暗号化/認証技術とその応用

稲村 雄

jane@cyborg.ne.jp /inamura@verisign.co.jp

日本ベリサイン (株) マーケティング部

- 自己紹介
- 暗号化技術概説
- 認証技術の発展
- X.509 証明書と公開鍵基盤
- *The Internet Security Protocol*

自己紹介

稲村 雄 (いなむら ゆう)

1960. 11. 生まれ

1986. 3. 某大学工学部物理工学科卒

1986. 4. NEC 情報システムズ (当時 日本電気技術情報システム開発) 入社

1986. 7. (財) 新世代コンピュータ技術開発機構 (ICOT) 出向

1993. 4. NEC 情報システムズ復帰

1998. 3. 日本ベリサイン移籍

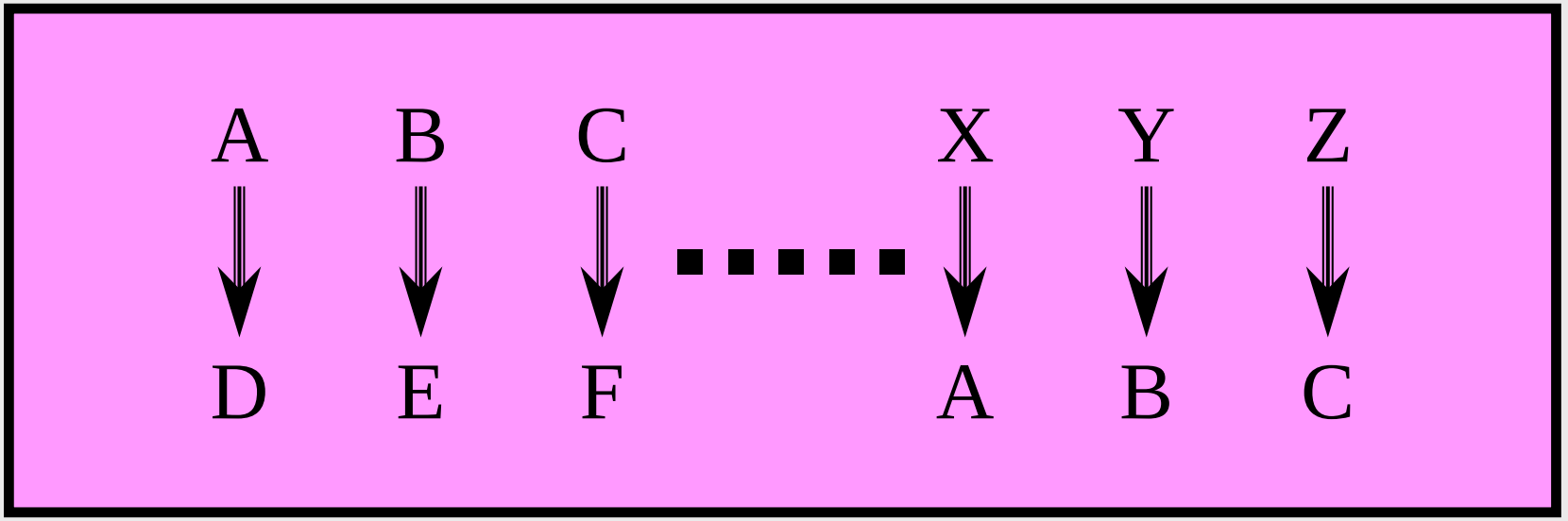
- 自己紹介
- 暗号化技術概説
- 認証技術の発展
- X.509 証明書と公開鍵基盤
- *The Internet Security Protocol*

- ユリウス･カエサル
  - 共和政ローマ末期の政治家/将軍/文学者 etc.
- カエサルが知人に宛てた手紙を託す使者を信頼できなかった時に、暗号通信の歴史は始まった。

Gaius Julius Caesar
(B.C. 100 - B.C. 44)

http://www.uni-paderborn.de/Admin/corona/chris/Caesar_0.html より

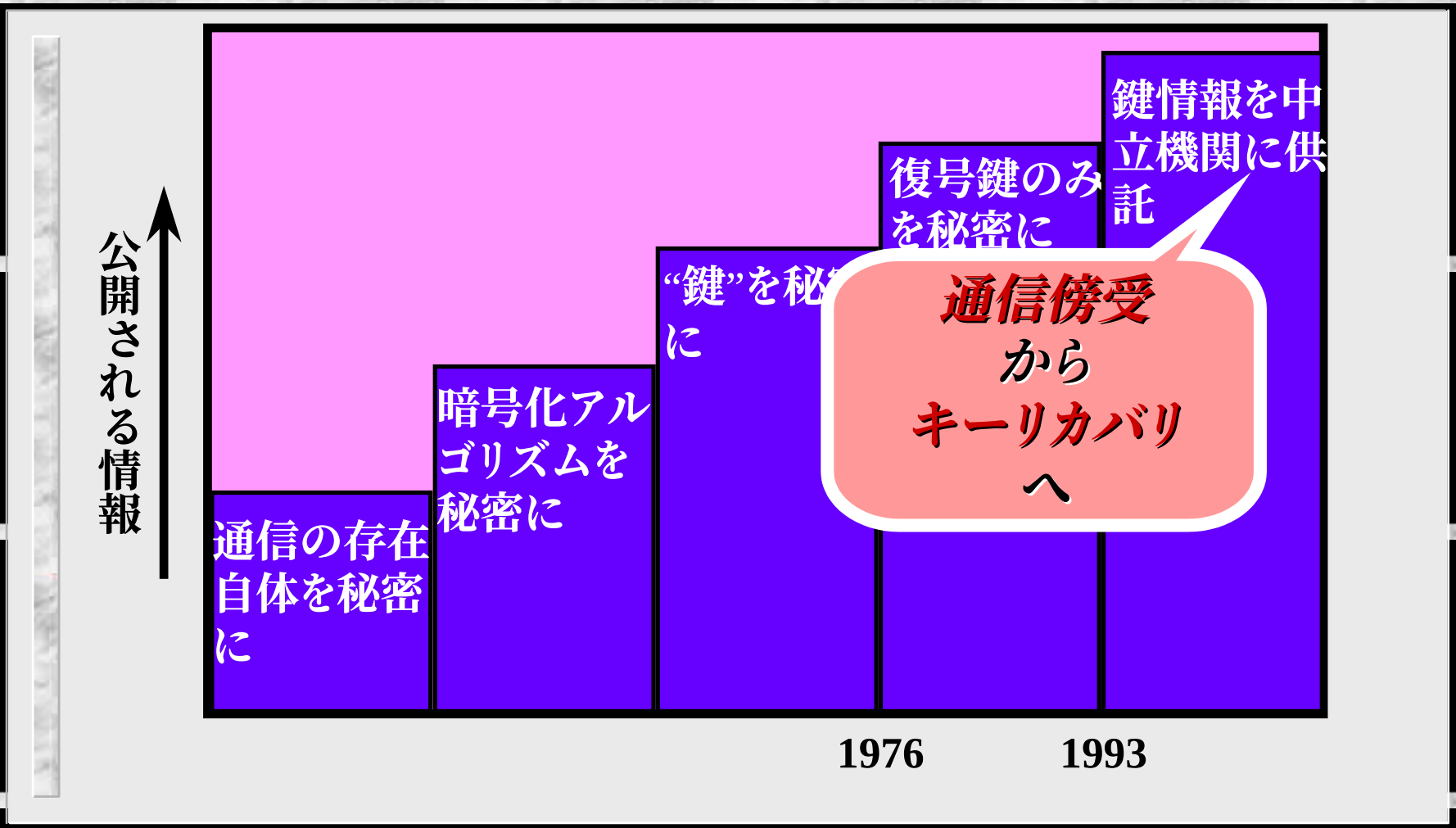
- 単純な換字式暗号
  - →現代ではほとんど実際の用に足りない
- NetNews などで利用例も



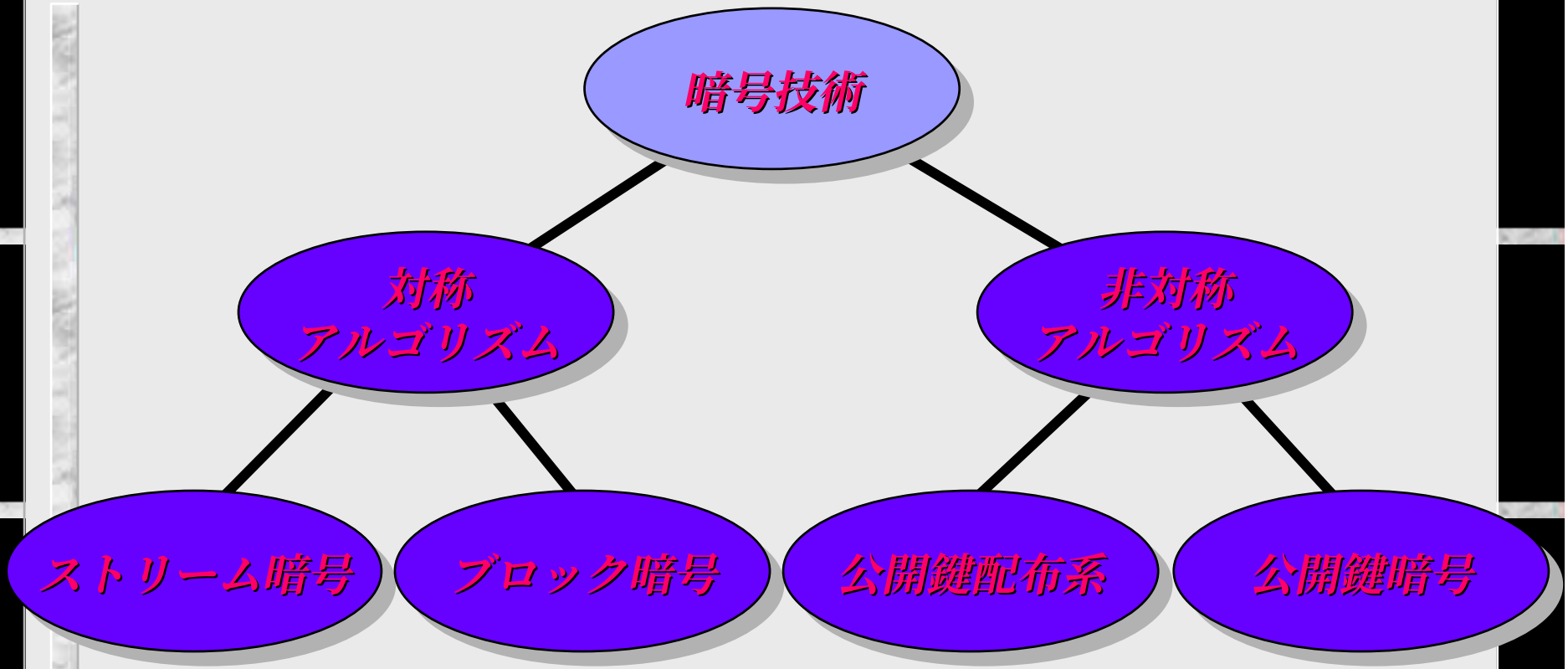
[illegible]

