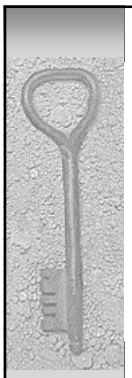
PKI: 基礎と応用

稲村雄
International Network Security, Inc.
mailto:jane@insi.co.jp

本日の講演内容

- ***PKI (Public Key Infrastructure)という概念***
- PKI要素技術
- PKIとは何か
- PKI ***Protocols/Applications***
- 実社会への ***implications***

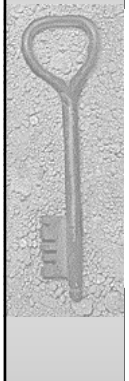
2



PKI *(Public Key Infrastructure)* という概念

- 話者／文脈により、意味のぶれ幅が非常に大きい単語
  - 認証機関(CA)の存在およびその構造自体
  - 公開鍵暗号技術(***Public Key*** Cryptography)利用環境(HW&SW)の総体
  - ***Infrastructure***=社会基盤としての公開鍵暗号技術利用可能性
  - 公開鍵暗号技術そのもの
  - ECで儲けるための新手の魔法のことば
    - ***FW, Vaccine Soft, IDS, etc.***に続く

3



PKI *(Public Key Infrastructure)* という概念 ***contd.***

- 話者／文脈により、意味のぶれ幅が非常に大きい単語
  - 認証機関(CA)の存在およびその構造自体
  - 公開鍵暗号技術(***Public Key*** Cryptography)利用環境(HW&SW)の総体
  - ***Infrastructure***=社会基盤としての公開鍵暗号技術利用可能性
  - ~~公開鍵暗号技術そのもの~~
  - ~~ECで儲けるための新手の魔法のことば~~
    - ~~***FW, Vaccine Soft, IDS, etc.***に続く~~

4

社会基盤としての公開鍵暗号技術利用可能性

- 交通機関／通信／電力など、他の社会基盤と同じレベルの公開鍵暗号技術を(主として)インターネット上に提供
  - 安全な要素技術の設計および実装
    - 対称／非対称暗号
    - ハッシュ関数
  - 安全なProtocol/Applicationの設計および実装
  - 運用
  - ユーザ(=**セキュリティに関する最も弱いリンク**)
    - 利用環境整備
    - 教育

PKIに関するよくある誤解

- 公開鍵暗号は共通鍵暗号より優れている
- 公開鍵暗号ならば、鍵の配布は問題ない
- デジタル証明書は秘密にしてないと危険
- デジタル証明書を送付すると認証される
- PKIさえ導入すれば、セキュリティは万全

本日の講演内容

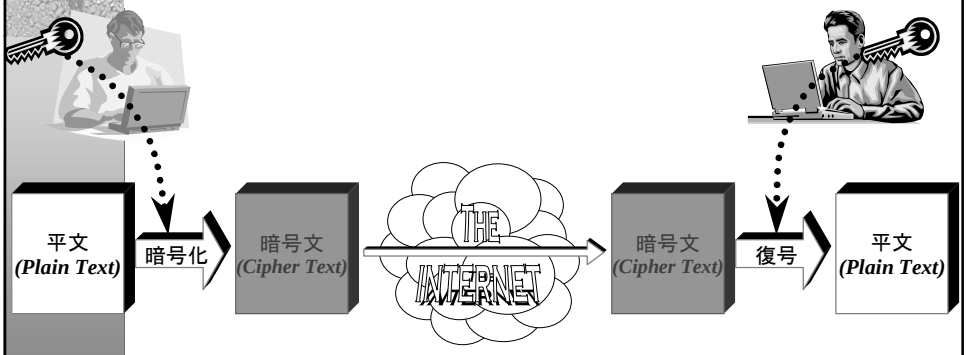
- PKI (***Public Key Infrastructure***)という概念
- ***PKI要素技術***
- PKIとは何か
- PKI ***Protocols/Applications***
- 実社会への ***implications***

PKI要素技術

- 対称暗号
- 公開鍵配布系
- 非対称暗号
- 一方向ハッシュ関数

対称暗号

- ＝秘密鍵暗号／共通鍵暗号／慣用暗号
- 暗号化／復号に同じ鍵が用いられる



9

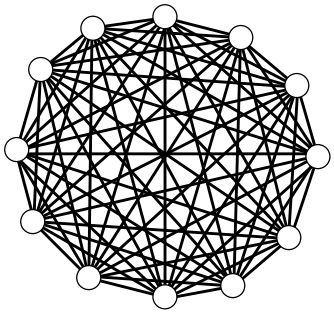
対称暗号の問題点

- 基本的方式として、鍵と平文／暗号文との間で複雑な演算を行なうことで暗号化／復号を実現
- ブロック暗号とストリーム暗号の二種類に大別
- 問題点 **1:** 如何にして鍵を安全に共有できるか?

10

対称暗号の問題点 *contd.*

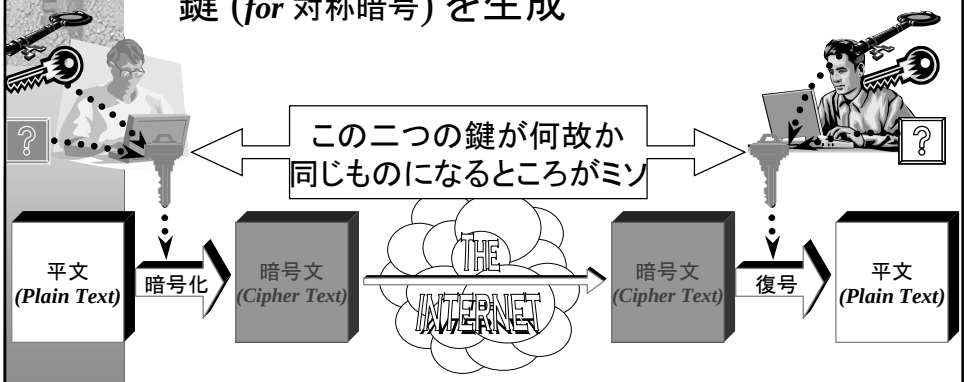
- 問題点 2: 相通信するペア毎に異なる鍵が必要
  - ➡必要な鍵数の爆発

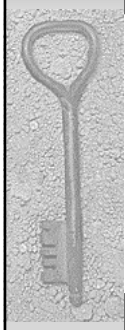


$$\frac{N(N-1)}{2}$$

公開鍵配布系

- 送受信者双方が個々に公開している情報から、その二者のみしか知り得ない秘密の鍵 (*for* 対称暗号) を生成





公開鍵配布系 *contd.*

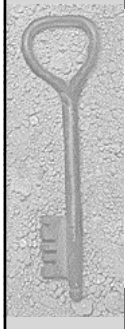
♦ ***Diffie-Hellman* 方式**

– ***W.Diffie* と *M.Hellman* によって考案された世界初のアルゴリズム** (1976)

– **離散対数問題の困難性に依拠**

離散対数問題：

α, q, yが既知整数のとき、$y = \alpha^x \pmod q$を満たすような整数 xを見付ける



公開鍵配布系 *contd.*

♦ ***Diffie-Hellman* 方式** *contd.*

大きな素数 q と $GF(q)$ 上の原始元αを決定
各ユーザは $GF(q)$から任意の整数 X_p を選び、
$K_p \equiv \alpha^{X_p} \pmod q$を公開情報とする
AがBにメッセージを送る:
$K_{AB} \equiv (K_B)^{X_A} \pmod q$を暗号化鍵とする
BがAからメッセージを受ける:
$K_{BA} \equiv (K_A)^{X_B} \pmod q$を復号鍵とする
K_{AB}とK_{BA}は等しく、また、A, B以外がこの値を計算するにはK_AもしくはK_Bの対数計算が必要

公開鍵配布系 *contd.*

♦ ***Diffie-Hellman* 方式** *contd.*

$q = 11, GF(q) = \{1,2,3,4,5,6,7,8,9,10\}, \alpha = 2$

$X_A = 3, X_B = 4, K_A \equiv 2^{X_A} (\mathrm{mod} 11) = 8, K_B \equiv 2^{X_B} (\mathrm{mod} 11) = 5$

*A*が*B*にメッセージを送る:

$$K_{AB} \equiv (K_B)^{X_A} (\mathrm{mod}\, q) = 5^3 (\mathrm{mod}\, q) = 4$$

*B*が*A*からメッセージを受ける:

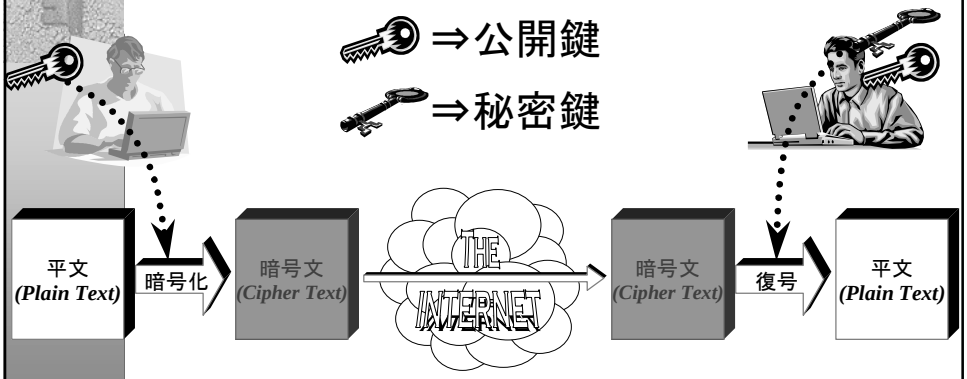
$$K_{BA} \equiv (K_A)^{X_B} (\mathrm{mod}\, q) = 8^4 (\mathrm{mod}\, q) = 4$$

