情報ネットワークⅡ（情213）

【第4回】

暗号技術（2）

（教科書：第2章）

担当教員：長田智和

E-Mail: nagayan@ie.u-ryukyu.ac.jp

URL: http://n-lab.info/

（講義日：2017年10月26日）

第2章：暗号技術（2）

2.3 公開鍵暗号化技術

- 2.3.1 RSAの仕組み
  - 公開鍵=鍵の生成者が広く多くの人に配布し、それを用いて暗号化するための暗号化鍵
  - 秘密鍵=ペアとなる公開鍵で暗号化された暗号文を復号できる唯一の復号鍵 *秘密に管理する必要あり

2.3 公開鍵暗号化技術

- 2.3.1 RSAの仕組み
  - RSAの仕組み
    - 巨大な数の素因数分解が困難である性質を利用した暗号化アルゴリズム
    - 素因数分解問題の困難さとRSAの安全性は等しい
    - RSAはデジタル署名にも使える公開鍵暗号方式（後述）
    - 512,1024,2048,4096ビットの鍵長（2048ビット以上の鍵長の利用が推奨）
    - RSAは暗号化・復号に時間がかかる（DES比100倍程度）

2.3 公開鍵暗号化技術

2.3.1 RSAの仕組み

RSAの鍵生成手順

- 大きな素数 p と q をランダムに選ぶ。
- n = pq を求める（ n の長さが鍵長となる）
- (p-1)(q-1)と互いに素な正の整数 e を求める。
- ed ≡ 1(mod (p-1)(q-1))となる正の整数 d を求める。
- 公開鍵（暗号化鍵）として、e と n を公開する。
- d は秘密鍵として安全に管理する。p と q は破棄する。 *重要
- 以上より、暗号化、復号は下記の通りとなる。

暗号化: $C = M^e \bmod n$ （C=暗号文、M=平文）

復号: $D = C^d \bmod n$ （D=復号文（＝平文（M））

2.3 公開鍵暗号化技術

2.3.1 RSAの仕組み

RSAが安全であるとされるポイント

暗号化（平文を e 乗する）は容易であるのに対し、復号演算は秘密鍵 d（$d \equiv e^{-1} \pmod{(p-1)(q-1)}$）を知らなければ、巨大な数の素因数分解をしなければならず、それは数学的に困難な問題とされている。このことが安全性の根拠とされている。

2.3 公開鍵暗号化技術

- 2.3.1 RSAの仕組み
  - RSAの演算の例
    - 鍵の生成
      - 2つの素数を p=5 及び q=11 とする。
      - n = pq = 55　→ 55 を法とする世界を考える。
      - (p-1)(q-1)と互いに素な正の整数を e=3 とする。

        ＊互いに素＝最大公約数が 1 となる数
      - ed ≡ 1(mod (p-1)(q-1))となる正の整数をd=27 とする。
      - 以上より、e=3, n=55 を公開鍵、d=27 を秘密鍵とする。

2.3 公開鍵暗号化技術

- 2.3.1 RSAの仕組み
  - RSAの演算の例
    - 暗号化
      - 平文を 7 とする。
      - $C = 7^3 \bmod 55 = 13$
    - 復号
      - $D = 13^{27} \bmod 55 = 7$

2.3 公開鍵暗号化技術

2.3.1 RSAの仕組み

RSAの仕組みのまとめ

- ランダムに選んだ大きな2つの素数を p, q（p≠q）とすると、n = pq を法とする世界を考える。
- 平文Mを（p-1）（q-1）と互いに素である整数e（公開鍵）で冪乗した値を暗号文M^e とする。
- 平文Mは{（p-1）と（q-1）の最小公倍数+1}回冪乗すると元の値に戻り、さらに{（p-1）と（q-1）の最小公倍数}回冪乗するたびに元の値に戻る性質がある。

  すなわち、整数 d を秘密鍵とすると、

  $(M^e)^d = M^{\{(p-1)(q-1)+1\}} \rightarrow ed = (p-1)(q-1)+1$

  $\therefore ed \equiv 1 \pmod{(p-1)(q-1)}$

  が成り立つ。

2.3 公開鍵暗号化技術

- 2.3.2 RSA以外の公開鍵暗号化方式
  - エルガマル暗号
    - 離散対数問題の困難さを安全性の根拠とした暗号化方式
  - 楕円曲線暗号
    - 楕円曲線上の離散対数問題の困難さを安全性の根拠とした暗号化方式
  - DSA（Digital Signature Algorithm）
    - デジタル署名のみで利用可能
    - 1994年にFIPS Publication 186として策定
    - 離散対数問題の困難さを安全性の根拠とした暗号化方式
    - 同じ鍵長の場合、RSAよりも処理が遅い