- 二千年に渡る難題
- インターネットは現代の代表例
  - 元来、研究者向けのネットワークなので、非常に性善説的
  - 広大すぎて、誰も把握しきれていない

暗号化技術概説



暗号化技術概説



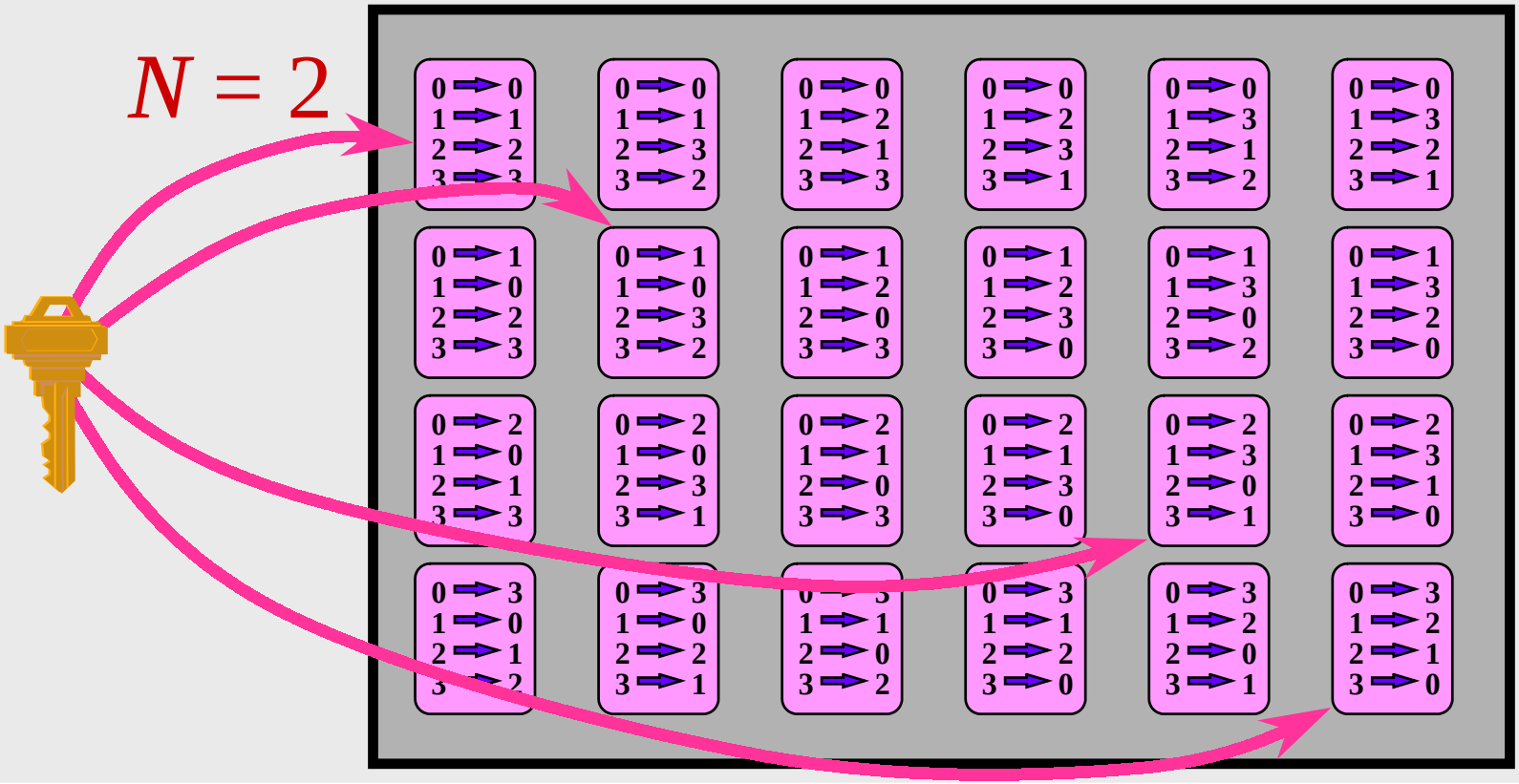
暗号化技術概説

- 通信文など (= 平文) を第三者には意味不明な形 (= 暗号文) に変換することで、当事者以外にとっての有用性を失わせしめるための技術
- 要は“可逆な”データ変換技術
  - ⇨ 平文空間が N ビットならば $2^N!$通り