15

非対称暗号

♦ ＝公開鍵暗号

♦ 暗号化／復号に異なった鍵が用いられる



16

非対称暗号 *contd.*

♦ ***RSA* 方式**

大きな素数 P,Q を選択

$N \equiv P \cdot Q, \quad \Phi(N) = (P-1)(Q-1)$

$\Phi(N)$ に関して互いに素な数 E を選ぶ

$E \cdot D = 1 \pmod{\Phi(N)}$ を満たす数 D を選ぶ

暗号化処理: $C = M^E \pmod{N}$

復号処理: $M = C^D \pmod{N}$

非対称暗号 *contd.*

♦ ***RSA* 方式** *contd.*

E を選ぶ(通常 $2^{16}+1$)

大きな素数 P,Q を選択

条件:“ $P-1, Q-1$ がそれぞれ E と互いに素 ” を満たす

$N \equiv P \cdot Q, \quad \Phi(N) = (P-1)(Q-1)$

$E \cdot D = 1 \pmod{\Phi(N)}$ を満たす数 D を選ぶ

暗号化処理: $C = M^E \pmod{N}$

復号処理: $M = C^D \pmod{N}$

非対称暗号 *contd.*

♦ ***RSA*** 方式 *contd.*

$E=3, P=5, Q=11$

$N \equiv P \cdot Q = 55, \quad \Phi(N) = (P-1)(Q-1) = 40$

$E \cdot D = 1 \pmod{\Phi(N)}$を満たす数$D = 27$

$M = 5$

暗号化処理: $C = 5^3 \pmod{55} = 15$

復号処理: $M = 15^{27} \pmod{55} = 5$

19

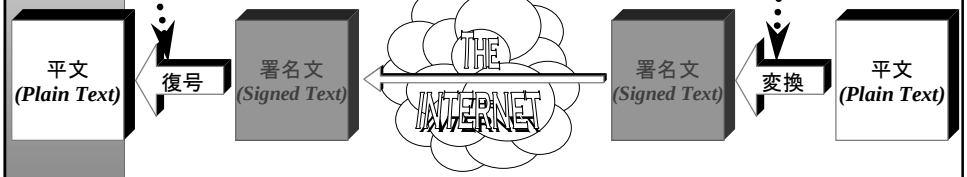
非対称暗号 *contd.*

♦ デジタル署名

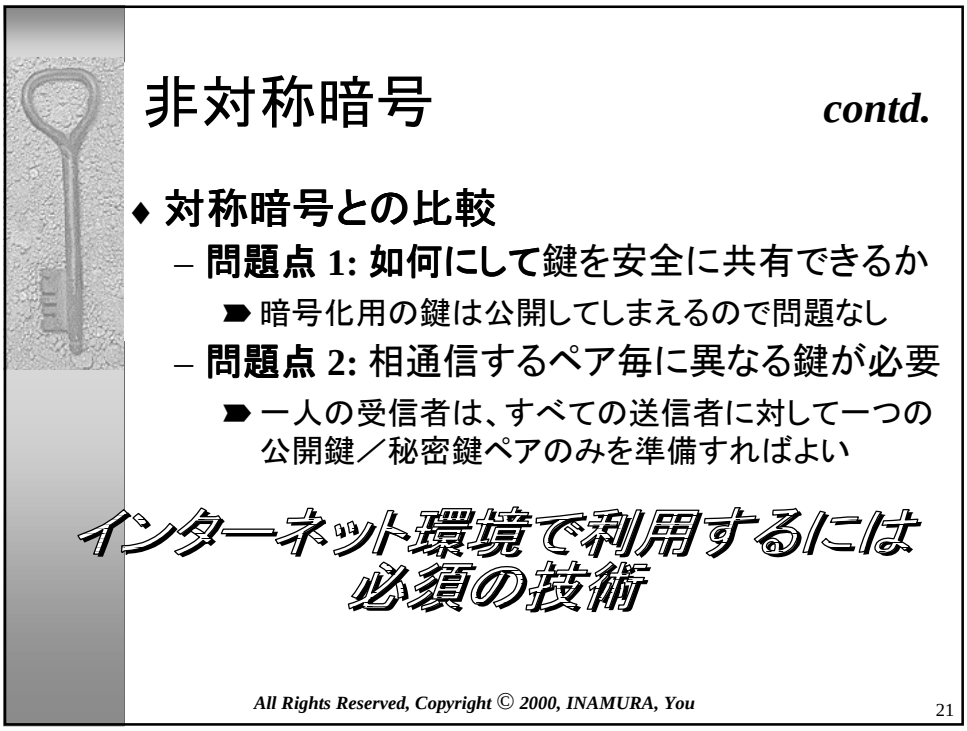
♦ 署名者の身許証明およびデータ完全性保証が可能


⇒公開鍵


⇒秘密鍵



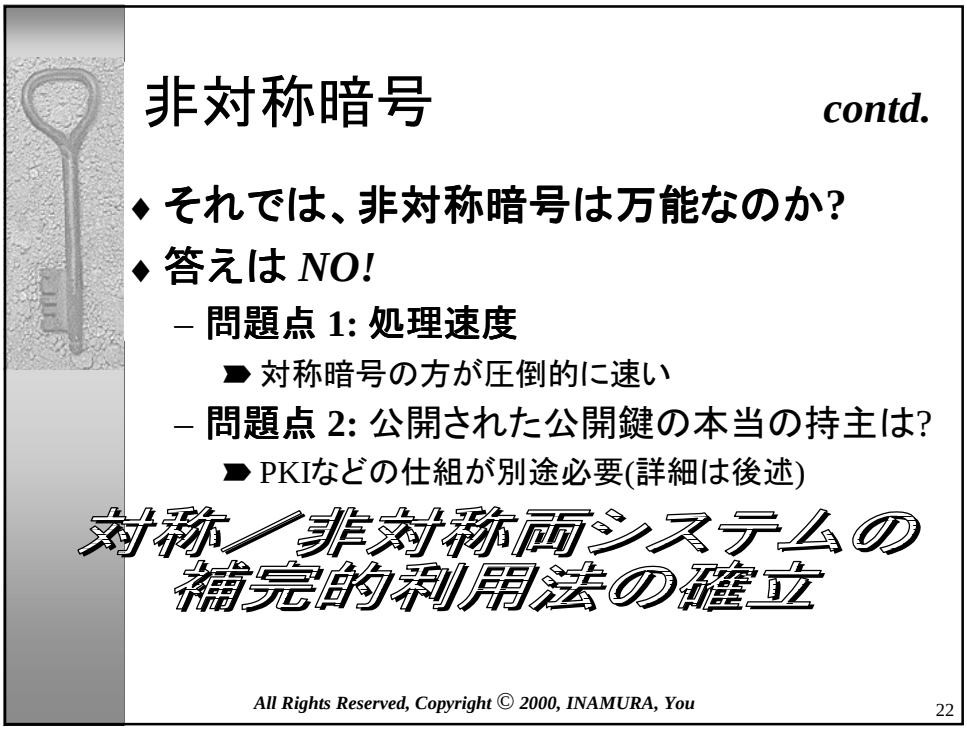
20



非対称暗号 *contd.*

- **対称暗号との比較**
  - **問題点 1: 如何にして**鍵を安全に共有できるか
    - 暗号化用の鍵は公開してしまえるので問題なし
  - **問題点 2:** 相通信するペア毎に異なる鍵が必要
    - 一人の受信者は、すべての送信者に対して一つの公開鍵／秘密鍵ペアのみを準備すればよい

