ネットワークセキュリティプロトコル:
入門編（仮訳）

2004年 3月

Radia Perlman
(radia.perlman@sun.com)

この入門編の目的

- ちょっと怖い領域へのいざない。
- 「知っておく必要があること」に対して「他人が知ったことで、あなたが信用できること」についての説明
- この分野（セキュリティ）を「手におえない分野」ではないものにする。
- Russ Housley氏からの言葉：
  「決して真似をしないように。」（WWE風）

問題点

- インターネットは、「捕食者が、存在しない（平和な）世界で発展した」。サービス妨害攻撃（DoS攻撃）は、「非論理的な行動」ではあっても、「無害である」と見なされていた。
- （しかし）今日の世界は、悪意に満ちており、ほんの少数の行動が膨大な被害を引き起こす。
- 中央管理無しに、相互に不信感を抱いている組織や人を相互に接続する必要がある。
- 社会は、「従来」のセキュリティの関心事のみならず、信頼性について、これ依存するようになってきている。

「セキュリティ」は、人によって違うことを意味する

- 意図した集団にのみにデータを開示する
- テロリストの逮捕のために、通信を監視する
- データが壊されないように保つ
- 海賊行為によってコンピュータを破壊する
- 悪いヤツを追いつめる
- 匿名で通信する

反セキュリティ

*インターネットは、反セキュリティではないが、**セキュアではない**と言えるであろう。反セキュリティは、精神的な状態である。インターネットのユーザは、反セキュリティである可能性があり、おそらく、そうであろう・・・。*

……Simson Garfinkel

侵入者：彼らに何ができるのか

- 盗聴する（ルータ／リンク／経路制御アルゴリズム／DNS）
- （IP ヘッダーを含む）任意のメッセージを送信
- 記録されたメッセージを再送
- 転送中のメッセージを変更
- 悪意のあるプログラムを書き、（騙して）人にそれを実行させる。

基本的な用語

- **認証** (Authentication)：**「誰？」**
- **認可** (Authorization)：**「あなたは、そんなことをする必要があるの？」**
- **サービス妨害** (DoS：Denial of Service)
- **インテグリティ保護**：**生成するため** (**および検証するため**) **に共有した秘密の知識を必要とするデータについてのチェックサム**

問題点に関心をもってもらうためのいくつかの例

- **複数のユーザでファイルを共有する場合**
  - **ファイルストアは、ユーザを認証しなければならない。**
  - **ファイルストアは、誰にファイルの読出し/更新を認可しているのか知らねばならない。**
  - **情報は、電線上での暴露・改ざんから保護されなければならない。**
  - **ユーザは、ファイルストアが本物であるかがわからねばならない。**(**秘密をあばいたり、不正な情報を読み出してしまわないために。**)

問題点に関心をもってもらうためのいくつかの例(続き)

- 電子メール
  - プライベートなメッセージを送る。
  - 誰がメッセージを送ったかが判る。(さらに変更されていないことが判る。)
  - 否認防止 –転送されたメールの受信者が元のメールの送信者を知ることができるようにすること
  - 匿名性

問題点に関心をもってもらうためのいくつかの例(続き)

- 電子商取引
  - クレジットカード番号を、知らせることなく決済を行う。
    - 盗聴者に対して
    - 電話の向こうの商人に対して
  - 匿名性を維持しつつ物を買う。
  - 商人は、商品が注文されたことを証明したい。

目標はときに背反する

- 「プライバシー」対「会社（もしくは政府）があなたが何をしているか知りたがること」
- 「データの喪失」対「公開（鍵の複製）」
- 「サービス不能」対「侵入予防」

暗号技術

- 暗号（アルゴリズム）
  - 共通鍵暗号
  - 公開鍵暗号
  - 暗号技術的ハッシュ
- ユーザの目的
  - 認証、インテグリティ保護、暗号化

共通鍵暗号

- 2つの操作（「暗号化」と「復号」）は、掛け算と割り算のように互いに逆。
- ひとつのパラメータ（「秘密鍵」）
- アルゴリズムの設計者でも、鍵がない限り復号できない。（（設計者が）悪意を持って裏口を作っていない限り。）
- 理想的には、ユーザのペアごとに違う鍵を使うこと。

共通鍵暗号(アリスとボブが秘密Sを共有する)

- encrypt=f(S, **平文**)=**暗号文**
- decrypt=f(S, **暗号文**)=**平文**
- **認証**: send f(S, **チャレンジ**)
- **インテグリティチェック**: f(S, **メッセージ**)=X
- **インテグリティチェックを検証**: f(S, X, **メッセージ**)

よくよく観察すると...