2.3 公開鍵暗号化技術

- 2.3.2 RSA以外の公開鍵暗号化方式
  - ECDSA（Elliptic Curve Digital Signature Algorithm）
    - 楕円曲線上の離散対数問題の困難さを安全性の根拠とした暗号化方式
  - 楕円曲線暗号はRSAと比較して、鍵長が小さくても安全性の確保が可能。
    - 例：

      楕円曲線暗号は鍵長が224ビットで、RSAの鍵長が2048ビットの暗号強度と同等の暗号強度を持つとされる。

2.3 公開鍵暗号化技術

- 2.3.3 ハイブリッド暗号
  - 共通鍵暗号化方式と公開鍵暗号化方式を組み合わせて利用する暗号方式
    - RSAは鍵長が長いほど処理速度が遅くなる欠点がある。
    - 平文はAESなどの共通鍵暗号化方式で暗号化し、その暗号化鍵をRSAで暗号化するのが一般的。
    - ハイブリッド暗号方式で一時的に作成される暗号化鍵をセッション鍵と呼ぶ。
- （教科書p.52の図2.28を参照）

2.4 鍵共有アルゴリズム

- 2.4.1 Diffie-Hellman（DH）鍵共有アルゴリズム
  - インターネットなどの通信路で暗号化鍵（共通鍵暗号化方式の暗号化鍵）を交換するための技術

2.4 鍵共有アルゴリズム

- 2.4.1 Diffie-Hellman（DH）鍵共有アルゴリズム
  - アルゴリズムの説明
    - 鍵を交換する主体をA, B とする。
    - A, B はあらかじめ素数 p と原始根 g を交換しておく。
      （p, g は秘密にする必要はない）
    - 鍵の共有時は、A は乱数 x、B は乱数 y を生成し、秘密に管理する。
    - A は $n = g^x \bmod p$、Bは $m = g^y \bmod p$ を計算する。A, B は n, m を互いに交換する。（n, m は秘密にする必要はない）
    - A, B は、$K = \{m|n\}^{\{x|y\}} \bmod p = (g^{xy} \bmod p)$ を共有する暗号化鍵として使用する。
- （教科書p.53の図2.29を参照）

2.4 鍵共有アルゴリズム

- 原始根とは？
  - 原始根とは、素数 p を法とした場合、冪乗した値によって、素数 p より小さい値（p で割ったときの余り）がすべて得られる値のこと。
  - （例）5 を法として 3 の冪乗を計算してみる。

$3^0 \equiv 1 \pmod 5$

$3^1 \equiv 3 \pmod 5$

$3^2 \equiv 4 \pmod 5$

$3^3 \equiv 2 \pmod 5$

$3^4 \equiv 1 \pmod 5$

$3^5 \equiv 3 \pmod 5$　∴ 3 は法 5 の原始根である。

2.4 鍵共有アルゴリズム

- 2.4.1 Diffie-Hellman（DH）鍵共有アルゴリズム
  - DH鍵共有アルゴリズムに対するMITM攻撃
    - 主体A, B の間に攻撃者I が入り込み、I は自身が生成した n', m' をA, B に成りすましてそれぞれに転送する。
    - 問題点: 素朴なDH鍵共有アルゴリズムでは、A, B が受け取ったメッセージの真正性を確認できない。このため交換するメッセージにデジタル署名を施すなどの防御策が必要。
- （教科書p.54の図2.30を参照）

2.5 ハッシュ関数とデジタル署名

- 2.5.1 ハッシュ関数
  - 任意の長さのデータを固定長（128-512ビット程度）に圧縮する関数のこと。以下の性質を持つ。
    - 一方向性: 出力値から入力値を発見することが困難であること。すなわち、

      h = hash（m）

      を満たすm を求めることが困難であること。
    - 第2原像計算困難性: ある入力値と同じハッシュ値となるような別の入力値を求めるのが困難であること。

      すなわち、

      hash（m） = hash（m’）（ただし、$m \neq m'$）

      となるような m’ を求めるのが困難であること。

2.5 ハッシュ関数とデジタル署名

- 2.5.1 ハッシュ関数
  - 任意の長さのデータを固定長(128-512ビット程度)に圧縮する関数のこと。以下の性質を持つ。
    - 衝突困難性: 同じ出力値を生成する2つの入力値を発見することが困難セあること。