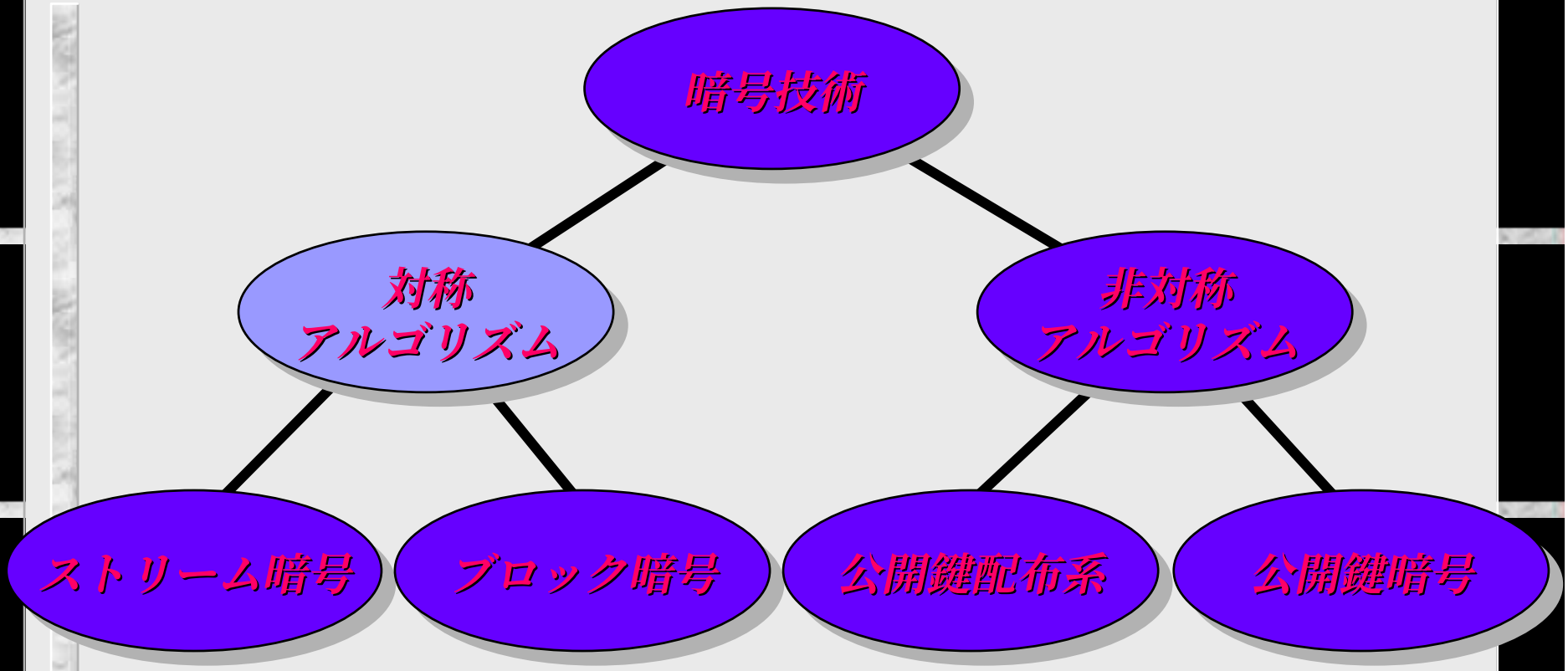
平文となり得るデータの集合

可逆な

- 平文空間が N ビットならば 2^N！通り

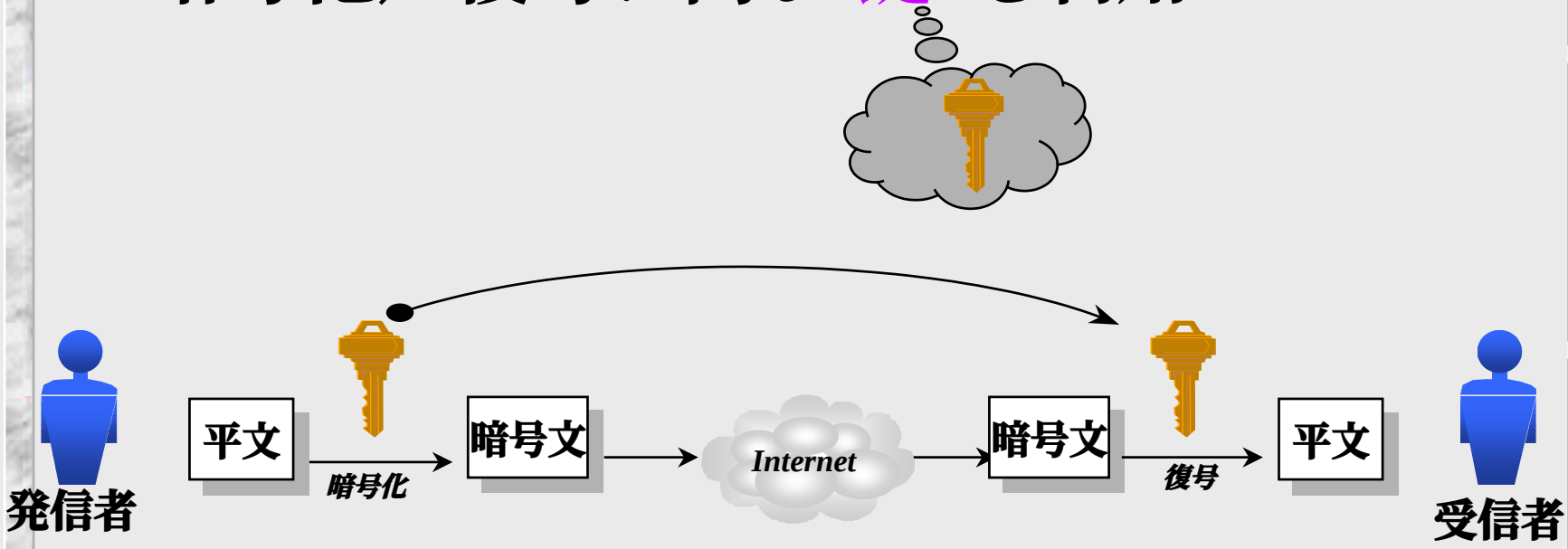


暗号化技術概説

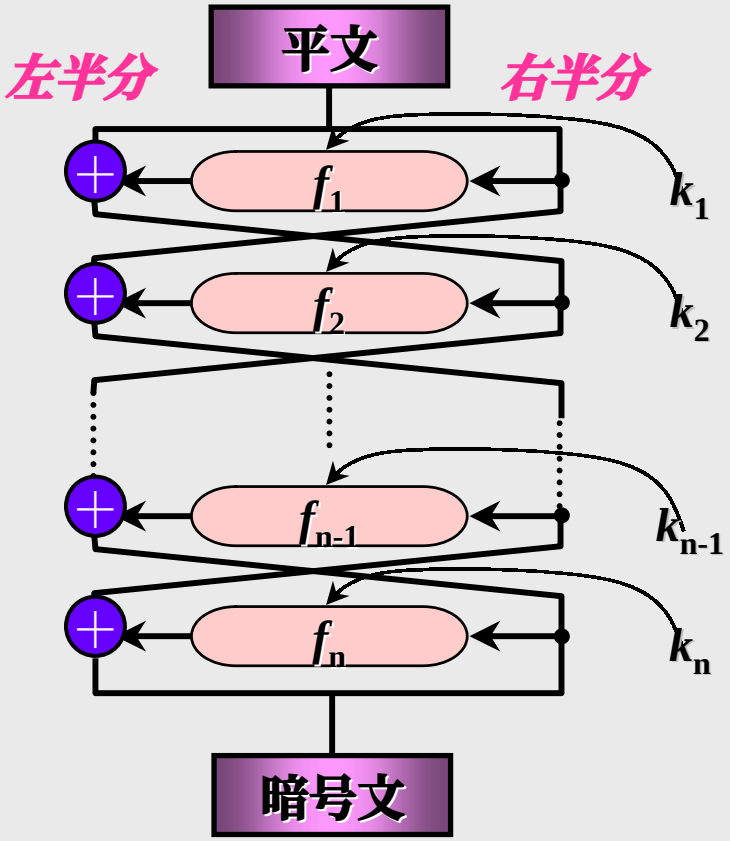


暗号化技術概説

- ＝秘密鍵暗号／共通鍵暗号／慣用暗号
- 暗号化／復号に同じ“鍵”を利用



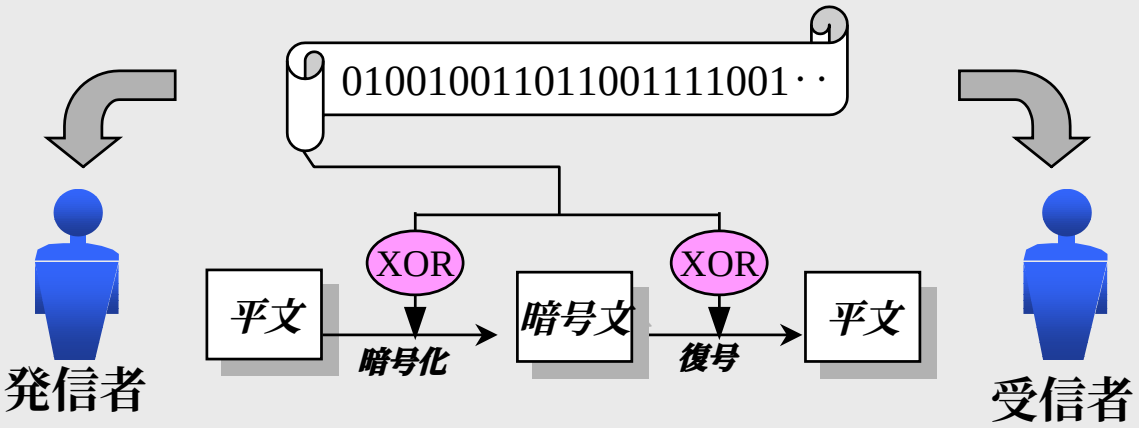
- *Confusion* (混乱) & *Diffusion* (拡散)
  - *Shannon* の情報理論に基づく概念
  - 置換操作による混乱と転置操作による拡散
  - SPN *(Substitution-Permutation Network)*
    - ➡ 効果的に *C-D* を実現



Feistel Network
(SPNの実現)

