDpS0L7HMURNkBA/4qk4BDXaiLag9tOS8srdd09IP4Pbocw8ERnYZKc8BJZHRq
bmePoSNRpv8QwRPttwB3pkUhPH9ET5BbGiyuw36hLvIet5z5ot3RS+XnfSz1Tyxw
xkXT+nNDCE6Gntb6JqBUym2/FRowwMNOc1bnKD6eIqZfekDUWBuHKSRduH6BfqYA
AAA3YBJcBDcrQtcIuA5R+bvivZ8gc8Fx3JCcUtW4yH+embVTTSUw+xTt0JSUoo93
 u5+LHGrrzBESSg==
=00WV
-----END PGP MESSAGE-----
```

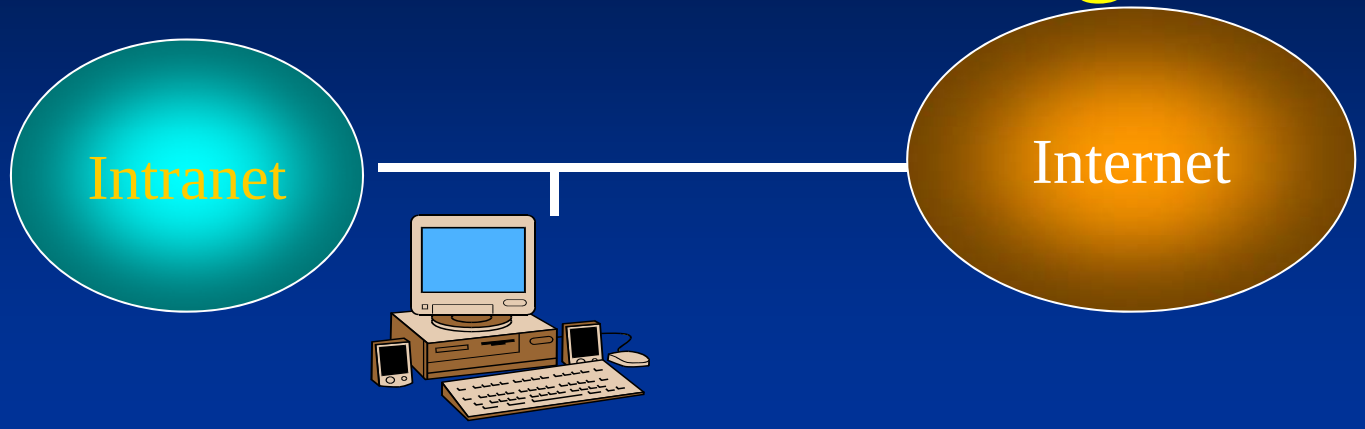
出典: http://www.psn.or.jp/TROUBLE/security.html

ネットワークセキュリティー

1. セキュリティーポリシーを決める
2. ユーザーの認証(パスワード管理)
3. ファイルの保護
4. Secureなホストへの アクセスの方法