      すなわち、

      hash（m） = hash（m’）（ただし、$m \neq m'$）

      を満たすm, m’ を求めるのが困難であること。
  - ハッシュ関数で生成された値をハッシュ値という。
- （教科書p.55の図2.31を参照）

2.5 ハッシュ関数とデジタル署名

- 2.5.1 ハッシュ関数
  - ハッシュ関数の具体例
    - MD4, MD5: 128ビット
    - SHA: 1993年にFIPS Publication 180として策定。128ビット。
    - SHA-1: 1995年にFIPS Publication-180-1として策定。160ビット。
    - SHA-224, 256, 384, 512（SHA-2）: アルゴリズムはMD5, SHA-1 を継承。鍵長が異なる。
    - SHA-3: 2012年に次世代のハッシュ関数として策定。

2.5 ハッシュ関数とデジタル署名

- 2.5.1 ハッシュ関数
  - ハッシュ関数の利用例
    - パスワード認証におけるハッシュ関数の利用
      - システムのパスワードを平文ではなくパスワードのハッシュ値として保持しておき、認証の際はその都度入力されたパスワードのハッシュ値を求めて保存されているハッシュ値と比較を行う。
    - ハッシュ関数による改竄検知
      - 改竄を検知したいファイルのハッシュ値を求めておき、ファイルに改竄があるかを確認したいときに再度ハッシュ値を求め、元のハッシュ値と比較する。
- （教科書p.57の図2.32-2.33を参照）

2.5 ハッシュ関数とデジタル署名

- 2.5.1 ハッシュ関数
  - ハッシュ関数の利用例
    - メッセージ改竄検知
      - 送信するメッセージm と共通鍵暗号方式の暗号化鍵Kとに、ハッシュ関数hashを適用して作るMAC（Message Authentication Code）と呼ばれる付加情報と、送信するメッセージm を併せて送ることにより、メッセージmの真正性と完全性を保証する仕組み。
- （教科書p.58の図2.34を参照）

2.5 ハッシュ関数とデジタル署名

- 2.5.2 デジタル署名
  - メッセージのハッシュ値を秘密鍵で暗号化することで、メッセージの改竄検知、否認防止を実現する仕組み。
  - デジタル署名方式:
    - RSA, DSA, ECDSAを用いるのが一般的
- （教科書p.59の図2.35を参照）

2.5 ハッシュ関数とデジタル署名

2.5.2 デジタル署名

デジタル署名の流れ

- 署名者はあらかじめ公開鍵P 、秘密鍵S を用意し、公開鍵は署名の検証者に安全に渡しておく。
- ハッシュ関数を用いて、送信する平文M のハッシュ値を求める。
- 署名者はあらかじめ用意しておいた秘密鍵S を用いて、ハッシュ値H を暗号化し、署名値H_Sを得る。
- 平文M と署名値H_Sを検証者に送る。
- 検証者は平文M からハッシュ値H’ を求める。
- 署名者が発行した公開鍵P を用いて署名値H_Sを復号し、ハッシュ値H を求める。なお、秘密鍵S で暗号化したデータは秘密鍵S とペアとなる公開鍵P でしか復号できないことに注意。
- 検証者は、署名から得られたハッシュ値H と平文M から求めたH’ が一致するかどうかを確認する。（不一致なら、何らかの攻撃があったとみなす）

2.5 ハッシュ関数とデジタル署名

- 2.5.2 デジタル署名
  - デジタル署名では、公開鍵暗号方式として何を使い、ハッシュ関数として何を使うか組み合わせる必要がある。
- （教科書p.59の表2.6を参照）

2.6 暗号技術における安全性

- 2.6.1 等価安全性
  - 異なる暗号技術を同一尺度で安全性を示す概念。
  - 評価対象の暗号技術に対して、最も効率的な攻撃を行った場合に解読に必要な計算量として評価される。
    - （例）計算量が2^xの場合は「xビット安全性」という。
- （教科書p.60の表2.7を参照）

2.6 暗号技術における安全性

- 2.6.2 暗号技術の利用における安全性
  - 設計段階での安全性
    - システムの設計段階で暗号技術を安全に導入すること。
  - 実装段階での安全性
    - システムの実装段階で暗号技術を安全に導入すること。
  - ITシステムにおける暗号アルゴリズムの安全性の評価
    - システムの運用段階で暗号技術が安全に導入されているかを評価すること。
- （教科書p.62の図2.36を参照）

【次回予告】
第5回
認証技術
（第3章）

また来週！