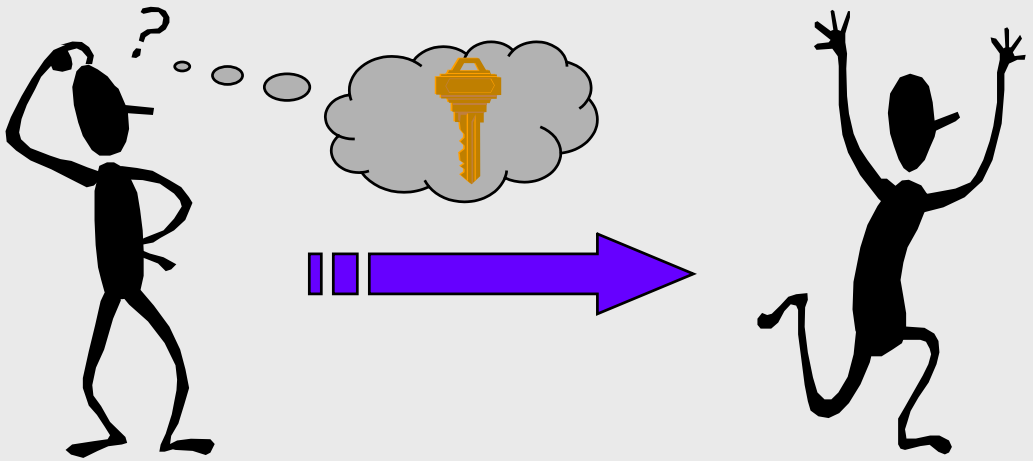
絶対に破られない暗号

- = *Onetime Pad*
- メッセージと等長の乱数鍵を利用
- 数学的に安全性が証明されている暗号方式

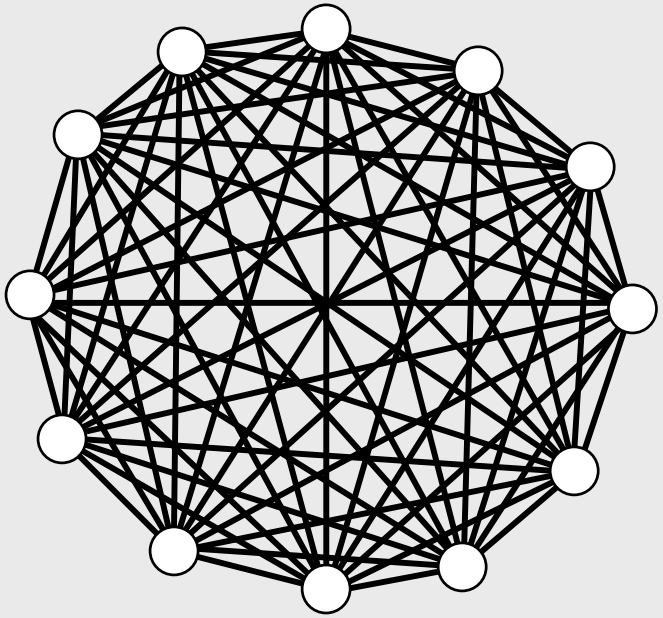


暗号化技術概説

- 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
- 難点 1: 鍵の安全な共有方法が大問題

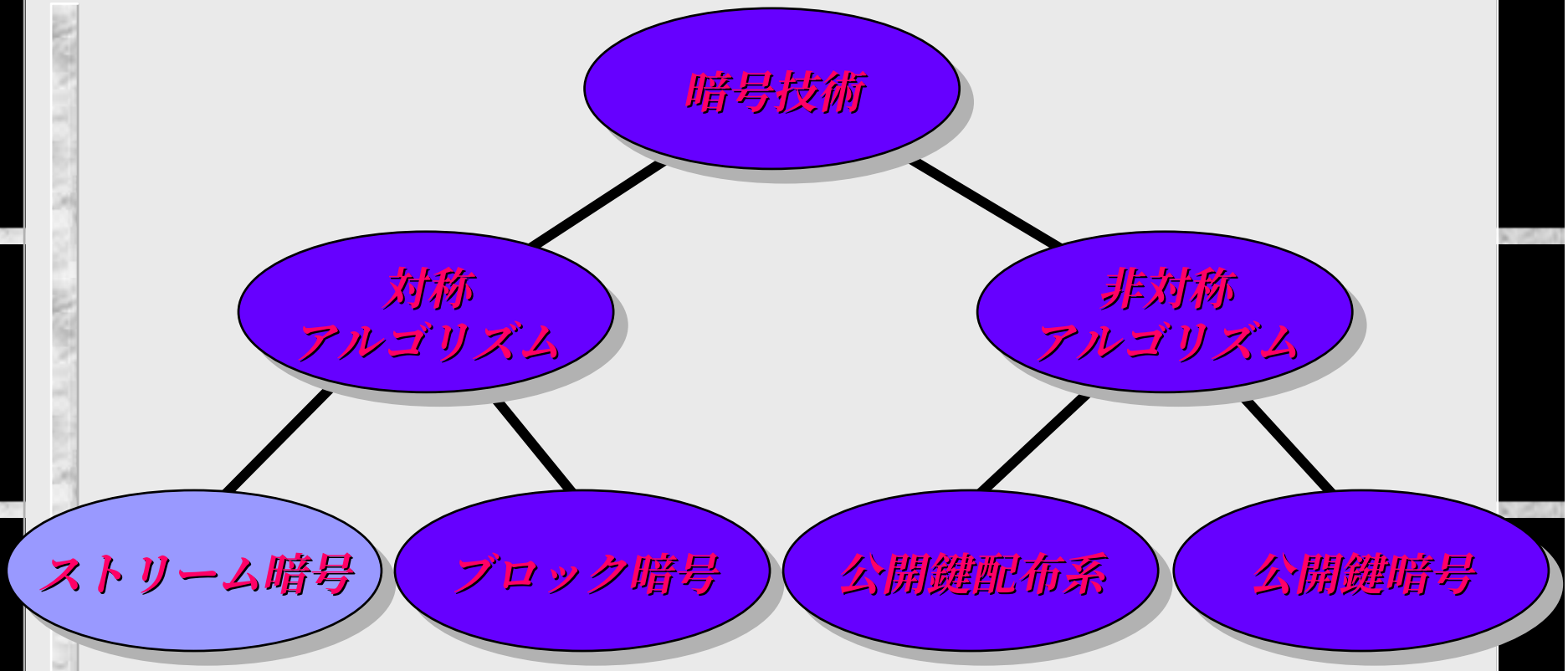


- **難点 2: 相通信するペア毎に異なる鍵が必要**
  - ⇒ **必要な鍵数の爆発**

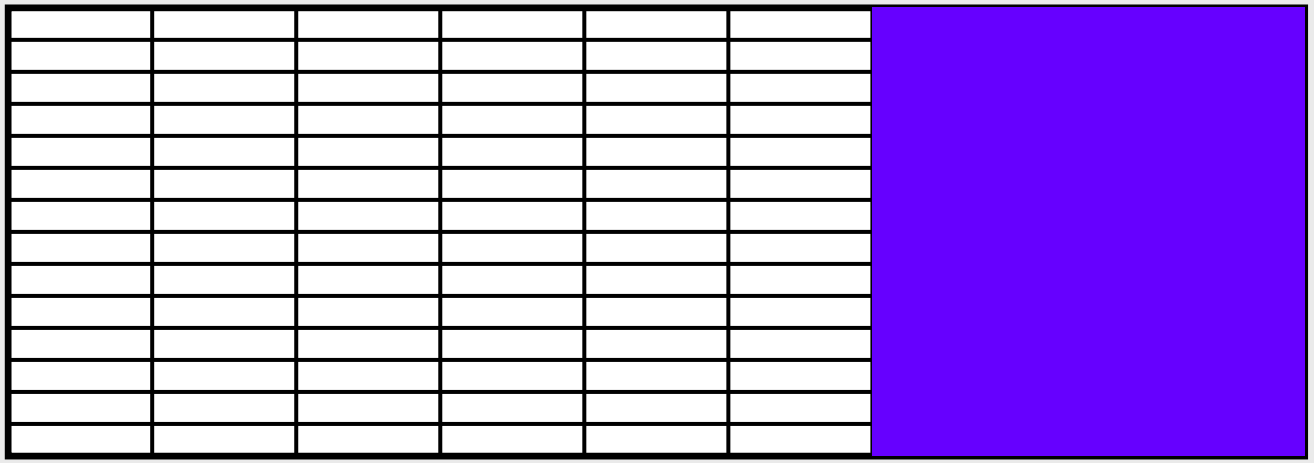


$$\frac{N(N-1)}{2}$$

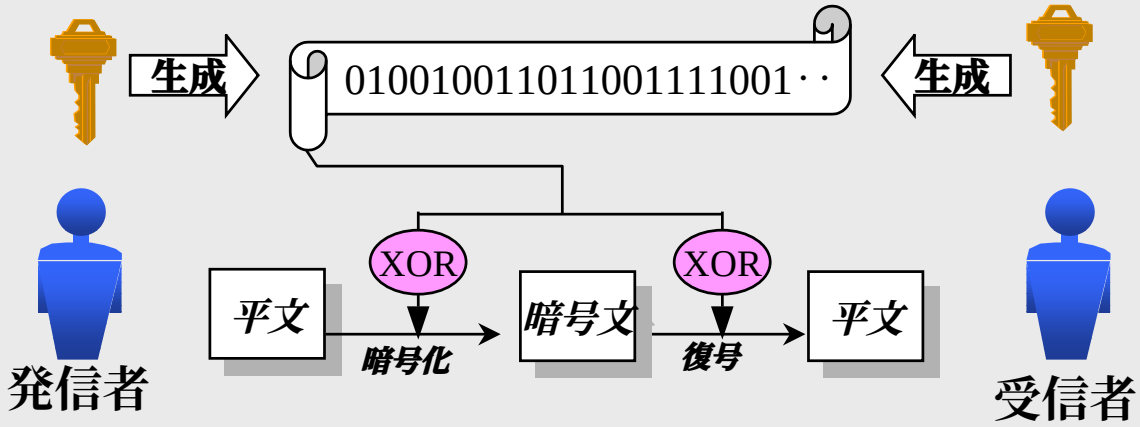
暗号化技術概説



- 平文／暗号文を頭から順番に処理
  - 処理単位は bit/byte/word など、いろいろ



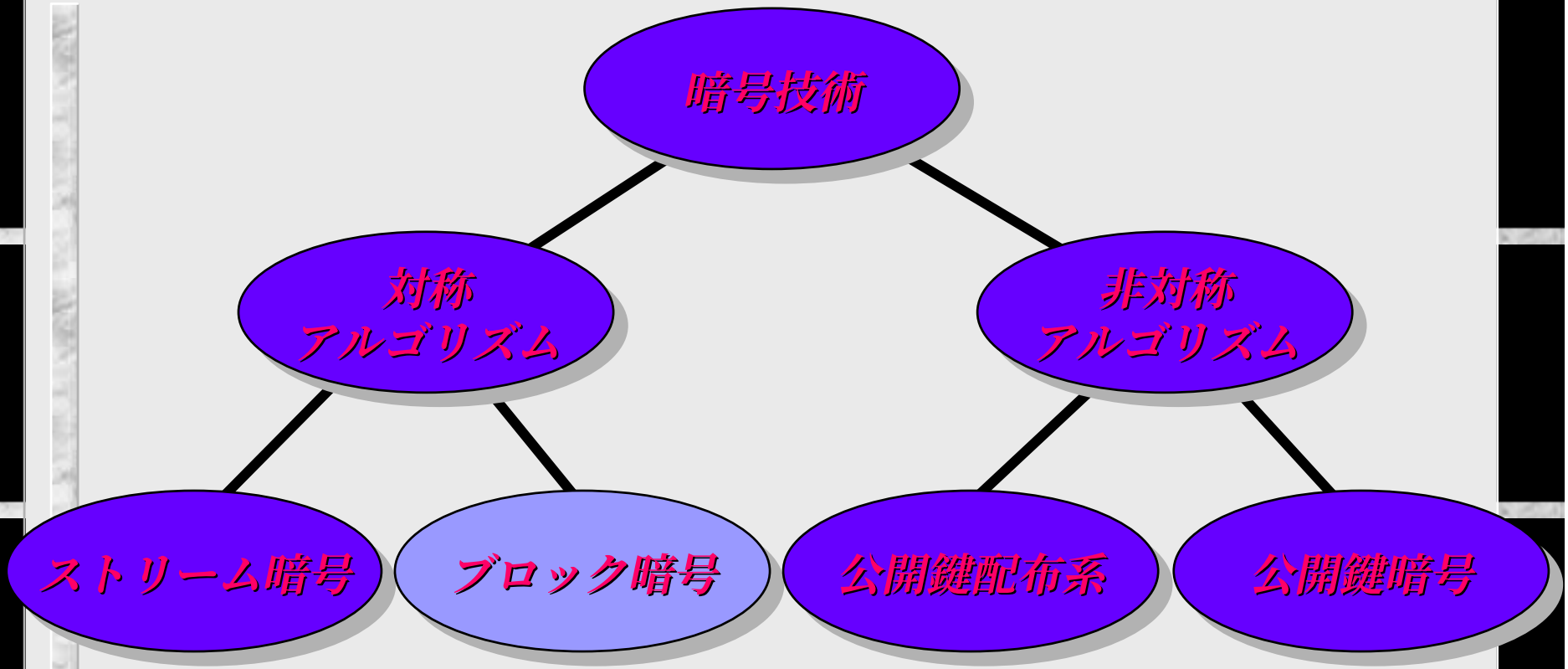
- 送信者／受信者ともに、鍵から擬似乱数列を生成
  - *Onetime Pad* の簡易版



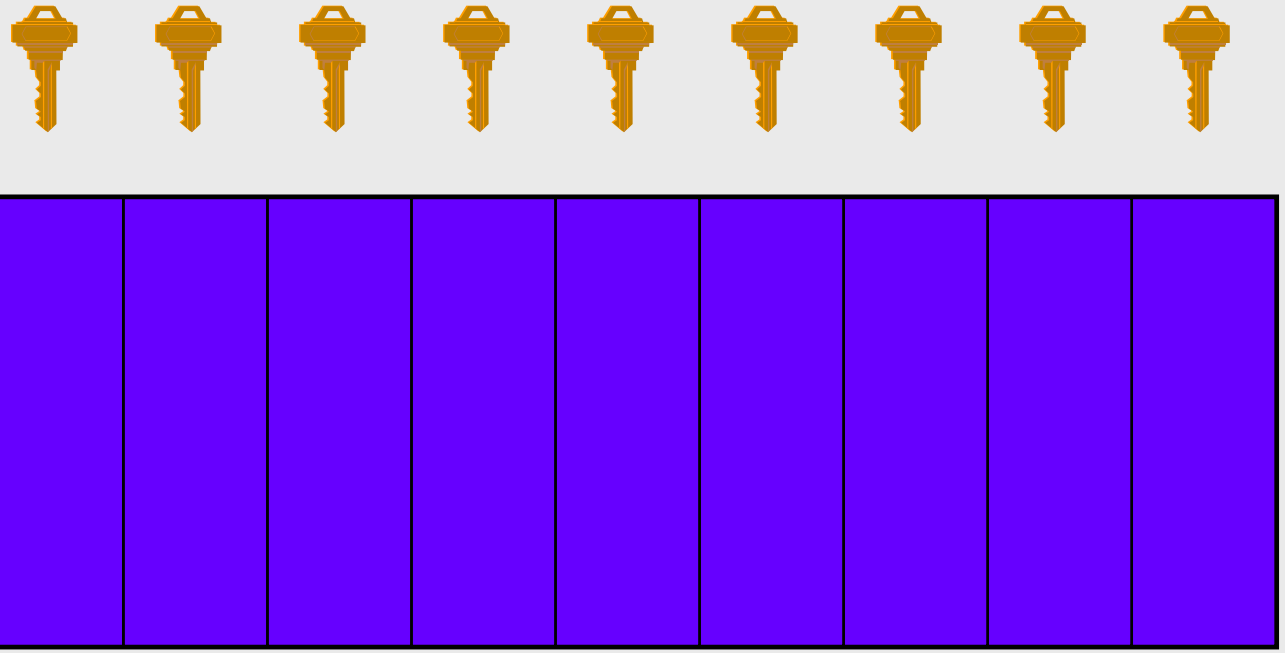
暗号化技術概説

- RC4
  - *Ron Rivest (RSA の R)* が開発
  - 可変長鍵の利用が可能
  - SSL *(Secure Sockets Layer)* でのデフォルト
  - 本来は非公開アルゴリズムだったが…
- SEAL
- WAKE