   - Secure Shell； `ssh`
5. アクセス制御(xinetd，TCP wrappers)
6. 暗号化 (IPsec)

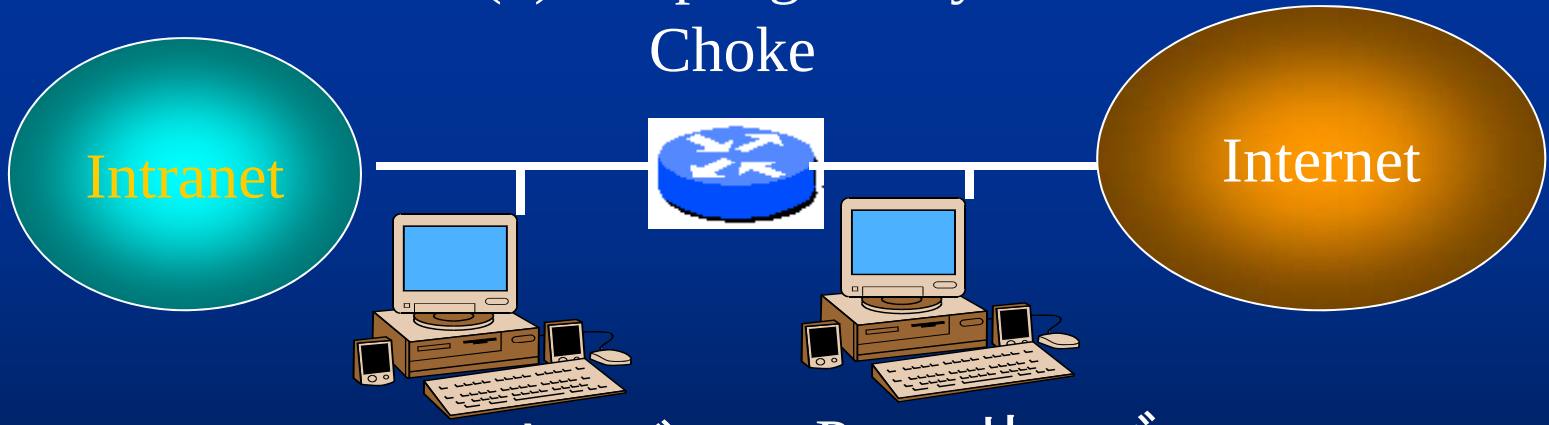
→ 7. Firewall

4 Levels of Firewall Configurations



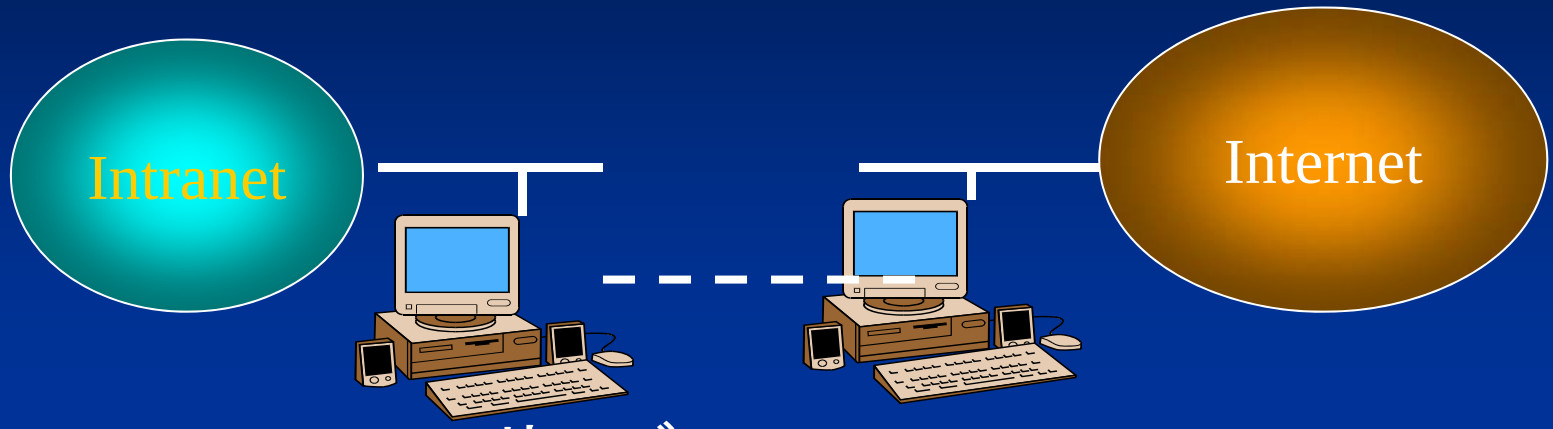
(1) Simple gateway

Choke



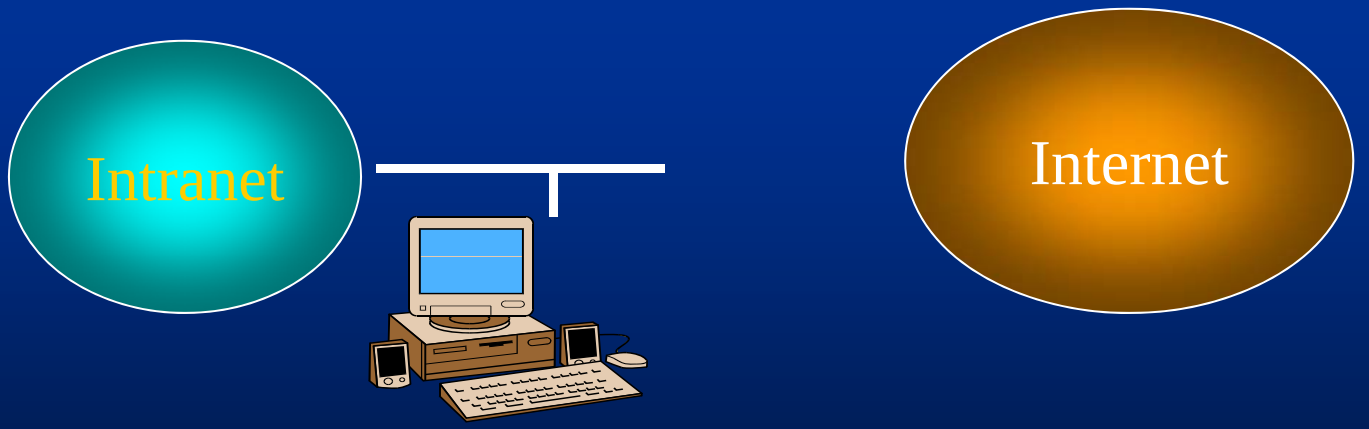
(2) Belt and Suspender

4 Levels of Firewall Configurations



Proxyサーバ

(3) TIPなどによる接続



(4) Disconnect

ファイアウォール

1. 経路情報の交換を停止
   → FW内の経路情報は外部に広告されない。
   (注) Source routingで進入可能

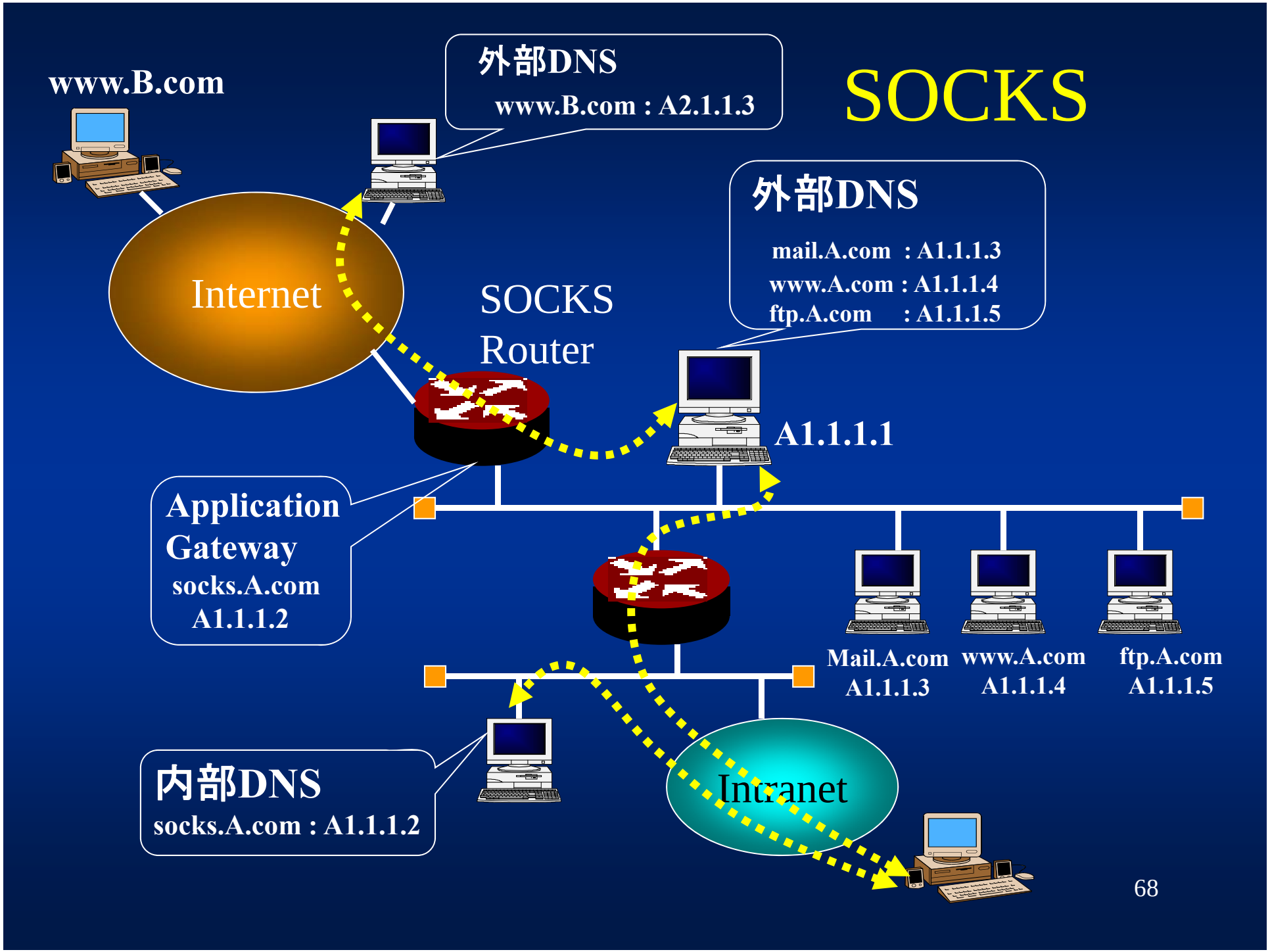