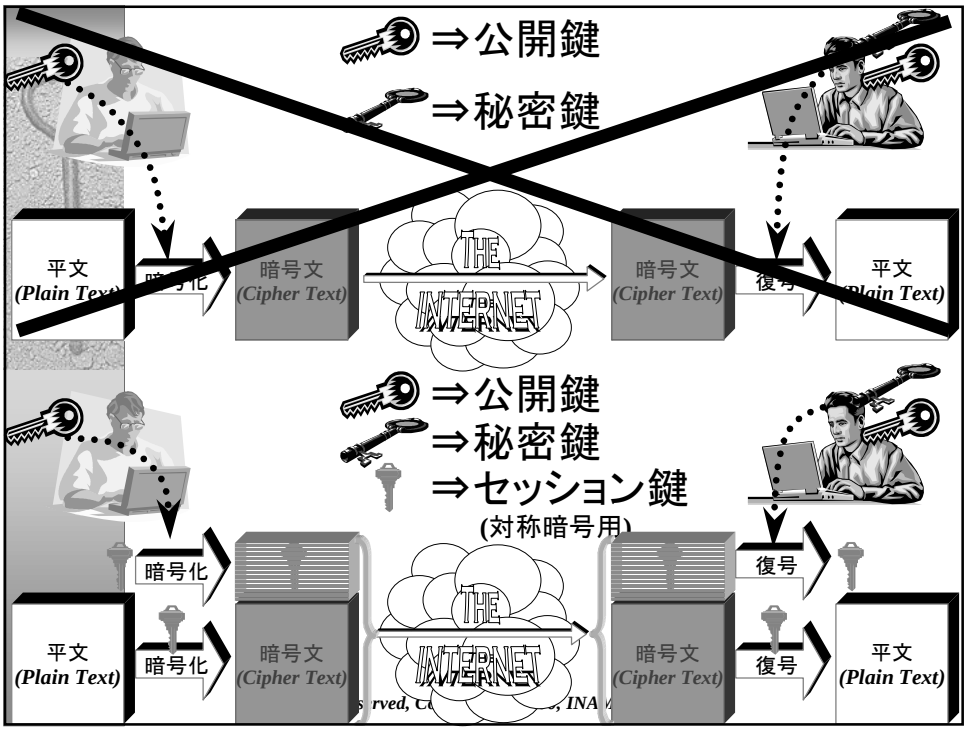
インターネット環境で利用するには必須の技術



非対称暗号 *contd.*

- **それでは、非対称暗号は万能なのか?**
- **答えは *NO!***
  - **問題点 1: 処理速度**
    - 対称暗号の方が圧倒的に速い
  - **問題点 2:** 公開された公開鍵の本当の持主は?
    - PKIなどの仕組が別途必要(詳細は後述)

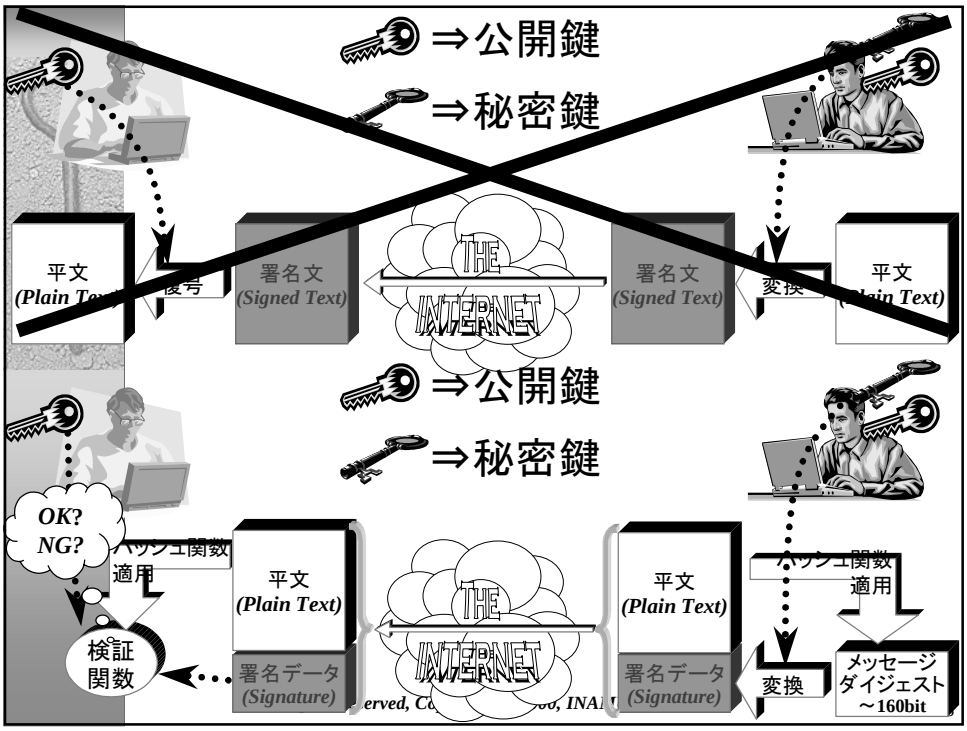
対称／非対称両システムの補完的利用法の確立



デジタル・エンベロープ

- 対称アルゴリズムの速度と非対称アルゴリズムの柔軟性
  - ＝デジタル・エンベロープ(封筒)方式
  - ―対称／非対称両システム補完的利用法の一例

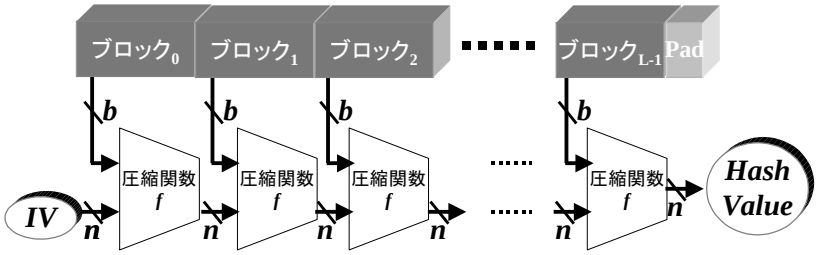
24



一方向ハッシュ関数

- ＝メッセージダイジェスト
- 任意のデータから、そのデータ特有とみなせる短い (百数十ビット程度) 情報(=メッセージダイジェスト)を抽出する技術

b=512bit
n=128～160bit
あたりが一般的



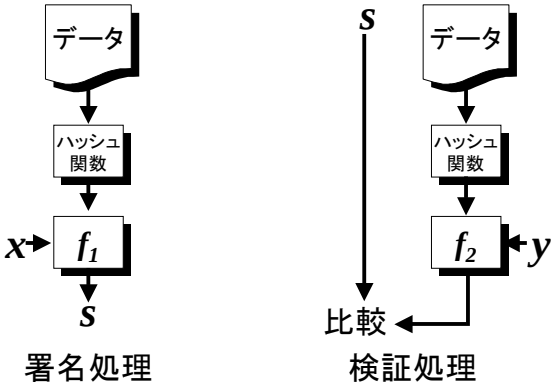
26

一方向ハッシュ関数 *contd.*

- MD2/4/5
  - ***R. Rivest*** らによるアルゴリズム
  - 128bit 長のメッセージダイジェストを抽出
    - 通常攻撃に 2^{128}、誕生日攻撃に 2^{64}
- SHA-1
  - 米国 NIST が NSA とともに開発したアルゴリズム
  - 160bit 長のメッセージダイジェストを抽出
    - 通常攻撃に 2^{160}、誕生日攻撃に 2^{80}
- より長いメッセージダイジェスト(256bit～512bit)も近年提案されつつある

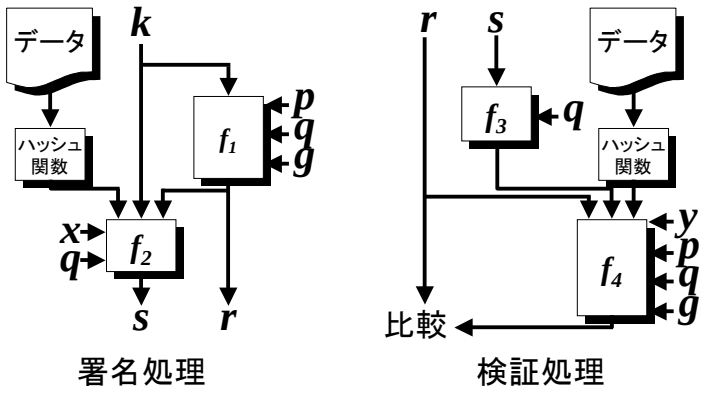
デジタル署名方式

- ***RSA*型**



デジタル署名方式 *contd.*

- ***DSA*型**



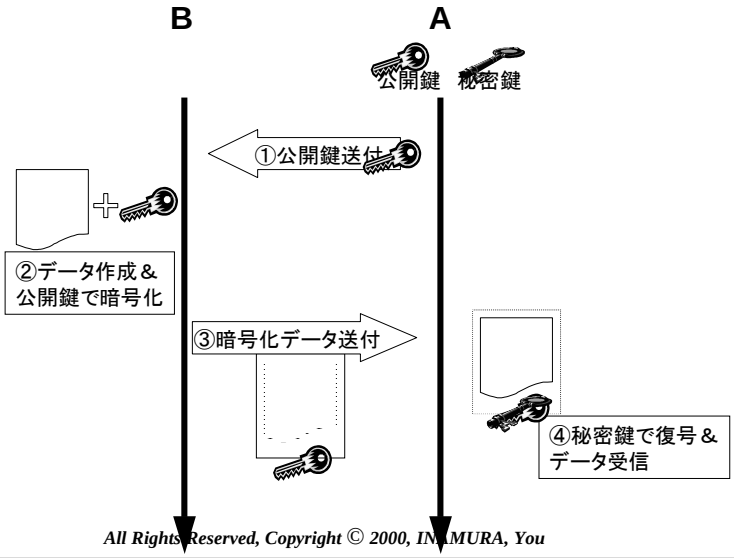
本日の講演内容

- PKI (***Public Key Infrastructure***)という概念
- PKI要素技術
- ***PKI***とは何か
- PKI Protocols/Applications
- 実社会への ***implications***