暗号化技術概説



- 平文／暗号文を一定サイズのブロックに分割して処理



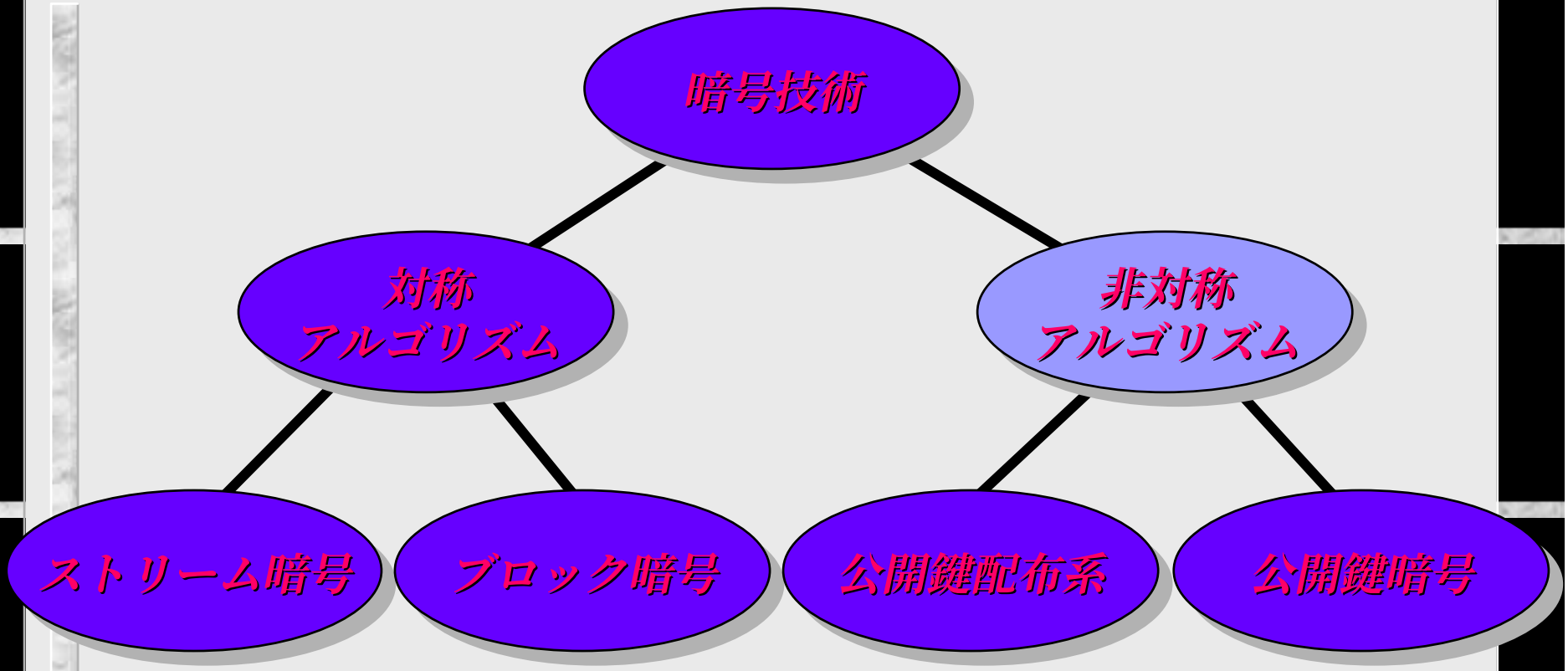
- ***DES*** ***(Data Encryption Standard)***
  - 米国 NBS (現 NIST) の公募に応じた IBM 提案に基づく暗号化アルゴリズム
  - 1976 年以来、標準的な暗号方式として利用が進む
  - 鍵長の短さのため、そろそろ寿命が尽きつつある
    - RSA 社による DES Challenge
    - AES ***(Advanced Encryption Standard)*** の公募開始

- ***AES*** ***(Advanced Encryption Standard)***
  - DES に代わるアルゴリズムとして米国 NIST が 1997 年 9 月から公募を開始。
  - ***http://csrc.nist.gov/encryption/aes/aes_home.htm***
  - 最低条件：
    - 対称鍵暗号
    - ブロック暗号
    - 鍵-ブロック長として128-128, 192-128, 256-128 の組み合わせをサポート

暗号化技術概説

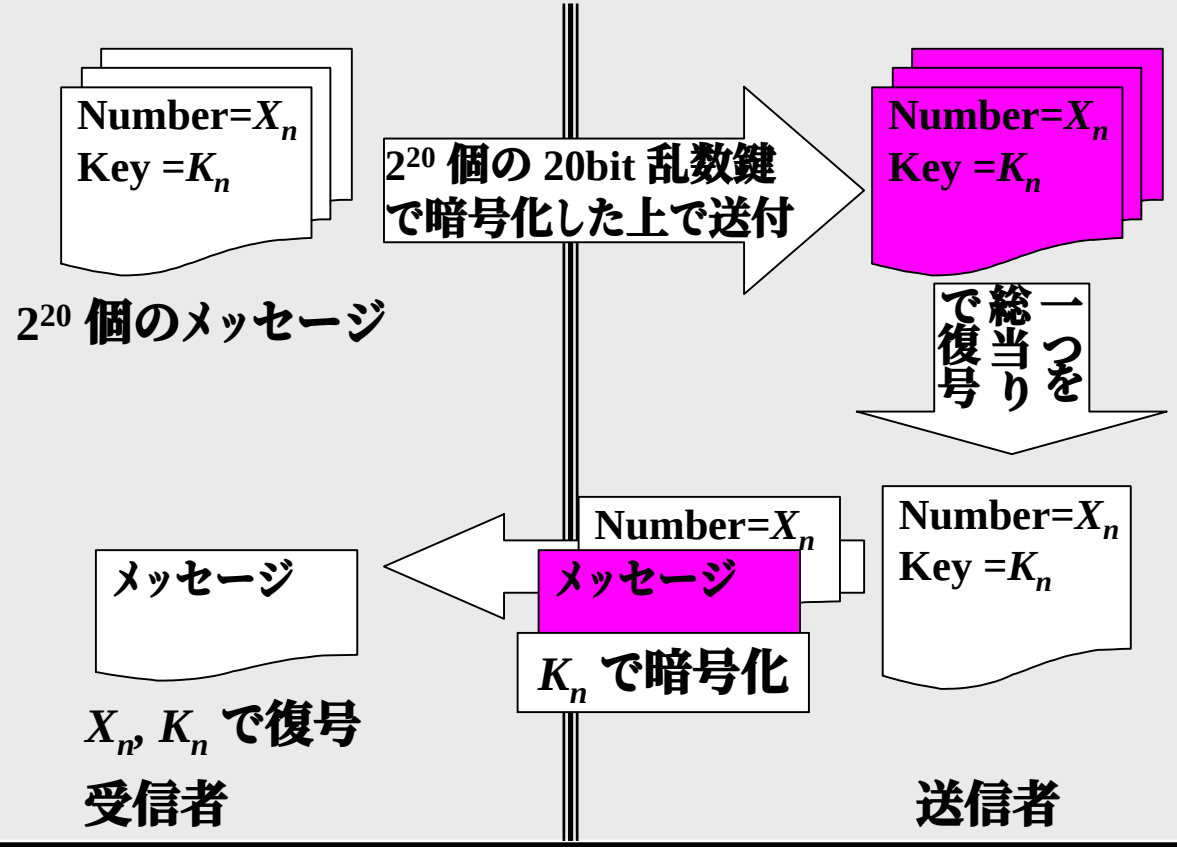
- *CAST-256*
- *CRYPTON*
- *DEAL*
- *DFC*
- *E2*
- *FROG*
- *HPC*
- *LOKI97*
- *MAGENTA*
- *MARS*
- *RC6*
- *RIJNDAEL*
- *SAFER+*
- *SERPENT*
- *TWOFISH*

暗号化技術概説

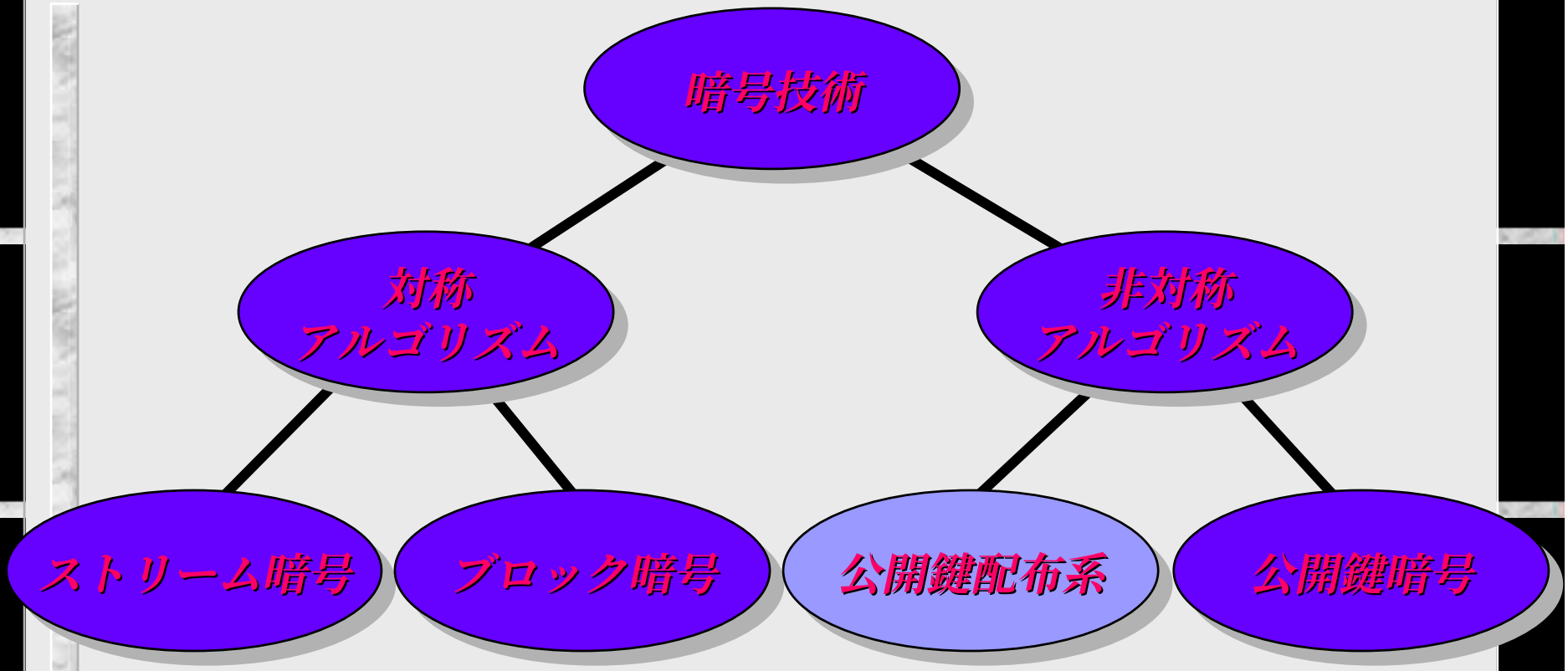


- 1976 年に *W.Diffie* と *M.Hellman* が考案
  - パブリックには。
  - 1966 年 (*NSA*)、1970 年 (*CESG*) といった主張がある
- 基本は落とし戸 (*Trapdoor*) 付一方向関数
  - 片方向への計算は誰にでもできるが、逆方向の計算は非常に困難
  - 特別な知識 (= *Trapdoor*) を持った者は逆方向の計算ができる