2. パケットフィルタリング
   socket{src_IP, src_port, dsrt_IP, dst_port}の情報でフィルタリングを行う。
   (注) - ftpなど相性がよくないアプリケーションが存在
   (a) 公開されるべきサービス
   WWW, anonymous-ftp, IRC
   (b) 公開されるべきではないサービス
   NIS, NFS, PRC, TFTP, SNMP
   (c) 内向きを禁止すべきサービス
   SMTP, NNTP, HTTP, FTP

ファイアウォール

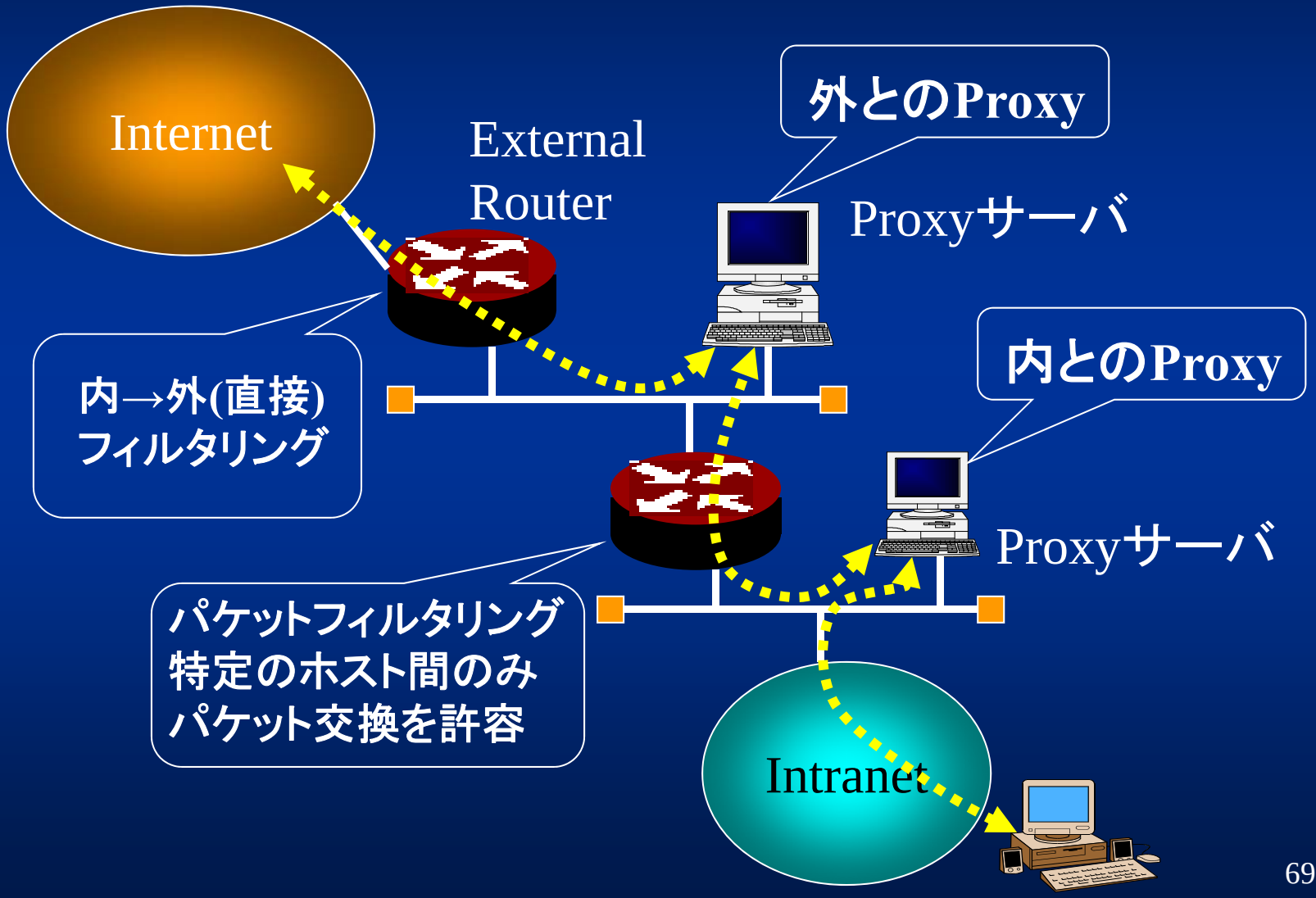
3. アプリケーション ゲートウェイ ; Proxyサーバ
→ 各アプリケーションでProxyサービスを提供する。

e.g., SOCKS

ftp://ftp.nec.com/pub/security/socks.cstc/socks.cstc.4.2.tar.gz


SOCKS
www.B.com
外部DNS
www.B.com : A2.1.1.3
Internet
SOCKS