- セキュリティは、悪者の限られた計算資源に依存する。
  (「鍵」を見つけるために総当りチェックをすることができる。)
  - 計算機が暗号文を解読した結果を「平文」を認識できると仮定すれば。
- 優秀な暗号アルゴリズムは、正義の味方には(計算量は)比例で効き、悪者には指数的に効く。
- 64ビットの「鍵」でさえ、完全に探索するのは気が遠くなる。
- より高速なコンピュータが「正義の味方」に利するように働く！

公開鍵暗号

- 2つの鍵から成るペアの各ユーザと各鍵は(誰も鍵分割を発明しなかったかのように)、互いに逆元である。
  - 公開鍵"e"は世界中に公開して、
  - 私有鍵"d"は内密なものにする。
- 「そう、魔法なんです。しかし、なぜ"e"から"d"が判らないのでしょう？」
- 「そして、それが難しいのであれば、どうやって(e,d)を求めるのでしょうか？」

デジタル署名

- 「公開鍵暗号」の最も良い機能
- インテグリティチェック
  - 作成するとき f (プライベート鍵, data)
  - 検証するとき f (公開鍵, data, 署名)
- 検証者は、秘密を知る必要がない。
- 対「共通鍵暗号」
  - インテグリティチェックを同じ鍵で行うと、検証者がデータを捏造できる。

暗号技術的ハッシュ

- 公開鍵暗号処理が遅いために発明された。
- 私有鍵を使って、巨大なメッセージに署名するには時間がかかる。
- 暗号技術的ハッシュ
  - 固定長(例：160ビット)
  - 衝突が起きない! (同一のハッシュ値が、計算される事がない。少なくとも、決して見つけることはできないでしょう)
- それゆえ、実際のメッセージではなく、そのハッシュに署名する。
- もし、電子署名を行なうのなら暗号技術的なハッシュとあわせて署名すること！

普及している共通鍵暗号アルゴリズム

- DES（古い標準、56ビットの鍵、遅い）
- 3DES: 鍵長は固定。3倍遅い。
- RC4: 鍵長は可変。「ストリーム暗号」（鍵を使ってストリームを生成、データを排他的論理和をとる。）
- AES: DESの置き換えであり、後継になるであろう。

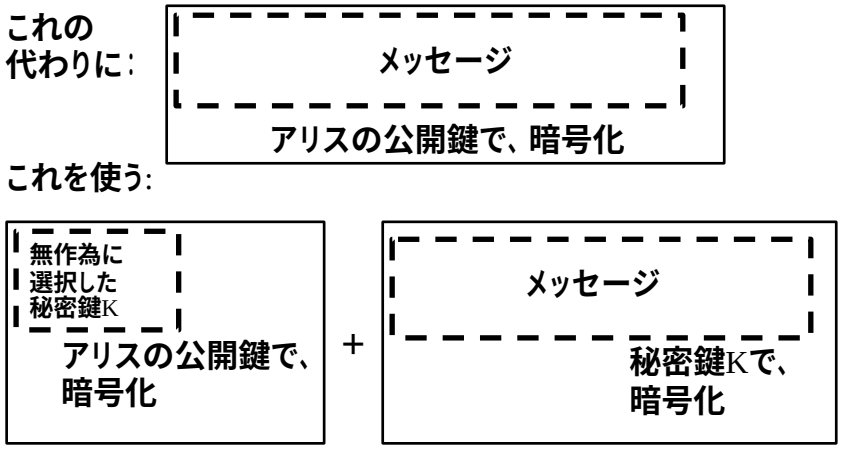
普及している公開鍵暗号アルゴリズム

- RSA：適切な機能：公開鍵での操作は、非常に早く、私有鍵での操作は、遅いかもしれない。特許はきれている。
- ECC（楕円曲線暗号）：より小さな鍵、RSAより速い。（ただし、公開鍵での操作は除く。）特許上の問題あり。