R. Markle の提案 (1974)

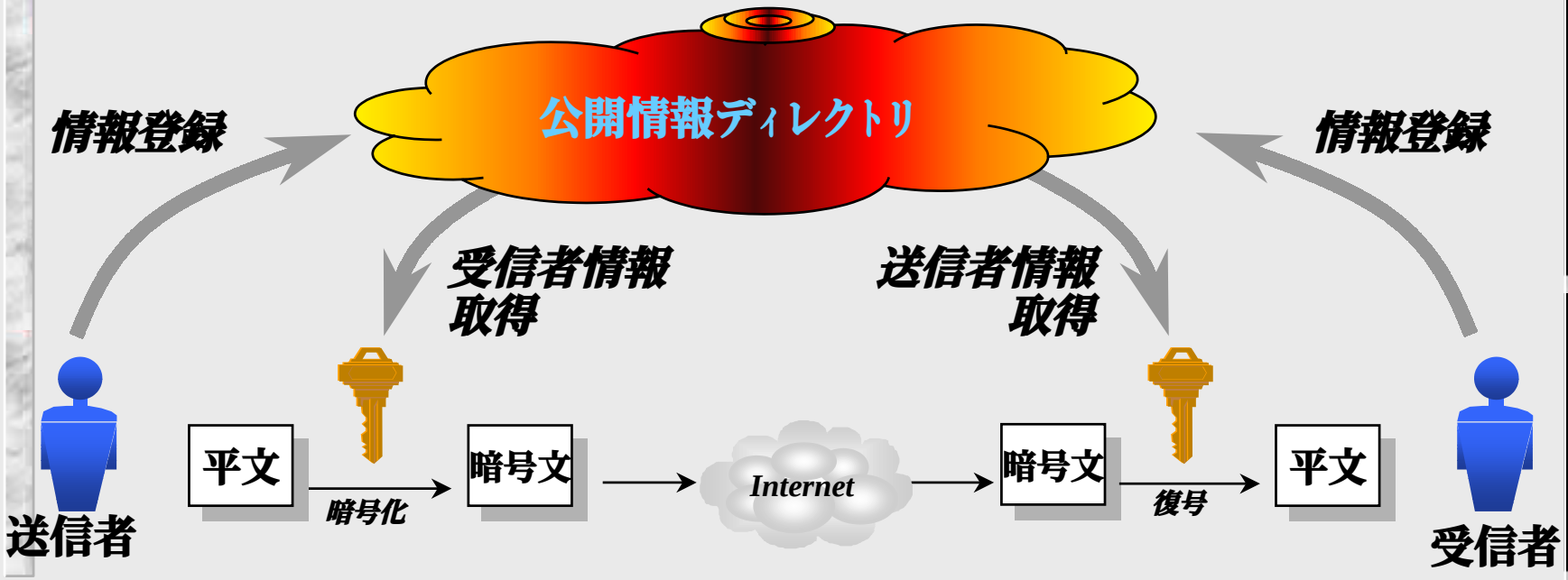


暗号化技術概説



■ 対称暗号の補助的役割

» 各ユーザが公開している情報から対称暗号方式で用いられる共通鍵を生成する



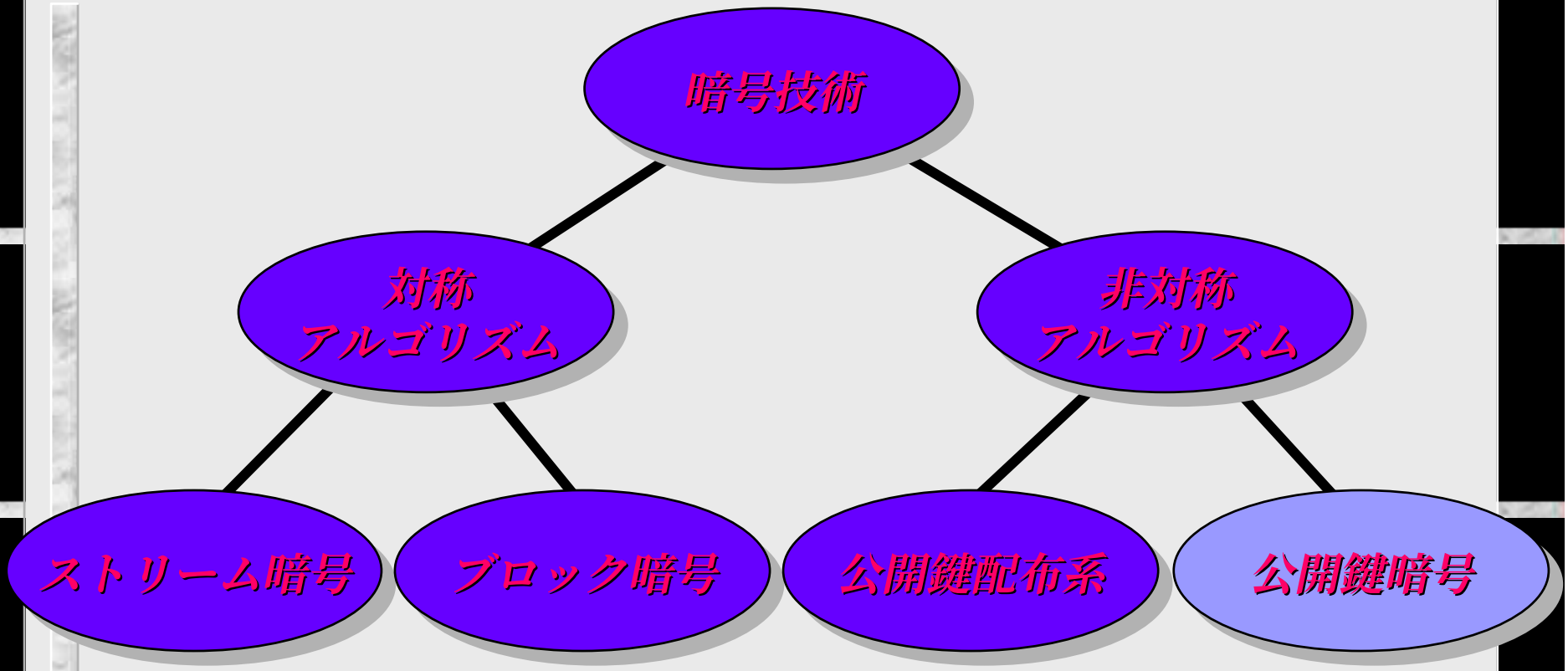
Diffie-Hellman 方式

- *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：

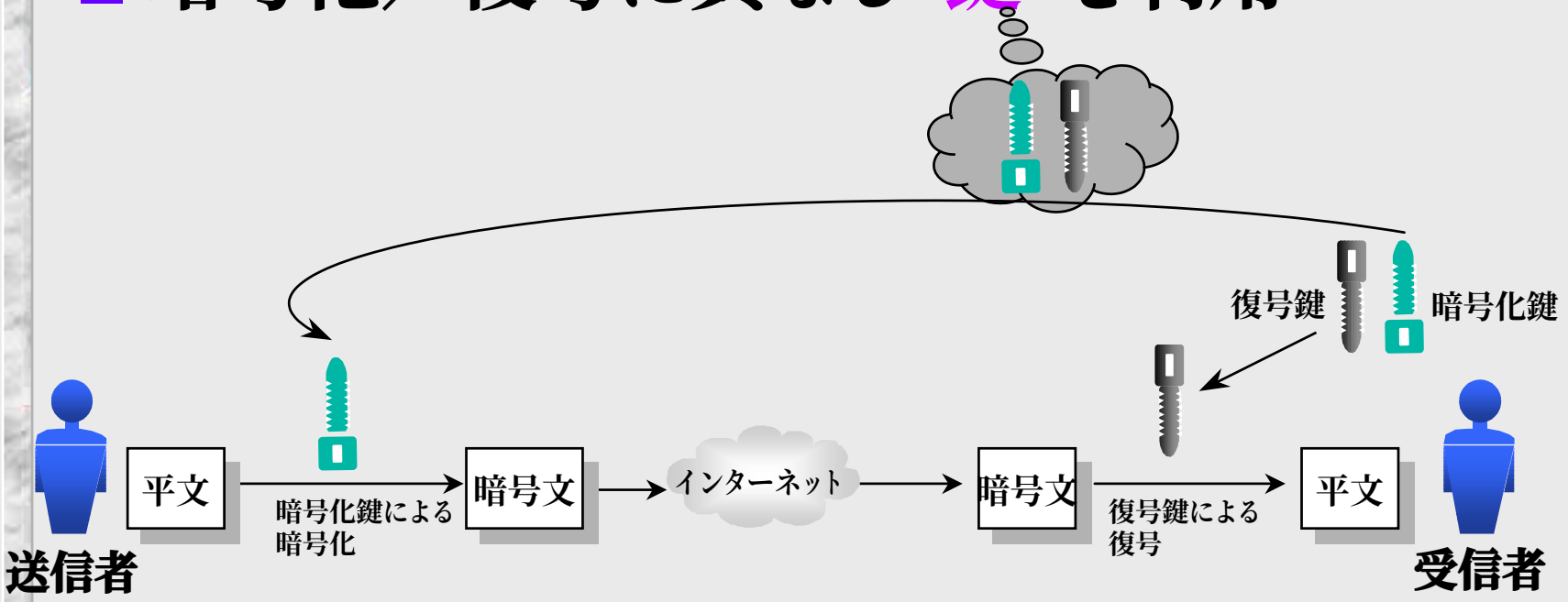
α, q, y が既知整数のとき、$y = \alpha^{x} \pmod{q}$ を満たすような整数 x を見付ける

暗号化技術概説



暗号化技術概説

- *W.Diffie* と *M.Hellman* のアイデア (1976)
- 暗号化／復号に異なる“鍵”を利用



暗号化技術概説

- 本システムのユーザは数学的に特殊な関係にある二種類の“鍵”を生成
  - 一方の鍵で暗号化されたデータは他方の鍵でしか復号できない
  - 一方の鍵データのみから他方の鍵を導き出すことは著しく困難
- 一つの鍵 (暗号化用鍵) を公開し、残り (復号用鍵) を秘密に保持する

暗号化技術概説

- 対称暗号との比較
- 難点 1: 鍵の安全な共有方法が大問題
  - 暗号化用の鍵は公開してしまえるので問題なし
- 難点 2: 相通信するペア毎に異なる鍵が必要
  - すべての送信者に対して一つの公開鍵／秘密鍵ペアのみで済む

インターネット環境で利用するには必須の技術

暗号化技術概説

- *R. Rivest, A. Shamir, L. Adelman* の三人が発明した暗号化アルゴリズム (1978 年)
- 大きな数の素因数分解の困難性に基づく
- 非対称暗号方式のデファクトスタンダード