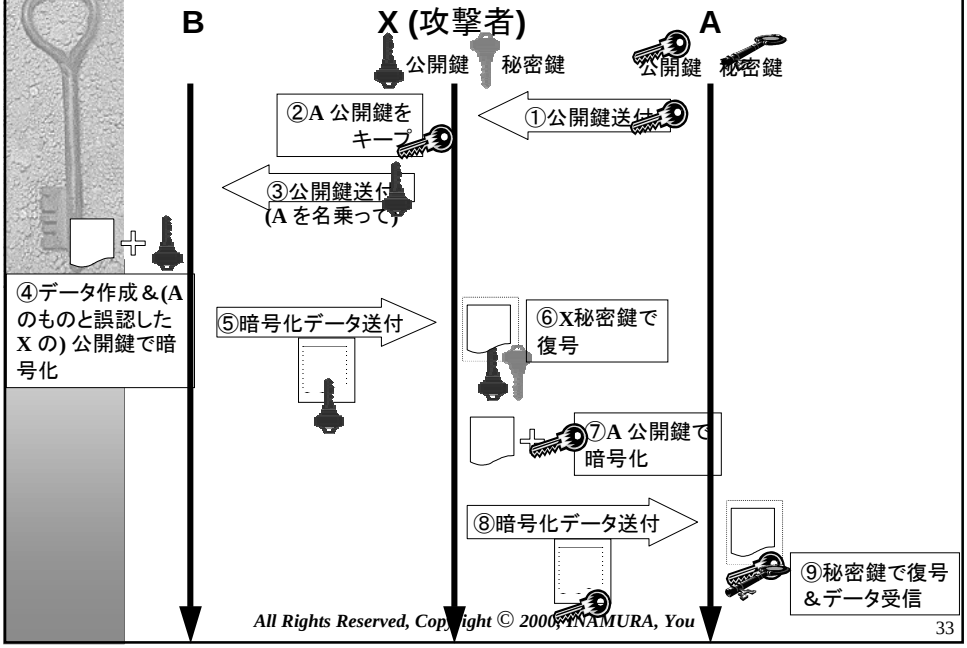
非対称暗号 *reprise*

- それでは、非対称暗号は万能なのか?
- 答えは ***NO!***
  - 問題点 **1:** 処理速度
    - ➡ 対称暗号の方が圧倒的に速い
  - 問題点 **2:** 公開された公開鍵の本当の持主は?
    - ➡ PKIなどの仕組が別途必要(詳細は後述)

犯しやすい過ち



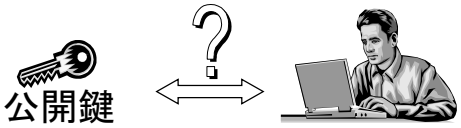
Man-in-the-middle Attack



33

非対称暗号のアキレス腱

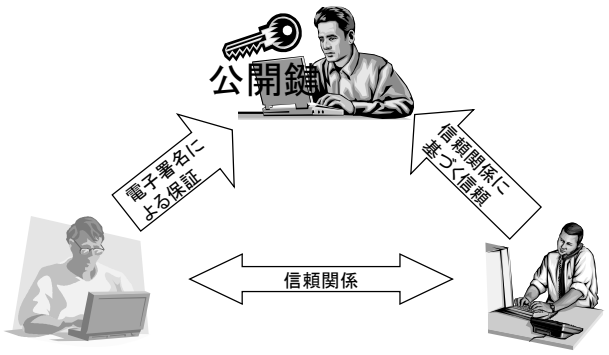
- ある公開鍵の持ち主が本当に申告通りの人物であるかどうか？
  - インターネットEC等の成否は、この結び付きを誤りなく判断できるかどうかに係っている
- 虚偽の申告が世間で受け入れられた場合、以降その人物になりすますことが可能に



34

草の根的解決 (PGP型)

- 『友達の友達は友達』方式



35

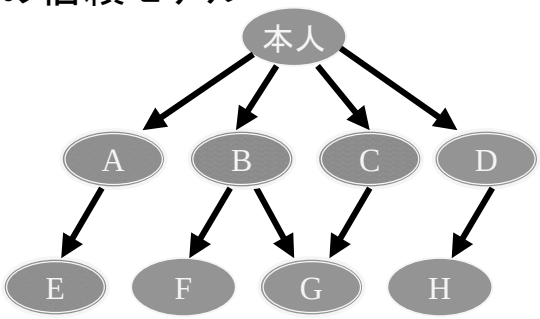
草の根的解決 (PGP型) *contd.*

- “草の根的解決”の特徴
  - とりあえず他の仕組が要らない
    - 初期段階での普及は容易
  - ユーザ層が拡大すると、信頼性に不安が
    - 友達の友達の友達の…が信頼する鍵は信頼できる？
    - 個人への信頼と、その個人による保証への信頼とを区別する

36

草の根的解決 (PGP型) *contd.*

♦ PGP の信頼モデル



X → Y　X が Y に署名

信頼できる鍵

信頼できない鍵

署名者として信頼

署名者として部分的に信頼

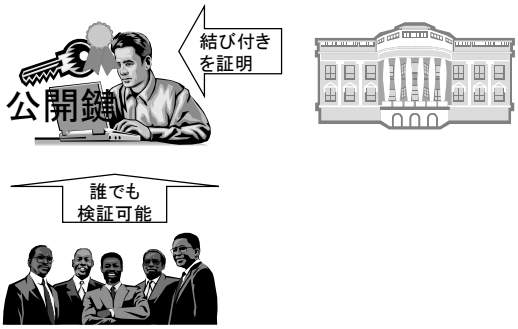
署名者として信頼しない

37

デジタル証明書/CA/PKI

♦ CA (Certification Authority, 認証機関) が、ユーザの身許を確かめた上で、“お墨付き” (= デジタル証明書) を発行

– 誰でもユーザ⇔公開鍵の関係が検証可能に



38

デジタル証明書/CA/PKI *contd.*

♦ X.509v3 証明書

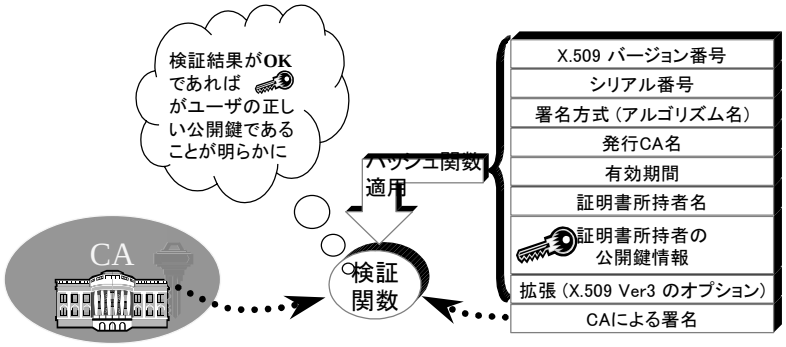
記載されるのは基本的に公開情報のみ∴誰にでも公開可能

– ITU-T X.509 勧告で定義されたフォーマット
– 同勧告第3版が現在事実上の証明書標準
– Internet上の標準“PKIX”もX.509v3ベース

X.509 バージョン番号	… … …	X.509 のバージョン（現行はv3）
シリアル番号	… … …	CAごとに一意に証明書を指定
署名方式（アルゴリズム名）	… … …	証明書に対する署名方式
発行CA名	… … …	証明書を発行したCAの名前
有効期間	… … …	証明書の有効期間（開始／終了日時）
証明書所持者名	… … …	証明書および公開鍵所有者の名前
証明書所持者の公開鍵情報	… … …	証明書所持者の公開鍵データそのもの
拡張（X.509 Ver3 のオプション）	… … …	拡張フィールド(パス検証処理補助情報などに利用)
CAによる署名	… … …	上記全項目に対してCAが施した電子署名