ハッシュ関連事項

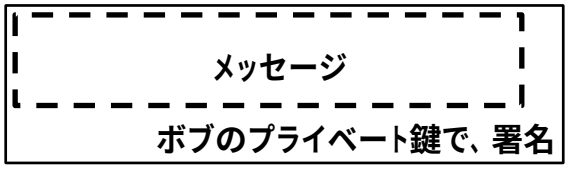
- 今日、もっとも使われているハッシュは、SHA-1（Secure Hash Algorithm）。
- 若干、古いもの（MD2, MD4, MD5）も使われている。
- 良く使われる共通鍵インテグリティチェック：秘密鍵とデータを一緒にハッシュする。
- IETFにおいて、そのためによく使われている他の標準：HMAC

ハイブリッド暗号



ハイブリッド署名

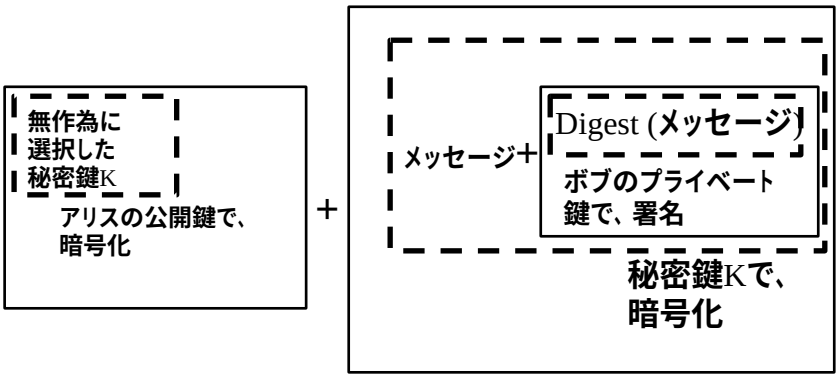
これの代わり:



これを使う:

メッセージ ＋

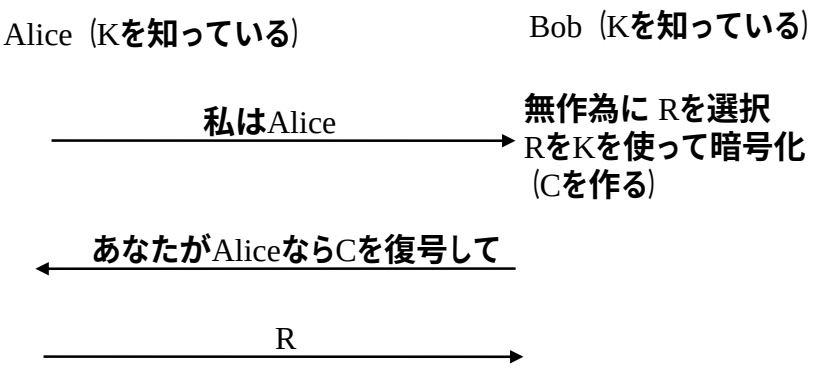
署名つき暗号化メッセージ



「決して真似をしないように!」

- 新しい暗号アルゴリズムを発明する理由はない。（暗号ギルドの会員を除く。）
- たとえ、あなたが、もっと良い（より速く、よりセキュアな）ものを発明できるとしても、誰もあなたを信じない。
- 良く知られ、良く評価された標準を使いましょう！

チャレンジ/レスポンス認証



非暗号技術的ネットワーク認証 (昔話)

- パスワードによるもの
  - 共有秘密を、あなたが知っている事を証明するために送る。
- (IP) アドレスによるもの
  - ネットワークアドレスが固定であり、アドレスの詐称が難しければ、送信元アドレスを使って認証できる。
  - UNIX .rhosts および /etc/hosts.equivファイル