- *N.Koblitz* と *V.S.Miller* の二人が独立に考案 (1985 年)
- 新しい方式ではなく、*Diffie-Hellman* などの既存システムを楕円曲線上に実装したものが主
- 他の非対称アルゴリズムと比較して短い鍵長で安全性を確保できるということで注目される

実際面

- ボトムアップ･アプローチ
  - アルゴリズム的欠陥がない (*brute-force attack* 以上に有効な攻撃法がない) と仮定
  - ***DES*** (鍵長 56bit) を破るのに 56 時間 (現レコード)
  - 1 秒で破れるようになったとして、鍵長 128bit では

$$2^{128-56}\ \mathrm{sec} = 2^{72}\ \mathrm{sec}$$

$$\cong 1.5\times 10^{14}\ \mathit{year}$$

$$\gg 5\times 10^{9}\ \mathit{year}\ (\text{太陽の余命})$$

1995 年時点での推測値

$100K	35 時間	10^{19} 年
$1M	3.5 時間	10^{18} 年
$10M	21 分	10^{17} 年
$100M	2 分	10^{16} 年
$1G	13 秒	10^{15} 年
$10G	1 秒	10^{14} 年
$100G	0.1 秒	10^{13} 年

Bruce Schneier, Applied Cryptography 2nd Ed. John Wiley & Sons, Inc. より

- 非対称暗号の場合
  - 鍵空間全体に対する *brute-force attack* は非現実的
  - 数学的攻撃 (素因数分解、離散対数計算等)
- 強度がほぼ等しくなるための鍵長

対称暗号鍵長 (bit)	56	64	80	112	128
非対称暗号鍵長(bit)	384	512	768	1792	2304

Bruce Schneier, Applied Cryptography 2nd Ed. John Wiley & Sons, Inc. より

- トップダウン・アプローチ
  - 相対論的限界 (光速)
  - 量子力学的限界 (不確定性原理)
  - 熱力学的限界:
    - 単なるカウンタ操作にもエネルギが必要

1 bit 操作に必要なエネルギ:

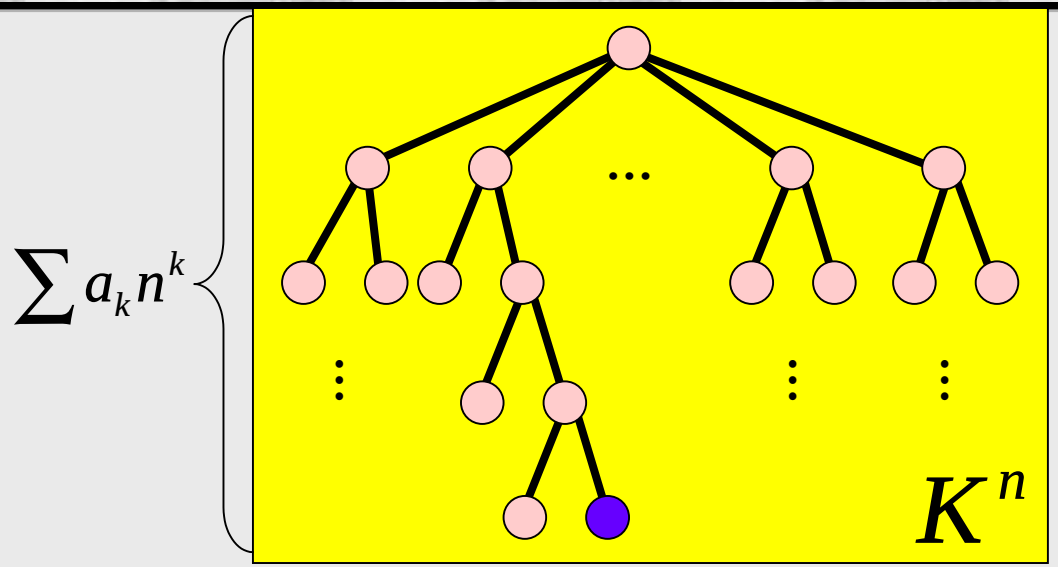
$$kT = 1.38\times10^{-23}\times3.2 \cong 4.42\times10^{-23}(J)$$

太陽の年間放出エネルギ($1.21\times10^{34}(J)$)で:

$$1.21\times10^{34} \div 4.42\times10^{-23} \cong 2.71\times10^{56}$$

$$\approx 187bit$$

Bruce Schneier, Applied Cryptography 2nd Ed. John Wiley & Sons, Inc. より



- *Onetime Pad* より弱いアルゴリズムの終末
- 量子コンピュータ (　　非決定的チューリングマシン) などというものが実現したら、$P \neq NP$ でも壊滅かも

暗号化技術概説

- ＝デジタル署名
- 内容が改竄されていないことを保証するために、作成者がデータに付加する“印”
  - » 実世界における署名／印鑑押捺などに相当
- 通常は、メッセージダイジェストと非対称暗号技術の併用によって実現

暗号化技術概説

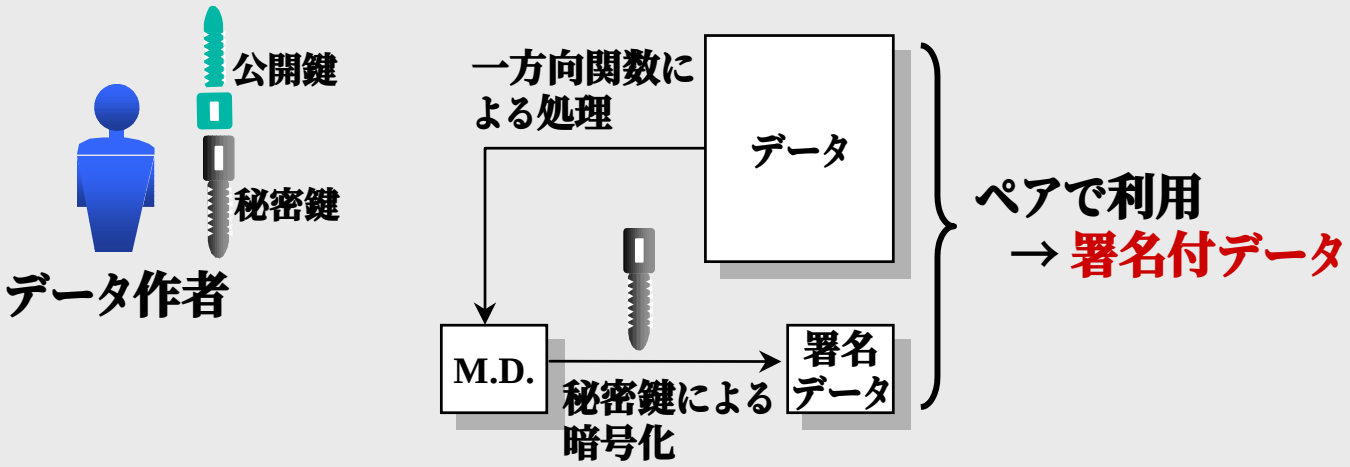
- 任意のデータから、そのデータ特有とみなせる短い (百数十ビット程度) 情報 (= メッセージダイジェスト) を抽出する技術
- 暗号学的一方向ハッシュ関数を利用
  - 逆演算不可
  - *Collision Proof* 性
- メッセージダイジェストは、元データの“指紋”として扱うことが可能

- 暗号学的一方向ハッシュ関数が持つべき特性
  - 元データが 1bit 異なっただけで、メッセージダイジェスト中の多くの(半数程度) bit に影響
  - あるメッセージダイジェストに対応する元データを見つけ出すのが非常に困難
  - 同じメッセージダイジェストを得られるような二種類のメッセージを見付けるのが非常に困難

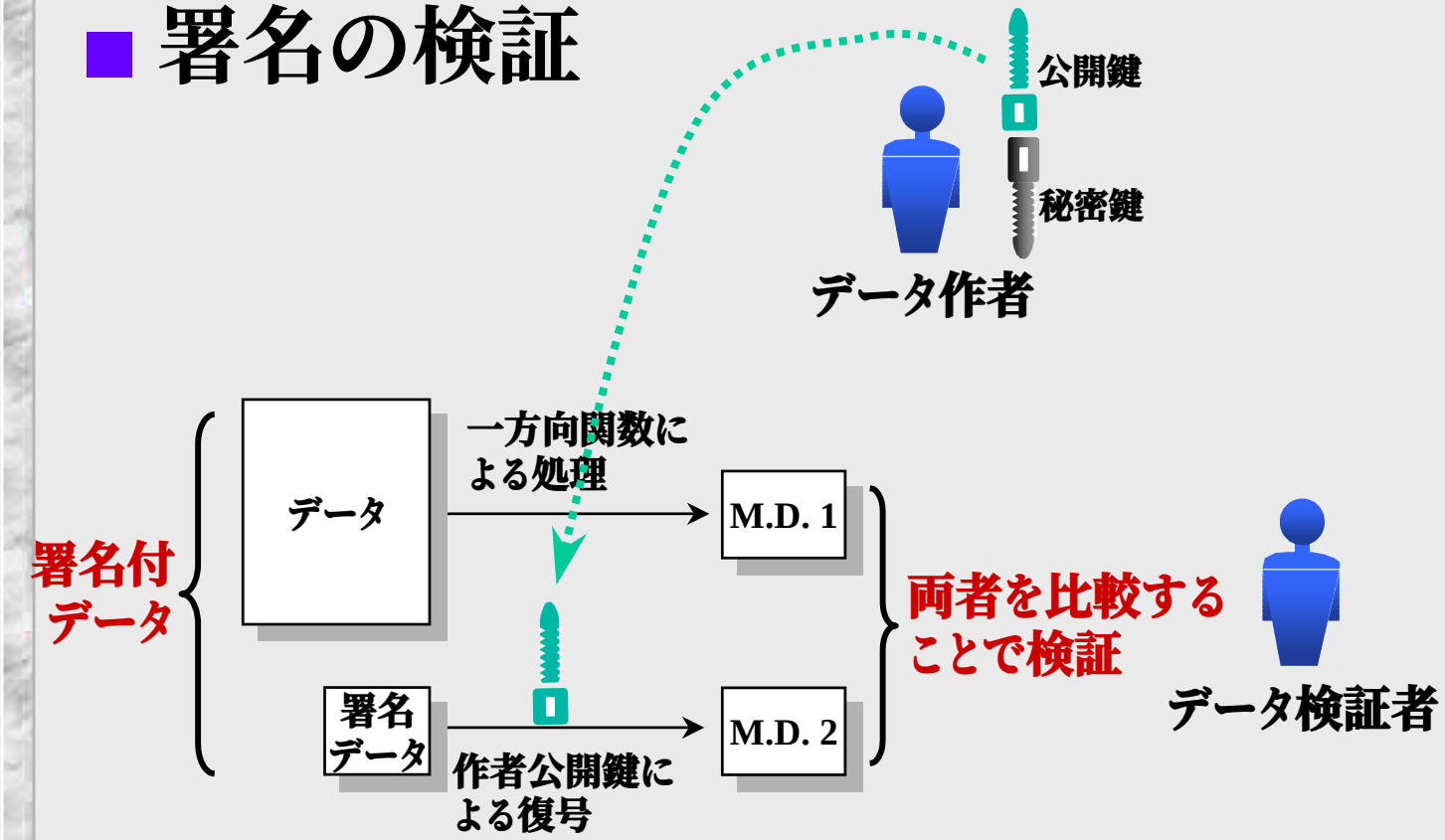
Birthday Attack
同じ誕生日の人間を見付けるのに何人必要か

- 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
- 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