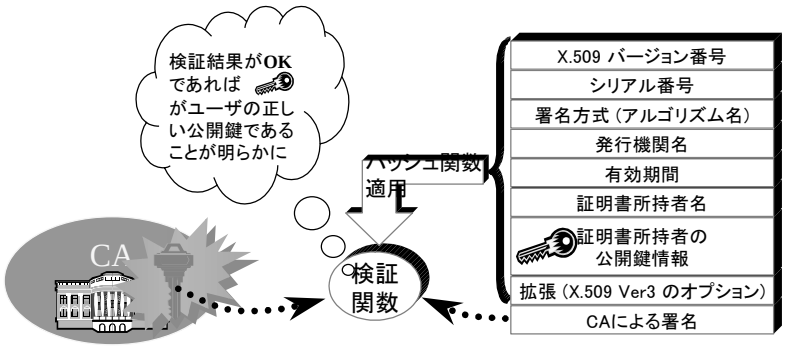
デジタル証明書/CA/PKI *contd.*

♦ 証明書の検証



デジタル証明書/CA/PKI *contd.*

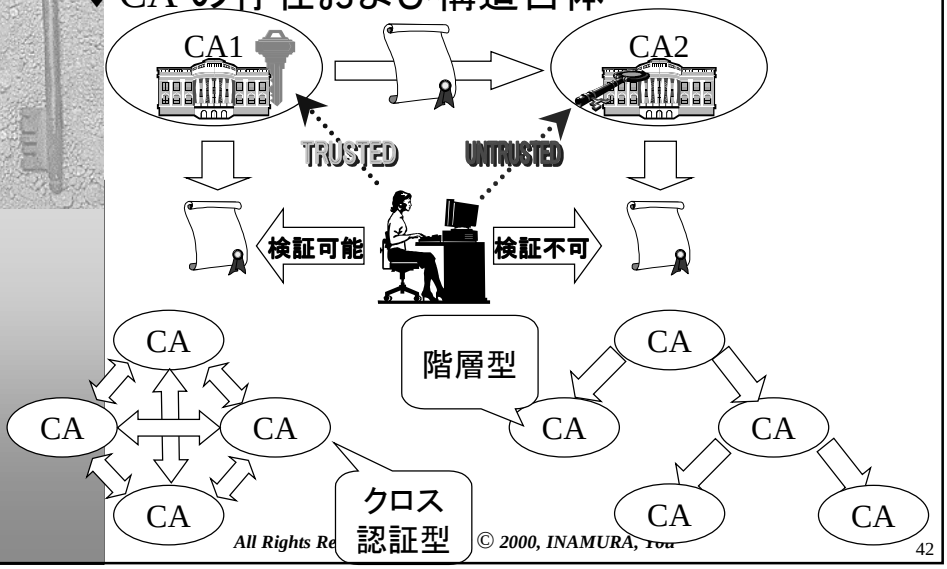
- CA 自体の公開鍵の保証は？



41

PKI の定義

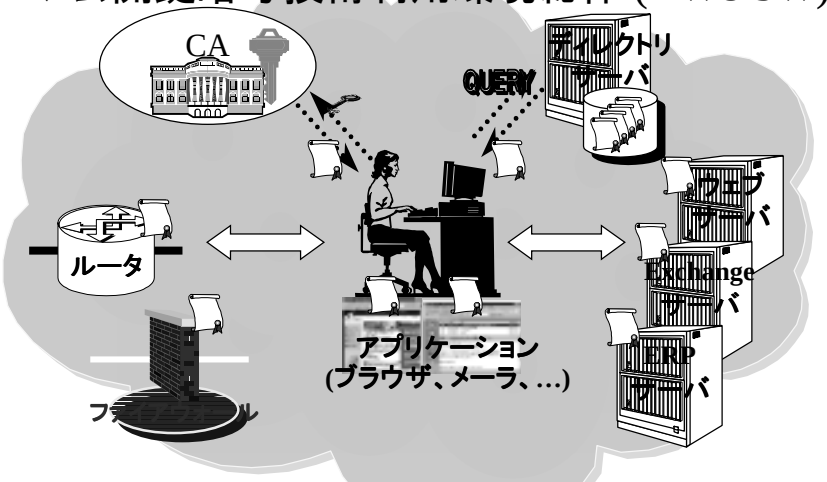
- CA の存在および構造自体



42

PKI の定義 *contd.*

- 公開鍵暗号技術利用環境総体 (HW&SW)



43

本日の講演内容

- PKI (***Public Key Infrastructure***)という概念
- PKI要素技術
- PKIとは何か
- ***PKI Protocols/Applications***
- 実社会への ***implications***

44

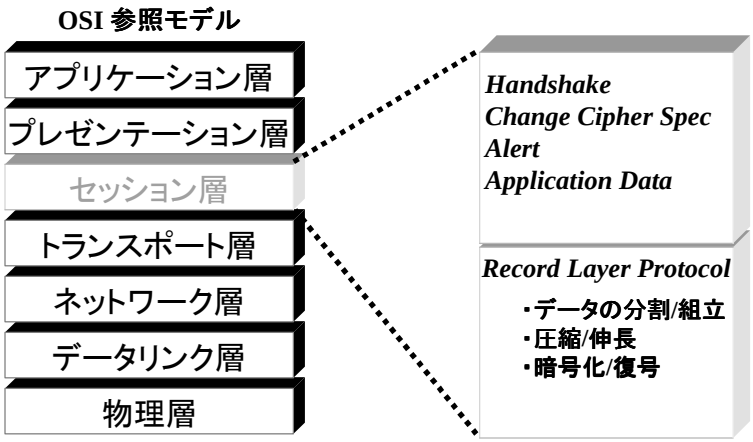
PKI *Protocols/Applications*

- S/MIME (***Secure/MIME***)
  - 電子メールに対する暗号化／署名付与
- Code Signing
  - Java/ActiveX等に対する保護
- Document Signing
  - PDF等電子書類に対する保護
- SSL (***Secure Sockets Layer***)
  - セキュアな汎用プロセス間通信機構
- TLS (***Transport Layer Security***)
  - SSL後継規格
- IPSec (***IP Security Protocol***)
  - IPレベルでのセキュリティ機能付与
- PKIX (***Public Key Infrastructure X.509***)
  - Internet上での相互運用可能なPKI構築

SSL *(Secure Sockets Layer)*

- *Netscape* 社提唱のセキュリティ・プロトコル
- 特徴
  - サーバ／クライアント・モデルでの利用
  - 相互運用性確保のためデジタル証明書を利用
  - 対称／非対称の両アルゴリズムを併用
  - 多くのウェブ・サーバ／クライアントに実装

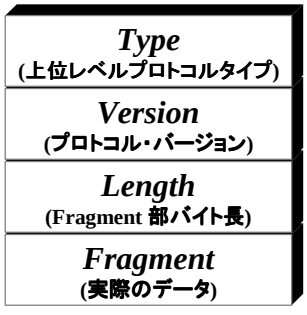
SSL プロトコル概要



47

SSL プロトコル概要 *contd.*

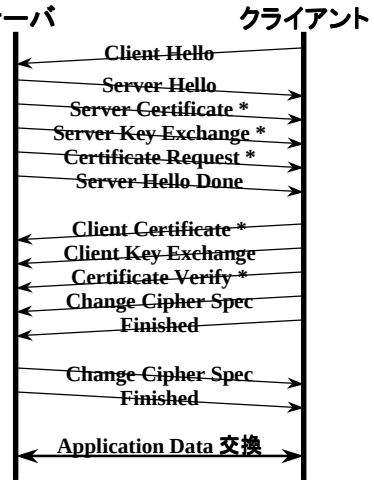
- ***Record Layer Protocol***
  - 2^{14} Byte 以下にデータを分割 (もしくは複数を統合)
  - (必要なら) 圧縮処理
  - (必要なら) 認証データ付加
  - (必要なら) 暗号化処理
  - 右図のようなデータ構造体を生成して送受



48

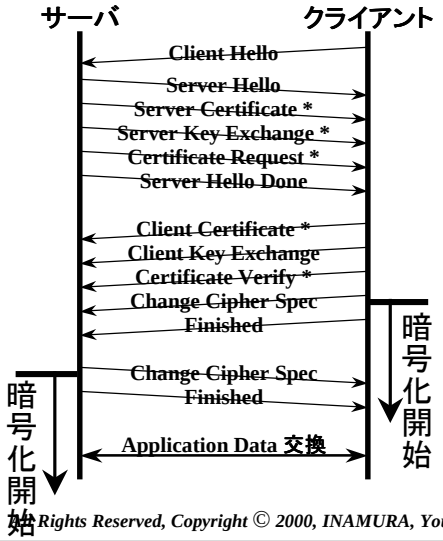
SSL プロトコル概要 *contd.*

- ***Handshake Protocol***
  - SSL の実質的な要
  - Client/Server 間の接続を確立
    - 暗号化アルゴリズム決定
    - セッション鍵の生成
    - 互いの認証



49

SSL プロトコル概要 *contd.*



50

SSL プロトコル概要 *contd.*

- ***C/S Hello***
  - 暗号化／圧縮アルゴリズムの決定
    - Clientが利用可能なアルゴリズムをリストとして通知し、Serverはその中から実際に利用するアルゴリズムを選択
  - 時間データを含み、接続のフレッシュさを保証

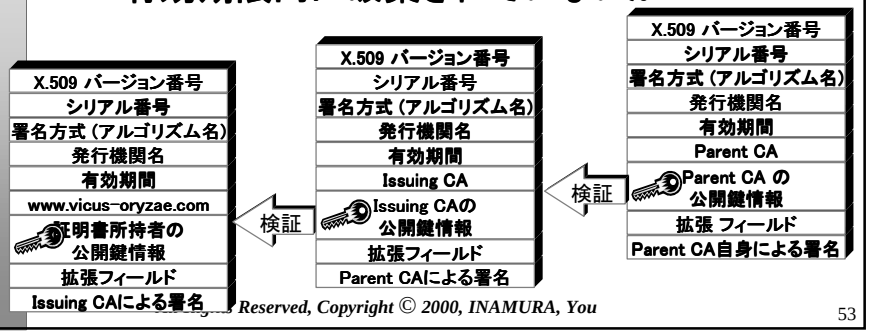
= **C/S Random**
マスタ・シークレット生成に利用

SSL プロトコル概要 *contd.*

- ***Server Certificate***
  - サーバ自身⇒CAの順番にすべての証明書を羅列
  - 証明書列を順番に利用することで、最終的にサーバ自身の証明書を検証

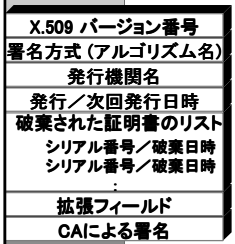
SSL プロトコル概要 *contd.*

- 検証
  - サーバ名が証明書のものと一致するか
  - 有効期限は切れていないか
  - 証明書チェーンにより順番に署名検証できるか
  - **有効期限内に破棄されていないか**