人類

- 「人類には、高品質の暗号鍵をセキュアに記憶する能力が無く、また、暗号技術的な処理を行うとき、必要となるスピード、性能を出すこともできない。また、人類は、数が多く、管理コストが高くつき、管理が難しく環境を汚染し続けている。そういう状況にもかかわらず、人類は増加し働き続けていることは驚くべきことである。しかながら人類は世界中に広がっているので、人類の限界を考えたプロトコルを設計しなければならない。」
  - ネットワークセキュリティ：公衆に開かれた世界におけるプライベートな通信

人を認証する

- 知識による認証
- 所持品による認証
- 身体的特徴による認証

知識による認証

- ほとんどの場合、パスワードを意味する。
  - 盗聴の対象となる。
  - オンラインにおける推測の対象となる。
  - オフラインにおける推測の対象となる。

オンラインのパスワード推測

- オンラインでのみ推測可能なら、パスワードは「多少、推測し難い」程度で良い。
- 監査を行なうこともでき、対策することが可能である。
  - ATM: カードを没収する。
  - 軍事: 射殺する。
  - ネットワーク: アカウントを閉鎖(DoSになる)、もしくは試行毎に速度を落とす。

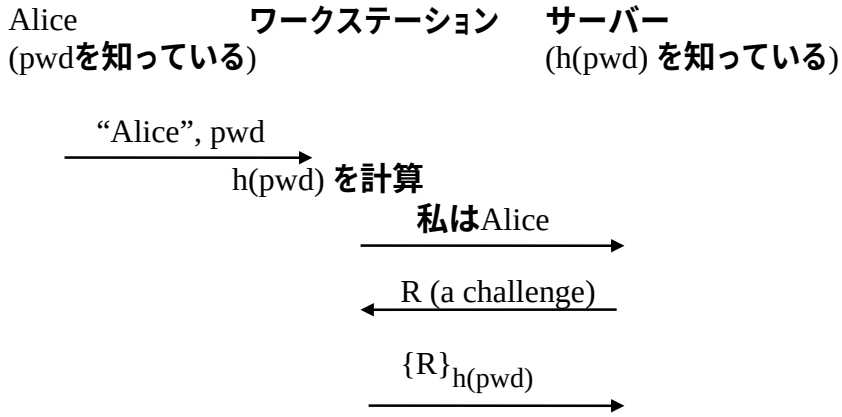
オフラインのパスワード推測

- パスワード推測が、ローカルの計算機資源によって検証される場合、パスワードは、無数の推測に対して耐えなければならない。

秘密鍵としてのパスワード

- パスワードは、秘密鍵に変換でき暗号技術を用いたデータの交換に利用できる。
- 盗聴者は、オフラインでの攻撃をすることにより十分な情報を得る事ができる。
- 大部分の人々は、このような攻撃に耐えうるパスワードを選択しない。

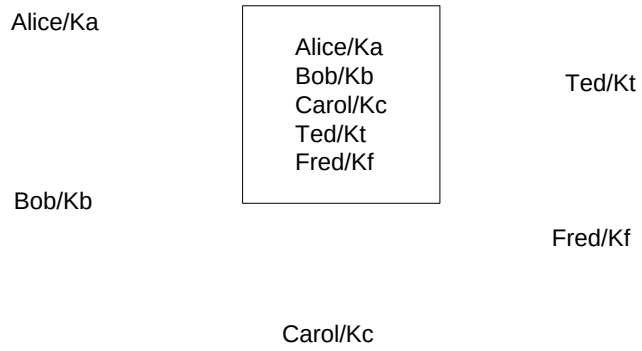
オフライン攻撃が可能である



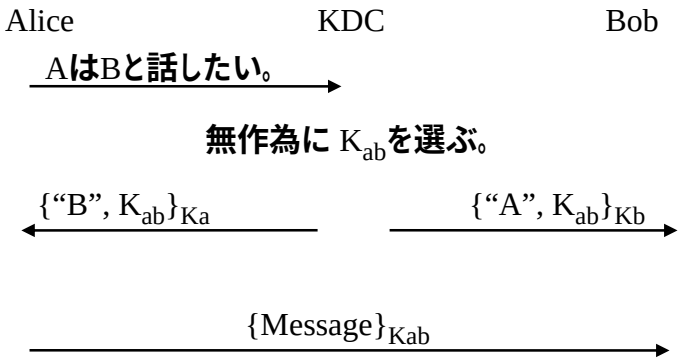
鍵配布(共通鍵暗号)