Router
外部DNS
mail.A.com : A1.1.1.3
www.A.com : A1.1.1.4
ftp.A.com : A1.1.1.5
A1.1.1.1
Application
Gateway
socks.A.com
A1.1.1.2
Mail.A.com
A1.1.1.3
www.A.com
A1.1.1.4
ftp.A.com
A1.1.1.5
内部DNS
socks.A.com : A1.1.1.2
Intranet

Firewall System Configuration



電子メールシステムでの セキュリティー技術

- メール関連のセキュリティー的な問題・対策
  - パスワード保護; APOP
  - 不正使用; SMTPサーバ アクセス制御
  - メール改竄; 暗号メール、ヘッダチェック
  - SPAMメール
  - ウイルスメール

パスワード保護
ー APOP ー

- POP3の問題点
  - パスワード情報が定期的に平文で交換される
  - telnet(1キャラクタ/パケット)よりも危険
- 使い捨てパスワード(OTP)の利用
  - パスワード＝
    MD5(PROCESS_ID、TIME_STAMP、
    HOSTNAME、
    APOP_PASSWORD)

不正使用
SMTPサーバアクセス制御

- SMTPサーバのドメインからのメールのみをメールの転送を受け付ける(`/etc/sendmail.cf`)
  - 送信元IPアドレス(Source IP address)
    - 移動ホストからのアクセスをどう扱うか？ IPアドレスは、同じドメインのIPアドレスではない。
  - From内の送信元メールアドレスのドメイン名
    - なりすまし(Fromフィールドの改竄)への対処

メールの改竄への対策

- 暗号・署名メール
  - PGP、S/MIME、PEM、MOSS、KPS
  - トランスポート(End-to-End)モード 、トンネル(GW)モード
- メールヘッダのチェック
  - Received: 行
  - Message-Id: 行

SPAMメール・電子メール爆撃

- SPAMメール; 電子メール版ダイレクトメール