SSL プロトコル概要 *contd.*

- 破棄の検証
  - 秘密鍵漏洩などのため、有効期限到達前に証明書の破棄が必要になる可能性がある
  - 一般的なのは CRL (***Certificate Revocation List***)の利用
    - 中途破棄された証明書をCAが署名付リストで公表
    - 差分CRL, CRL配布ポイントなどの改良版が考案中
  - オンライン確認プロトコルも
    - OCSP (***Online Certificate Status Protocol***)
    - SCVP (***Simple Certificate Validation Protocol***)
    - 正しい情報源からの回答かどうか確かめるのが厄介
    - CRT (Certificate Revocation Tree)というかなり賢い方式もある



CRL

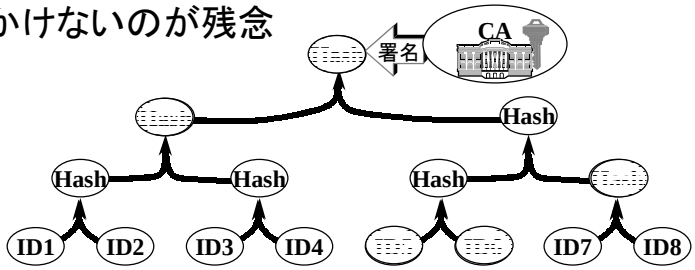
SSL プロトコル概要 *contd.*

♦ ***CRT***

– CAが破棄証明書のハッシュ値を元に木を生成後、ルートに署名

– 部分木を知るだけで破棄された否かを確実に判断可能

– 特許で保護されているため、一部製品でしか見かけないのが残念



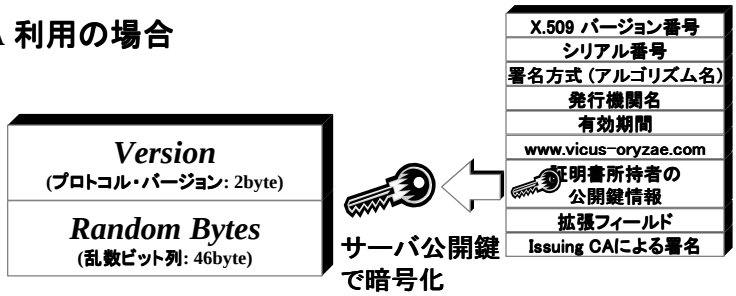
55

SSL プロトコル概要 *contd.*

♦ ***Client Key Exchange***

– マスタ・シークレットを生成する元となるデータ(プリ・マスタ・シークレット) をクライアントが生成し、サーバ公開鍵で暗号化した上で送付

RSA 利用の場合



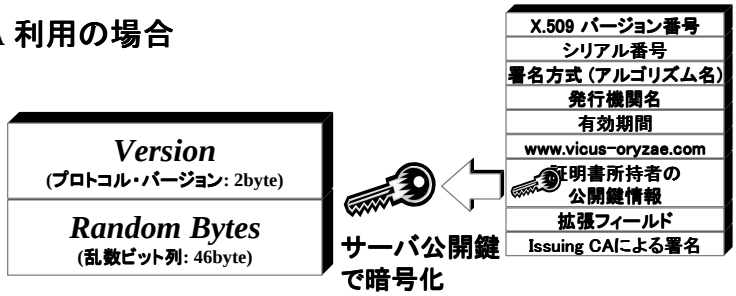
56

SSL プロトコル概要 *contd.*

♦ ***Client Key Exchange***

– マスタ・シークレットを生成する元となるデータ (プリ・マスタ・シークレット) をクライアントが生成し、サーバ公開鍵で暗号化した上で送付

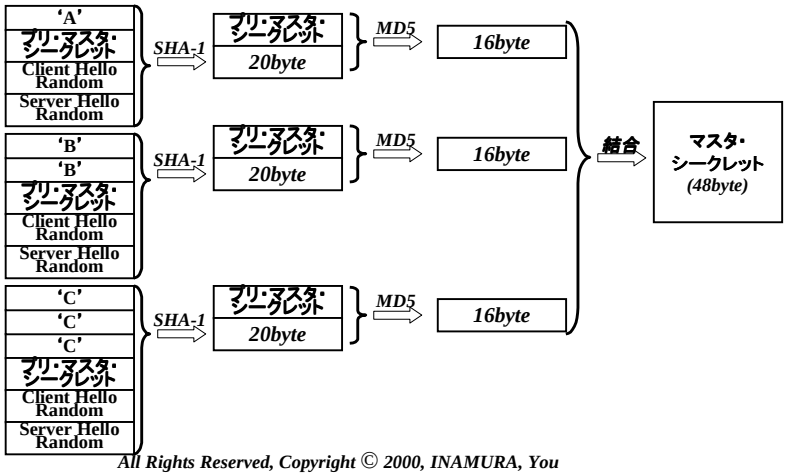
RSA 利用の場合



57

SSL プロトコル概要 *contd.*

♦ マスタ・シークレットの生成



58

SSL プロトコル概要 *contd.*

- キーブロックの生成
  - 暗号化鍵、初期化ベクタ、MAC 計算用秘密データなどとして利用するデータの生成処理
  - 各種データに十分な量に達するまで、以下の計算を行う

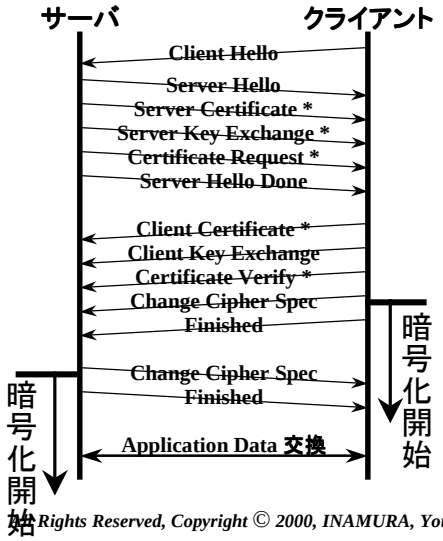
$$
\begin{aligned}
Key\_Block = {} & MD5(MasterSecret + SHA(MasterSecret + ServerHelloRandom + \\
& \quad ClientHelloRandom + 'A')) + \\
& MD5(MasterSecret + SHA(MasterSecret + ServerHelloRandom + \\
& \quad ClientHelloRandom + 'BB')) + \\
& MD5(MasterSecret + SHA(MasterSecret + ServerHelloRandom + \\
& \quad ClientHelloRandom + 'CCC')) + \cdots
\end{aligned}
$$

+: 結合演算

SSL プロトコル概要 *contd.*

- ***Change Cipher Spec***
  - それまでのネゴシエーションで合意を得た暗号化方式の利用を開始するという合図
  - 特に相手の***ack***を待ったりはせず、次のメッセージ(***Finished***)から構わず暗号化してしまう
  - 受信側もその旨対処
    - この辺りで、下位プロトコルTCPのパケット順保証に頼っている

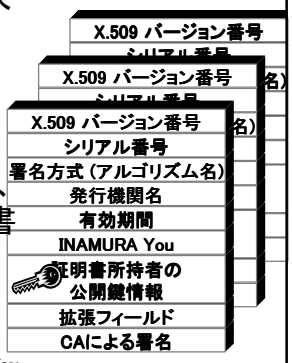
SSL プロトコル概要 *contd.*



61

SSL プロトコル概要 *contd.*

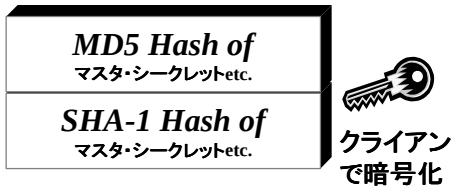
- ***Certificate Request***
  - 所謂***SSL***クライアント認証を実現するための仕組
  - クライアントに証明書の送付を要求
- ***Client Certificate***
  - クライアント自身⇒CAの順番にすべての証明書を羅列
  - 証明書列を順番に利用することで、最終的にクライアント自身の証明書を検証可能



62

SSL プロトコル概要 *contd.*

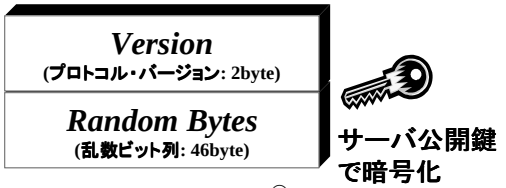
- ***Certificate Verify***
  - サーバがクライアントの認証を行なうのを補助する目的でクライアントが送付
  - **証明書の検証＝クライアントの認証ではない**
    - **証明書自体は誰でも入手可能**
    - **秘密鍵を使えることを検証して始めて身許が確認**
    - **同データをクライアント公開鍵で正しく復号できるか?**