- n^2 keysを使うこともできる。
- KDC (Key Distribution Center) を使う。
  - すべての人々が、鍵をひとつもっている。
  - KDCは、それらすべてを知っている。
  - KDCは、会話をする必要があるあらゆる相手に対して鍵を割り付ける。
- これは Kerberosの基本である。

KDC



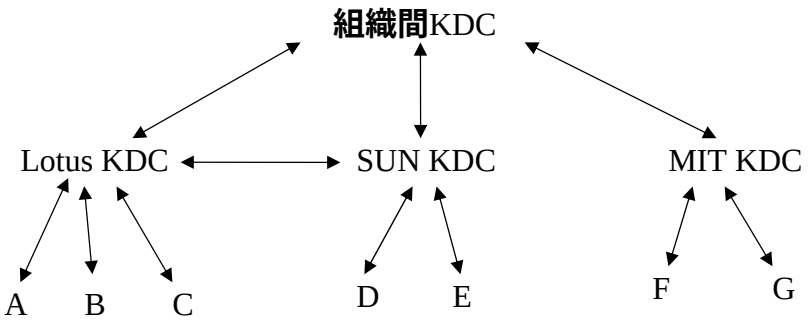
鍵配布(共通鍵暗号)



KDCレルム

- KDCは、数百のクライアントを扱うことができるが数百万は扱えない。
- 全世界の人を、信頼して秘密を喜んで共有しようという人はいない。
- KDCは、階層構造を扱うことができるので信頼は局所的にすることができる。

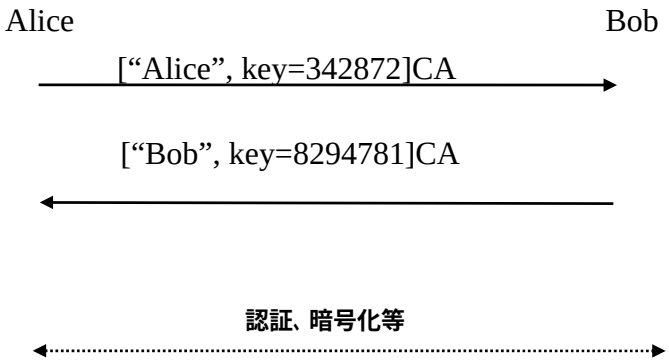
KDCレルム



鍵配布(公開鍵暗号)

- 認証局(Certification Authority、CA)は、「証明書」に署名する。
- 証明書 ＝「認証局である私は、489024729がRadiaの公開鍵である事を保証する」と署名されたメッセージ
- もし、みんなが証明書、プライベート鍵、CAの公開鍵を持っていれば認証できる。

鍵交換(公開鍵)



KDC対CA

- KDCのほうが、ややセキュリティが劣る。
  - 非常に、神経質に扱わねばならないデータベースをもつ(すべてのユーザの秘密がある)。
  - オンラインでなければならなく、ネットワークを通じてアクセスできなければならない。
    - 複雑なシステム、おそらく危険なバグがあり、標的として魅力的
  - 性能、有用性を確保するためにレプリカが必要
    - すべてのレプリカは物理的に安全でなければならない。

KDC対CA

- KDCの方が高価となる。
  - 大規模で、複雑、性能が重要、レプリカが必要
  - CAは賞賛されるべき仕組み
    - オフラインにできる(容易に物理的な安全性を確保できる)。
    - 数時間、止まっていても問題ない。
    - 性能は、それほど重要ではない。
- 性能
  - 公開鍵は遅いが、コネクションの開通時に第三者との通信しなくてよい。

KDC対CAの二律背反

- 遠隔地のCAへの接続が不要であるので、CAの方がレルム間ではうまく動く。
- 失効については似たり寄ったり。

失効処理

- 誰かが、クレジットカードを盗んだらどうする？
  - 期限切れにまかせる？
  - 利用できないクレジットカード番号の表を作って公開する？
    (CRLのメカニズムのように… Certificate Revocation List)
  - オンラインサーバを作る(OCSPのように… Online Certificate Status Protocol)。