  - 語源:肉の缶詰。TVコメディー番組での「SPAM」の連呼から転じた。
  - SPAMメールを専門に引きうける業者の出現
  - メールサーバを土台にしたSPAMメールの送信
- 電子メール爆撃
  - 直接攻撃
  - SPAM利用攻撃
  - メーリングリスト悪用攻撃

SPAMメール・電子メール爆撃 対策

- 送信元IPアドレスのDNSによる存在確認
- 外から入ってきて外に出て行くメールの拒絶
- メール中継の場合には、ヘッダにトレース情報をきちんと残す
- MLへの投げ込みはメンバーのみに制限
- MLのメンバーリストの秘匿
- MLへの自動登録には注意

SPAMメール 対策

- 送信元のチェック
  - クライアントのIPアドレス、ドメインによるアクセス制御
  - SPAMブラックリストの利用 (/etc/sendmail.cf)

- HTML メールは何も考えずに捨てる

ウィルスメール

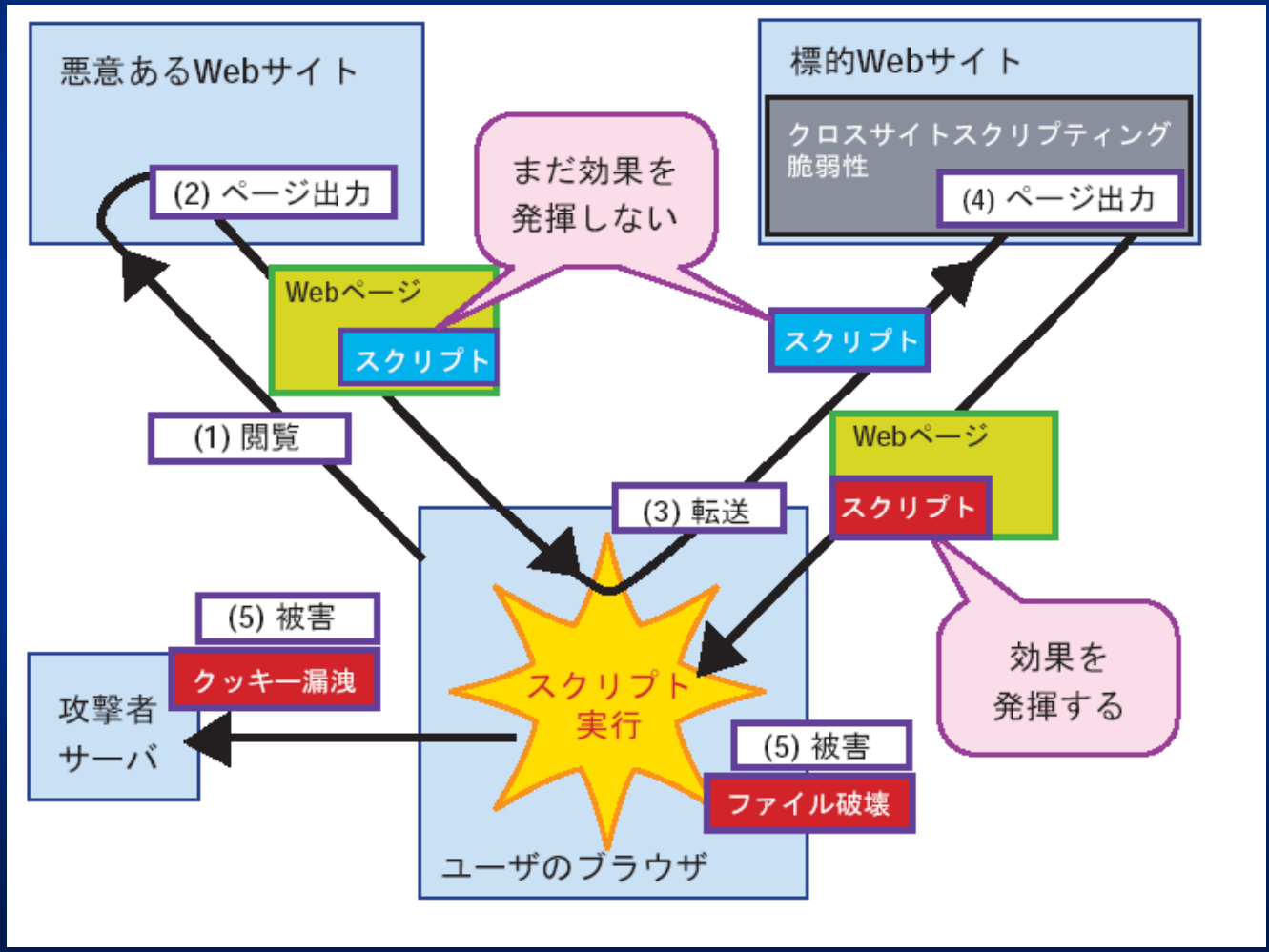
- 怪しい 添付ファイルは 迷わず捨てる
- 危ないメーラを使わない
- ウィルスチェックソフトを上手に使う。