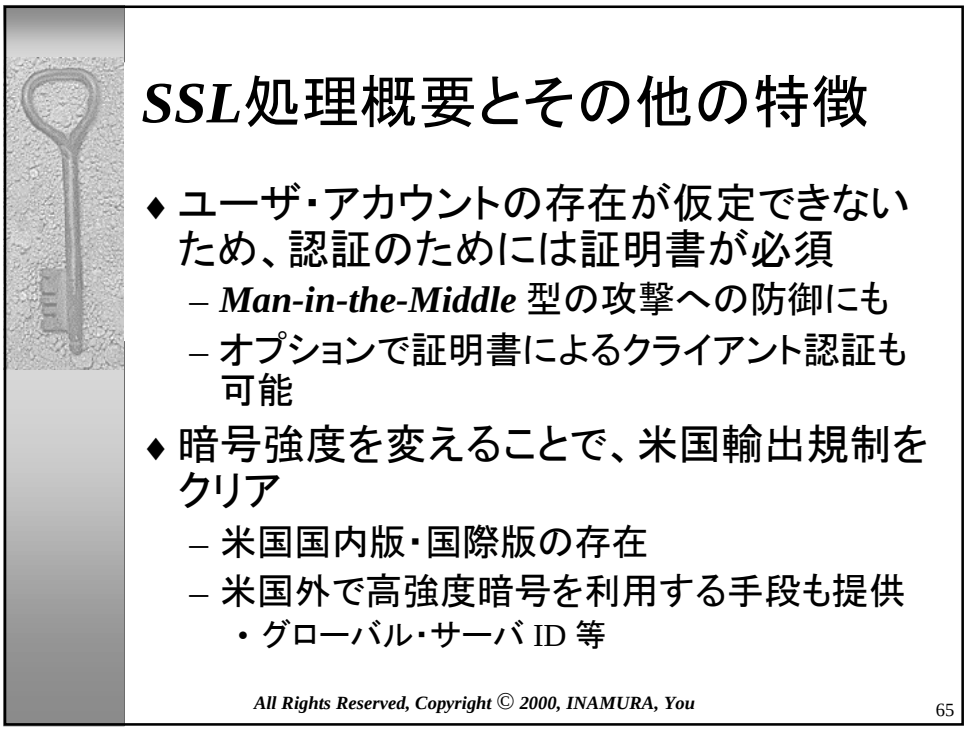
63

SSL プロトコル概要 *contd.*

- ***Certificate Verify*** *contd.*
  - それでは、何故***Server Verify***は必要ないのか？
  - 鍵は***Client Key Exchange***の構成方法
    1. サーバ公開鍵で暗号化されたデータを復号するためには、対応する秘密鍵の利用が必須
    2. 同データを元にセッション鍵などが生成される
    3. その後の通信が意味をなす⇒通信相手が正規サーバ秘密鍵を持っていることが保証

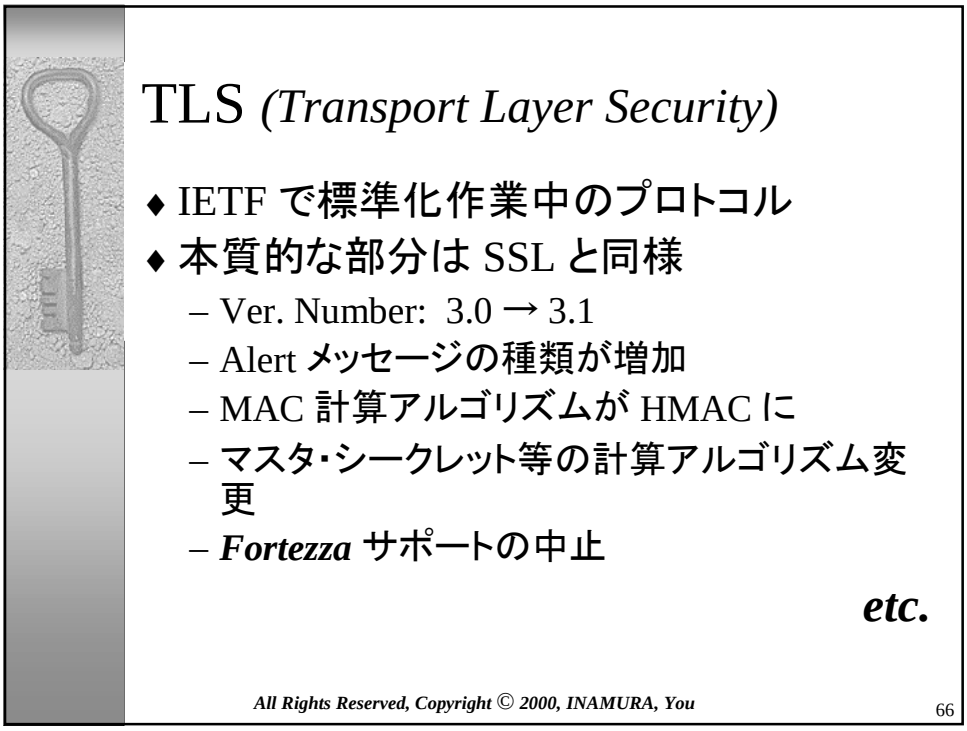


64



*SSL*処理概要とその他の特徴

- ユーザ・アカウントの存在が仮定できないため、認証のためには証明書が必須
  - ***Man-in-the-Middle*** 型の攻撃への防御にも
  - オプションで証明書によるクライアント認証も可能
- 暗号強度を変えることで、米国輸出規制をクリア
  - 米国国内版・国際版の存在
  - 米国外で高強度暗号を利用する手段も提供
    - グローバル・サーバ ID 等



TLS *(Transport Layer Security)*

- IETF で標準化作業中のプロトコル
- 本質的な部分は SSL と同様
  - Ver. Number: 3.0 → 3.1
  - Alert メッセージの種類が増加
  - MAC 計算アルゴリズムが HMAC に
  - マスタ・シークレット等の計算アルゴリズム変更
  - ***Fortezza*** サポートの中止

etc.

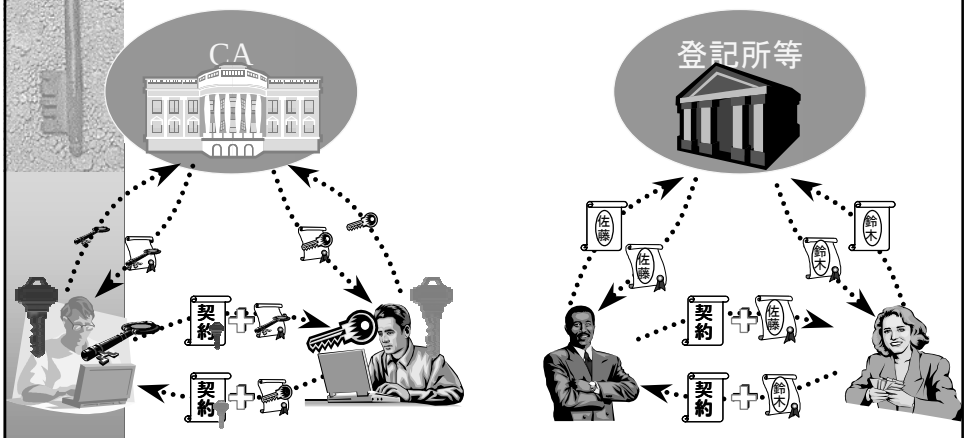
本日の講演内容

- PKI (***Public Key Infrastructure***)という概念
- PKI要素技術
- PKIとは何か
- PKI ***Protocols/Applications***
- ***実社会への implications***

PKI の定義 *contd.*

- **PKI** = ***Public Key Infrastructure***
  - インフラストラクチャとしての公開鍵暗号技術
    - 交通機関／通信／電力など、他のインフラと同じレベルでの公開鍵暗号技術利用性の提供

PKIと実社会取引との比較



69

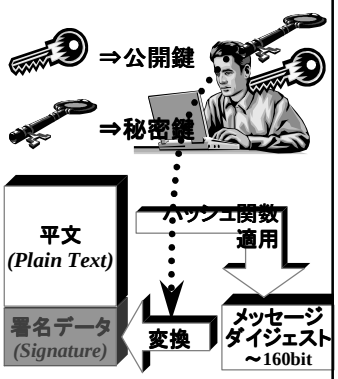
PKIに何が求められてしまうか

機能	実現方法	代替手段
機密情報の保護	暗号化による	共通鍵暗号
取引相手の身許確認	デジタル署名による	Challenge& Response等
取引内容の完全性保証	デジタル署名による	KeyedMAC等
取引相手による否認行為の防止	**デジタル署名による**	**代替手段なし**

70

PKIにおける否認防止とは

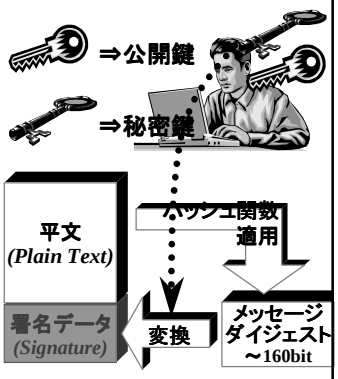
1. 本人の秘密鍵を用いずに署名データを生成することは(計算量的に)不可能
2. 秘密鍵を使い得るのは本人のみ
3. 本人の公開鍵を用いて検証できるデジタル署名が存在するならば、その署名を生成したのは本人