階層化に関する戦略

- 独占
- 寡占
- 無政府
- ボトムアップ

独占

- 例外なく、信用できる組織をひとつだけ選ぶ。
- すべてのものに、その公開鍵を組み込む。
- 独占させ証明書を発行させる。
- すべての人に、証明書をそこから取らせる。
- 理解し易く、実現も容易。

独占モデルの間違い

- 独占による価格問題
- 遠隔の組織より証明書を取るのは、危険であったり、高価（もしくは両方）である。
- 鍵は、変えることができなくなる。
- 世界のセキュリティは、ある一つの組織の誠実さと適性に永遠に縛られる。

CAの寡占

- 80くらいの信用できるCAの公開鍵（「自己署名」の形式の証明書）を使う。
- 通常、追加・削除できる。
- 独占による価格上昇を排除できる。

寡占モデルの悪いところ

- 安全性が多少悪い
  - セキュリティは、利用する「すべて」の鍵にかかっている。
  - 純真なユーザは、偽の鍵が入っているプラットフォームや偽の鍵を追加される（悪意のあるソフトウェアのインストールで容易に行なえる）ことによりごまかされる。
  - トラストアンカのチェックは、現実的ではない。
- 独占ではないとはいえ、特定の組織では好意的に扱われている。なぜ、こういうものを信じられる？

無政府

- 誰もが、他人の証明書に署名（発行）できる。
- 設計され権限を委譲されている利用者は、意識して鍵を入れる。
- 問題点
  - 規模を拡張できない。（膨大な数の証明書、計算量の問題でパス構築が難しい）
  - パスが、信用できることを示すのが現実的ではない。
  - 利用者に、負荷がかかる。（実務と判断）

重要なアイディア

- CAについての信用は、技術的なものではない。:「このCAは信用されているか？」
- 代わりに、CAは、特定の証明書についてのみ信用される必要がある。
- 名前に基づくことが合理的あると見うけられる。（また、他の手法を使っているのを見たことない。）

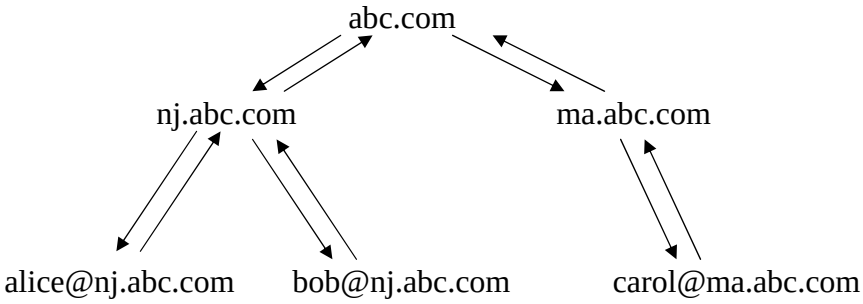
名前を基準としてトップダウンで行うポリシー

- 階層構造の名前空間を、想定する。
- 各CAは、自分をルートとした名前空間のみを信用する。
- 適切な連鎖を、容易に見つけられる。
- 何も考えていないユーザにとって、賢明なポリシーである。
- しかし、誰もがその鍵を用いて構成される必要があるので、トップは独占。

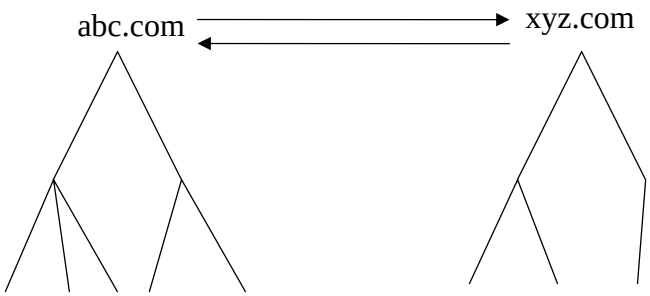
ボトムアップモデル

- 名前木内の各弧は、親証明書 (up) と子証明書 (down) をもつ。
- 名前空間において、すべてのノードはCAである。
- イントラネット間を接続する際、もしくはセキュリティを増加するために双方向の接続をする。
- 自分の公開鍵より始めて、up/cross/downする。