  - でも最近は、ウィルスのSignatureパターンを使ったフィルタリングを行うルータも出てきた

プログラミング上の注意点

基本中の基本

- 入力値のチェック
  - クライアント側でのチェックは信用できない
    - 例えば、クロスサイトスクリプティング
    - 特に、SCRIPT 文は危ない。
    - SSLなどを用いた、Secureな通信
  - 必ずサーバ側でチェックすべし。
    - 誰も信用しない！
- URLストリング や hidden は、盲点かも
- 重要な情報は Web公開 ディレクトリの外に置く
- アクセス権への配慮
  - オブジェクト指向(e.g., JAVA)での 継承(Inherit)

クロスサイトスクリプティング



基本中の基本

- 入力値のチェック
  - クライアント側でのチェックは信用できない
    - 例えば、クロスサイトスクリプティング
    - 特に、SCRIPT 文は危ない。
    - SSLなどを用いた、Secureな通信
  - 必ずサーバ側でチェックすべし。
    - 誰も信用しない！
- URLストリング や hidden は、盲点かも
- 重要な情報は Web公開 ディレクトリの外に置く
- アクセス権への配慮
  - オブジェクト指向(e.g., JAVA)での 継承(Inherit)

基本中の基本(続)

- スレッド間での同期/ロック
- Perlなどでのファイルのオープン
- セッションタイムアウト ➔ 短く
- バッファオーバーラン
- MS Windows での Short Name問題。。。

図１　プロセスのメモリ空間

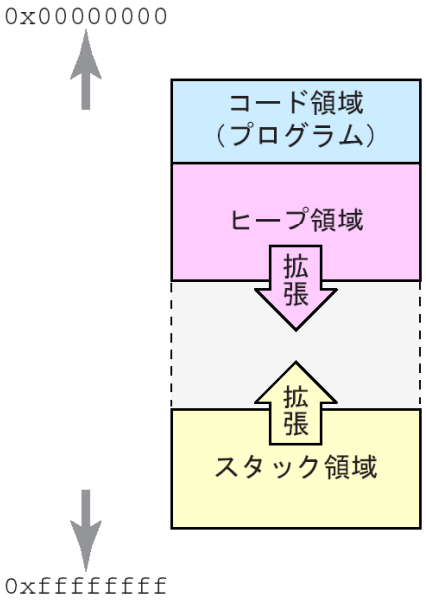


図２　スタック領域の拡張

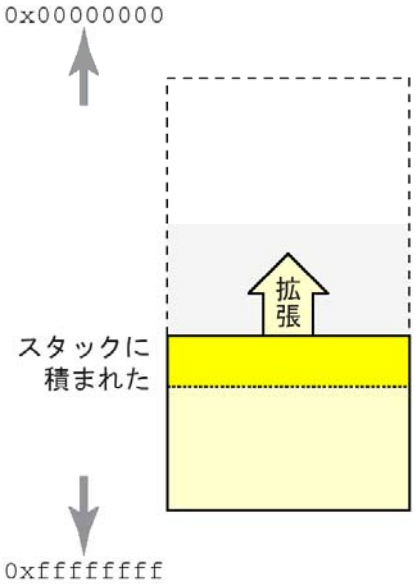
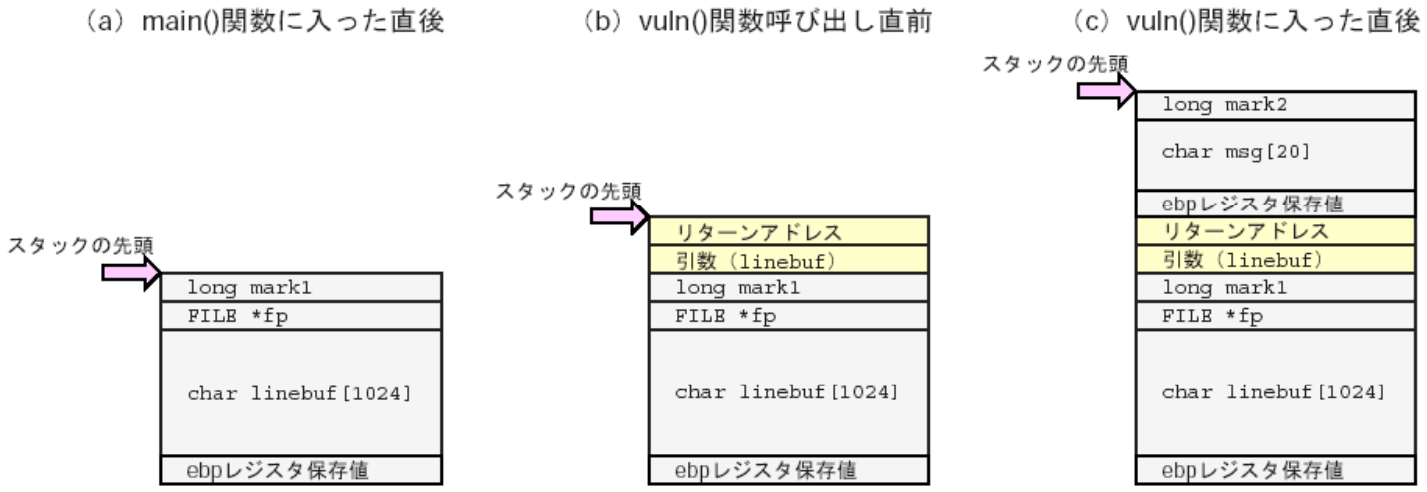


図３　スタックの変化



基本中の基本(続)

- スレッド間での同期/ロック
- Perlなどでのファイルのオープン
- セッションタイムアウト ➔ 短く
- バッファオーバーラン
- MS Windows での Short Name問題。。。