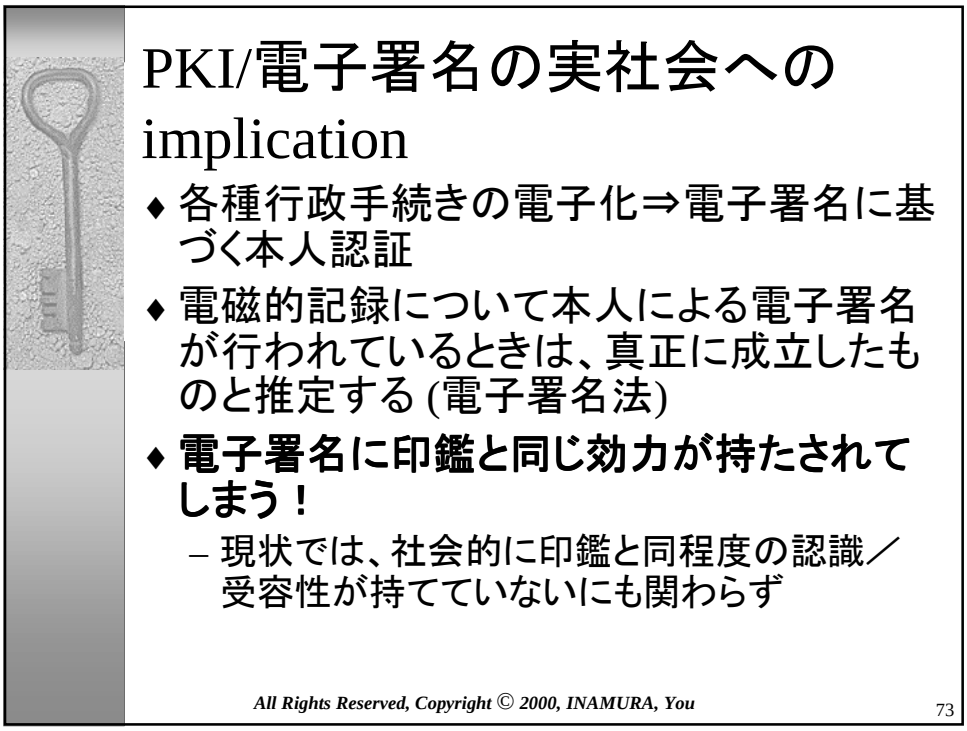
本来、純粋に技術上の概念

PKIにおける否認防止とは *contd.*

1. 十分長い鍵を使えばOK
2. 秘密鍵をどう保護するか?
   現状の一般的利用環境
   - ① Windows 9X/ME⇒OSによる保護が弱い
   - ② Windows Logon Password のみで保護⇒運用面に関する保護が弱い
3. “デジタル署名の存在⇔本人が署名を生成”という図式は成り立ち得ない

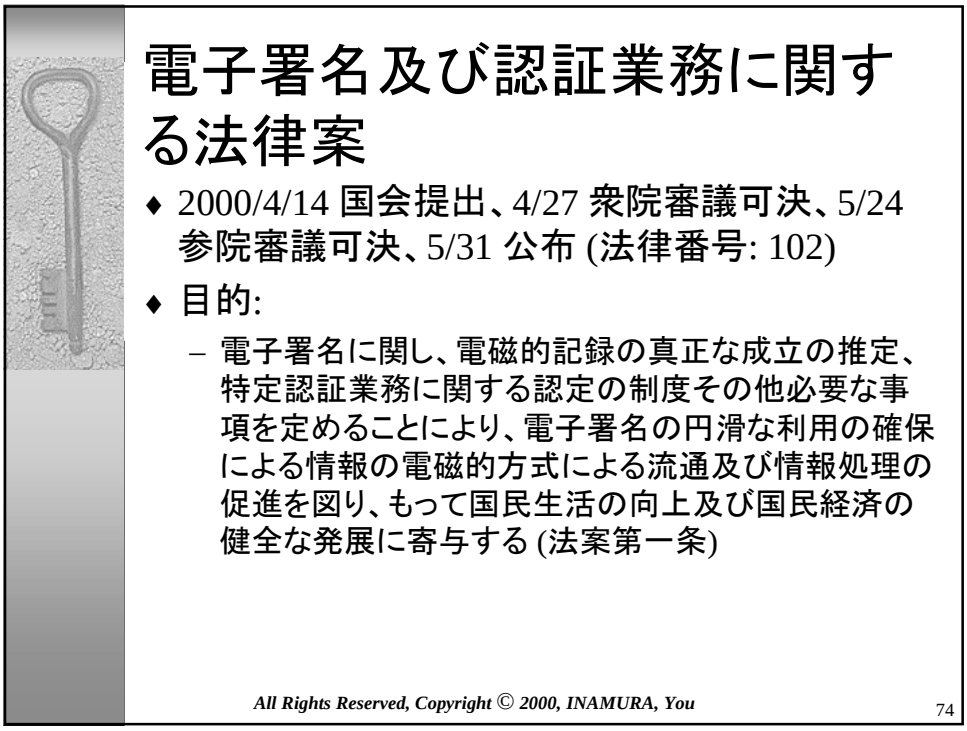


成立させるためには、社会的普及活動が必須



PKI/電子署名の実社会へのimplication

- 各種行政手続きの電子化⇒電子署名に基づく本人認証
- 電磁的記録について本人による電子署名が行われているときは、真正に成立したものと推定する (電子署名法)
- **電子署名に印鑑と同じ効力が持たされてしまう！**
  - 現状では、社会的に印鑑と同程度の認識／受容性が持てていないにも関わらず



電子署名及び認証業務に関する法律案

- 2000/4/14 国会提出、4/27 衆院審議可決、5/24 参院審議可決、5/31 公布 (法律番号: 102)
- 目的:
  - 電子署名に関し、電磁的記録の真正な成立の推定、特定認証業務に関する認定の制度その他必要な事項を定めることにより、電子署名の円滑な利用の確保による情報の電磁的方式による流通及び情報処理の促進を図り、もって国民生活の向上及び国民経済の健全な発展に寄与する (法案第一条)

電子署名及び認証業務に関する法律案 *contd.*

- 電磁的記録に対する電子署名に手書署名／印鑑押捺と同じ効力を認める
  - 署名者の本人性
  - 記録の真正性
- 電磁的記録について本人による電子署名が行われているときは、真正に成立したものと推定する
  - 民事訴訟法第228条第4項 相当
    - 「私文書は、本人又はその他の代理人の署名又は押印があるときは、真正に成立したものと推定」
  - 特定認証業務遂行認証業者発行の証明書に基づく電子署名には本人性の推定を付与
    - 運用における実印相当

PKI/電子署名の実社会へのimplication *contd.*

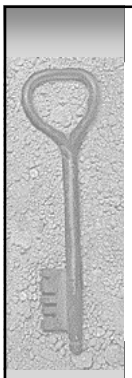
- たとえば、実社会での虚偽転問題
  - 本人が感知しないまま、勝手に住民登録を変更 (転出＆他所へ転入)
  - “本人”以外が行なうためには本人からの委任が必要だが、その“本人”の確認が問題
    - 「本人である」と言われてしまえば、確認は困難
    - 身分証明書による確認を取り入れた自治体もあるが…
      - そもそも、公的身分証明書が存在しない。代替品となる運転免許証などは、そもそも住民票に基づいている

PKI/電子署名の実社会への implication *contd.*

- とは言え、無茶苦茶頻発しているわけでもなさそう
- Real World で何が思い留めさせているか
  - 対面による手続き
    - 仕草／態度などへの影響
    - 記載内容をすらすら淀みなく口述できるか
- これがVirtual World になると？
  - 同様の抑制効果は期待薄
  - かなり危ないかも

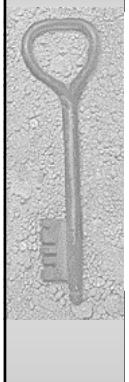
PKI/電子署名の実社会への implication *contd.*

- では何が必要か？
  - 安全性
    - 秘密鍵情報の盗用が行なえないように
      - 秘密鍵の利用＝本人性の推定
    - 実際的な知識の普及が急務
  - 使いやすさ
    - 非技術者層でも扱えるように
  - ➡ **秘密鍵をパスフレーズで保護してHDに保存するのでは、絶対的に不十分**
    - といった状況下で電子署名法／GPKI が動き出そうとしている
  - さらに、儀式性の確保といった問題も
    - Real world における署名／印鑑押捺と同等の



PKI/電子署名の実社会への implication *contd.*

- こんなものが欲しいなぁ
  - 電子印鑑
    - 署名機能をすべて含んだ自己充足型耐タンパ・デバイス (with private key activation by biometrics B-)
  - 仕様的には ***Java Ring*** 相当
    - ➡FIPS PUB 140-1 Level 3 耐タンパ性
      - 認証局署名鍵保護用暗号モジュール級
    - ➡1024bit RSA 署名≦1*sec*
    - ➡@ $65
  - 印鑑型デバイスで用紙型リセプタを撞くと、電子署名がなされる



結論(らしきもの)

- PKIは何でもこなす、ある意味、万能選手
  - 特に、否認防止機能
- ただし、もちろん、万全ではない
  - セキュリティとは、プロセスであり、一つの魔法のシステムで賄えるものではない
- Scalability/Flexibility などの点で、PKIを凌駕 (、もしくはそれに匹敵する)仕組は、他には見当たらない
  - 万一、このシステムがこけた場合、***Internet EC*** なる代物も終わり、もしくは大きく後退するだろう