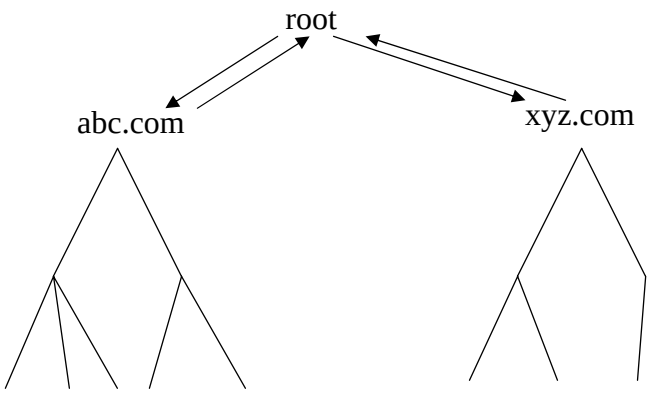
イントラネット



エクストラネット：双方向接続



エクストラネット: rootを追加する



ボトムアップの利点

- イントラネットについて、外の組織は必要ない。
- 自組織内のセキュリティを、自組織で管理できる。
- 一つの鍵が危殆化しても、大規模な再構成の必要はない。
- 構成が容易：自分自身の公開鍵を使える。

どのレイヤで？

- レイヤ2
  - 「ホップ by ホップ」で防護する。
  - 盗聴者に、IPヘッダを隠すことは可能である。（「トラフィック解析」から防護する。）
- レイヤ3/4（詳細については、次のスライド）
  - 「エンド to エンド」で守る。リアルタイム。
- 上位レイヤ（例：PGP, S/MIME）
  - メッセージを保護。蓄積・転送（リアルタイムでない）。

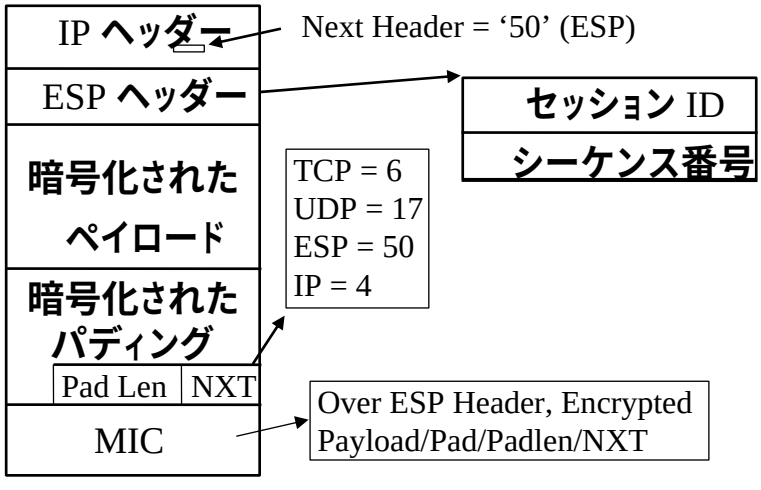
「鍵交換」

- 双方向（mutual）の認証／セッション鍵生成。（「セキュリティ協定」を結ぶ。）
- セッション全体を暗号技術で守るのが吉。（最初の認証だけではなく。）
- セッション毎に鍵を変えるのが吉。
- 例：
  - SSL/TLS もしくはSSH（Secure Shell）（「レイヤ4」）
  - IPsec（「レイヤ3」）

「レイヤ3」対「レイヤ4」

- **レイヤ3の思想**: **アプリケーションやAPIを変更する必要はない、OSのみの変更。**
- **レイヤ4の思想**: **OSは変更しない。アプリケーションのみを変更する。**
  レイヤ4（TCP/UDP）の上で動く。

ESP（Encapsulating Security Payload）



「レイヤ3」対「レイヤ4」

- 技術的には、「レイヤ3」の方が優位
  - 悪意あるパケットの問題
    - TCPは暗号には関与しない、そのため攻撃者は無効なパケットを挿入でき、TCPは回復できない。
  - 外付けハードウェアでの処理を行なうのが容易(各々のパケットは、独立して暗号化されているので)。
- 「レイヤ4」の方が、配備しやすい。
- さらに、APIを変更しない限り、「レイヤ3」は上位層に認証済みの識別情報を渡すことができない。

IETFのセキュリティエリアで行われていること

- Kerberos
- PKIX (証明書の形式) (次のスライドを参照)
- S/MIME, PGP
- IPsec, SSL/TLS, SSH (Secure Shell)
- SASL (認証プロトコルのネゴシエーション・シンタックス)
- DNSSEC (DNSにおける公開鍵、署名データの利用)
- sacred (クレデンシャルをダウンロードする)

PKIX

- X.509に基づいている（！）
- 2つの問題：
  - ASN.1エンコード：大きなコードになり、証明書も大きくなる。
  - インターネットのアプリケーションで、使われている名前ではない！そのため…
    - 名前を無視する、か
    - 別名にDNS名をいれる、か
    - CN=DNS名、あるいは
    - DC= とする。

PKI（続き）

- SSL/TLS、IPsecおよびS/MIMEで、PKIXは使われている。（まぁ、成功しているものも失敗しているものもあるけど。）。
- 名前の問題は、問題をはらんで入るが…
  - 同じ組織内にJohn Smithという人が複数名いたとしたら、どうなるのか？
  - 暗号以外に深い論点の一例として、証明可能なセキュアなハンドシェイクなどがある。

それでも、すべてのプロトコルは、「セキュリティに関する考慮事項」の章が必要

- 何について、考えなければならないのか?
- 「IPsecを使う」だけでは不十分。
- しばしば（例：VRRP（Virtual Router Redundancy Protocol）のように）、ひとつのプロトコルだけを守るのは無駄な努力。
  - 玄関が大きく開いているのに、窓に高価な鍵をつけても仕方がない。
- プロトコルを守るのではなく、ユーザを守るのだ。

例

- ルーティング情報についてインテグリティチェックを導入する
  - 部外者が「ルーティング情報を勝手に挿入するのを防ぐこと」自体は良いこと。
  - しかし、ARPに対して部外者が、応答することは防げないし、DNSの情報を壊すことも防げない。
  - 「ビザンチンの失敗（信用されているものが害悪となること）」を防ぐことはできない。

例

- SNMP（Simple Network Management Protocol）
- **直接的な「エンド to エンド」のセキュリティが必要**
- **ネットワークが脆い環境・おかしくなっていても、動作しなければならない。**
  - DNSがない。
  - **証明書を配布する LDAPデータベース（リポジトリ）がダウンしている。**

例

- **暗号化されていないもの**
  - **資源を使い尽くすもの**
    - DHCP可能な限りすべてのアドレスを要求できる。
    - リンクの帯域幅を使い尽くす。
  - **アクティブコンテンツ**
    - **アクティブコンテンツに関しては、非常に多くの隠れている例がある！**
- **暗号化は、インテグリティを保障しない！**

「セキュリティに関する考慮事項」の章に盛り込むべき内容

- 解決するセキュリティ論点
- 解決しないセキュリティ論点
- セキュリティに影響を与える可能性がある実装もしくは配備上の問題点

(私が考える) 良い「セキュリティに関する考慮事項」の章の例

- Kerberos NAS (Network Auth Service) のインターネットドラフト
- 抜粋
  - 認証問題を解決する。
  - しかし、認可／DoS／PFSの問題には対応していない。
  - 鍵をオンラインデータベースにする必要がある。そのため、NASは物理的にセキュアにしなければならない。
  - 辞書攻撃の恐れがある。(「良い」パスワード選択しないといけない。)
  - 適度に同期がとれている時間が必要。
  - チケットに個人情報が入る可能性がある。
  - NASリプレイ攻撃を防ぐために認証子を記憶する必要がある。

結び

- *つい2,3年前まではインターネットにつなぐとセキュリティを考慮することなくすぐに数億のノード接続できた。1990年代のインターネットは、1960年代のsexと良く似ている。これらは続いているうちは、偉大であるが本質的に不健康であり、最終的には悲劇に終わるように運命付けられている。私は今、この場におり、大変うれしい。*

—Charlie Kaufman