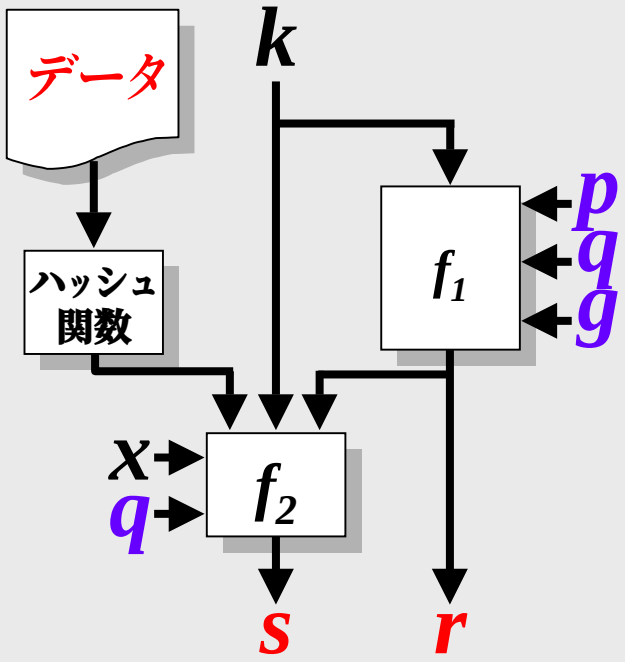
- メッセージダイジェストと非対称暗号技術の併用による実現
  - メッセージダイジェストの *Collision Proof* 性と暗号化で署名データの一意性を保証



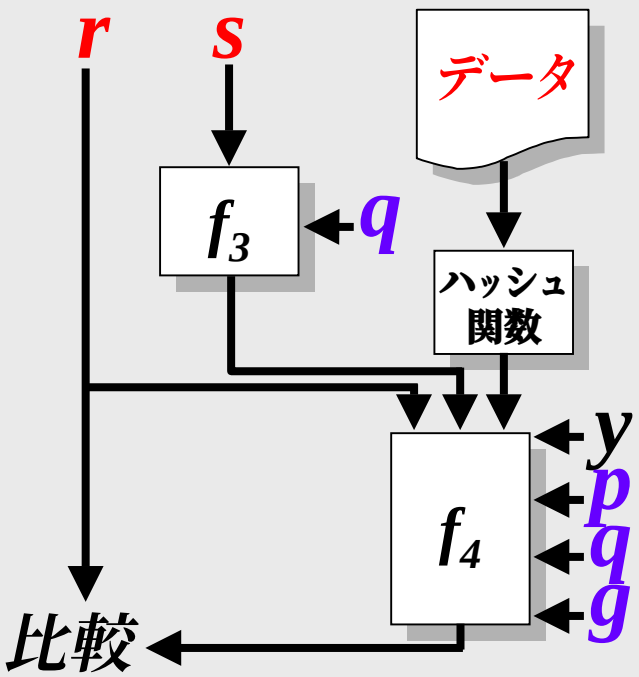
■ 署名の検証



■ DSA 風アプローチ



署名処理



検証処理

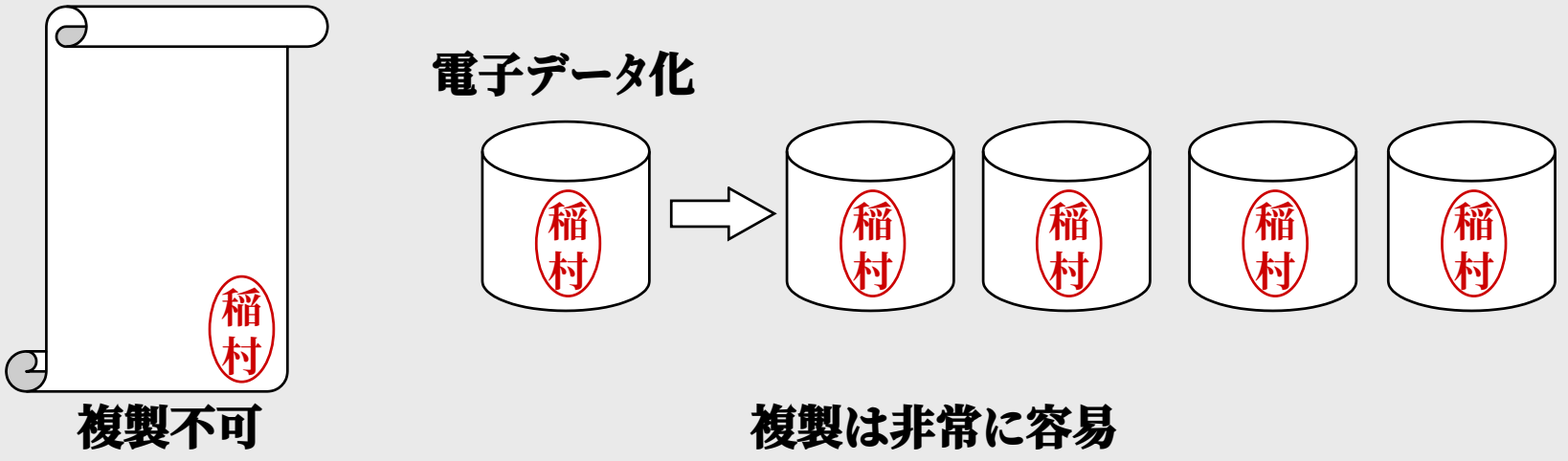
- 自己紹介
- 暗号化技術概説
- 認証技術の発展
- X.509 証明書と公開鍵基盤
- *The Internet Security Protocol*

電子認証技術

- 誰が誰であるのかを、確実に判断する必要性
  - ネットの向こうにいるのは誰？
    - 電子認証技術
- 基本は実世界での認証と同じ
  - 物理的特徴の確認
  - 所有物の確認
    - あらかじめ登録しておいたデータとの照合処理
- ただし、電子データ固有の事情も

■ 電子データ固有の事情

» 電子データは複製が容易なため、単純に実世界の仕組みを持ち込んでも機能しない。



■ 認証手段 (ECOM

» 本人に特有の

➠ 指紋

➠ 声紋

» 所持品による：

➠ クレジットカード、運転免許証

» 秘密情報による：

➠ パスワード、暗証番号

➠ デジタル署名、公開鍵証明書 (デジタル ID)

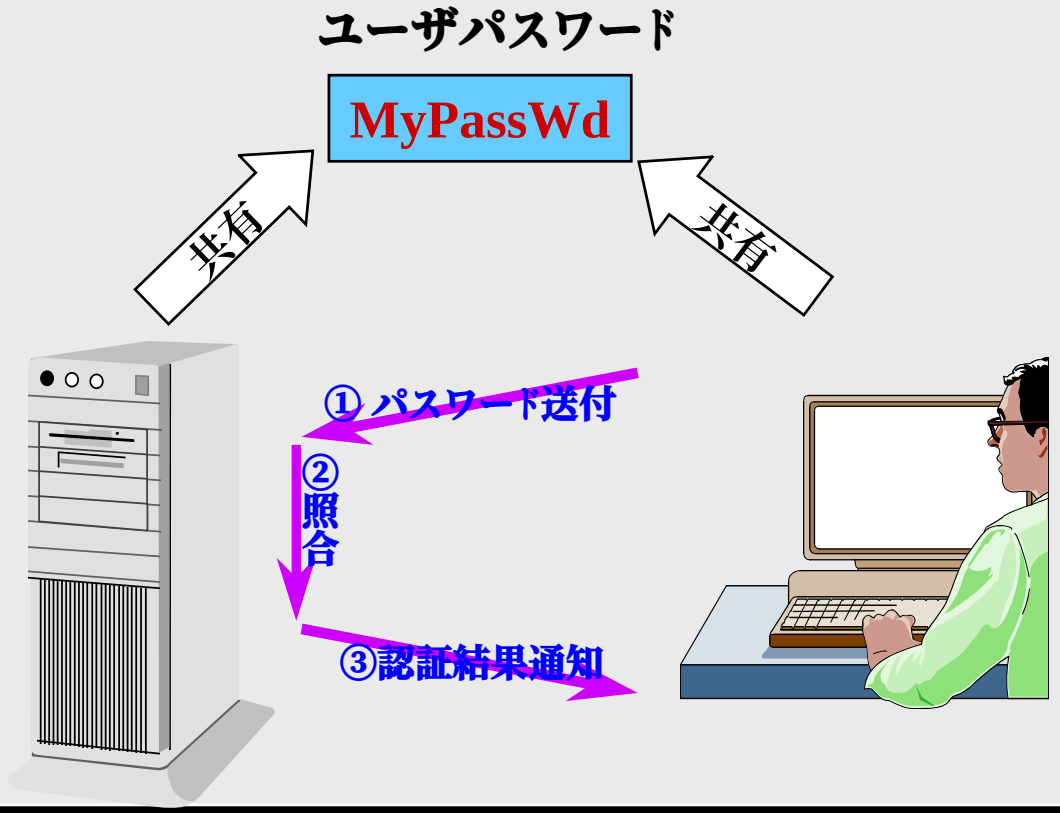
ネットワーク経由の
認証には不向き
入退室管理など特
定用途向け

ネットワーク経由の
認証には不向き
入退室管理など特
定用途向け

電子認証技術

- パスワード認証
  - 単純パスワード
  - 暗号化パスワード
- チャレンジ & レスポンス
- ワンタイムパスワード
- *Kerberos*
- 公開鍵証明書

■ 単純パスワード方式



■ 暗号化パスワード方式

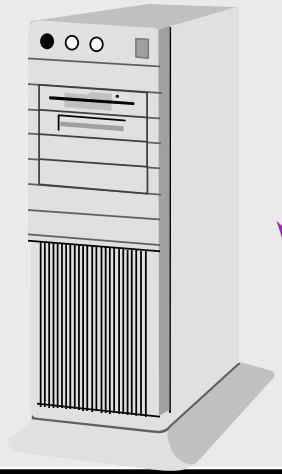
システムデータベース上

Aj6OlwJmA6czE

暗号化して保存

ユーザパスワード

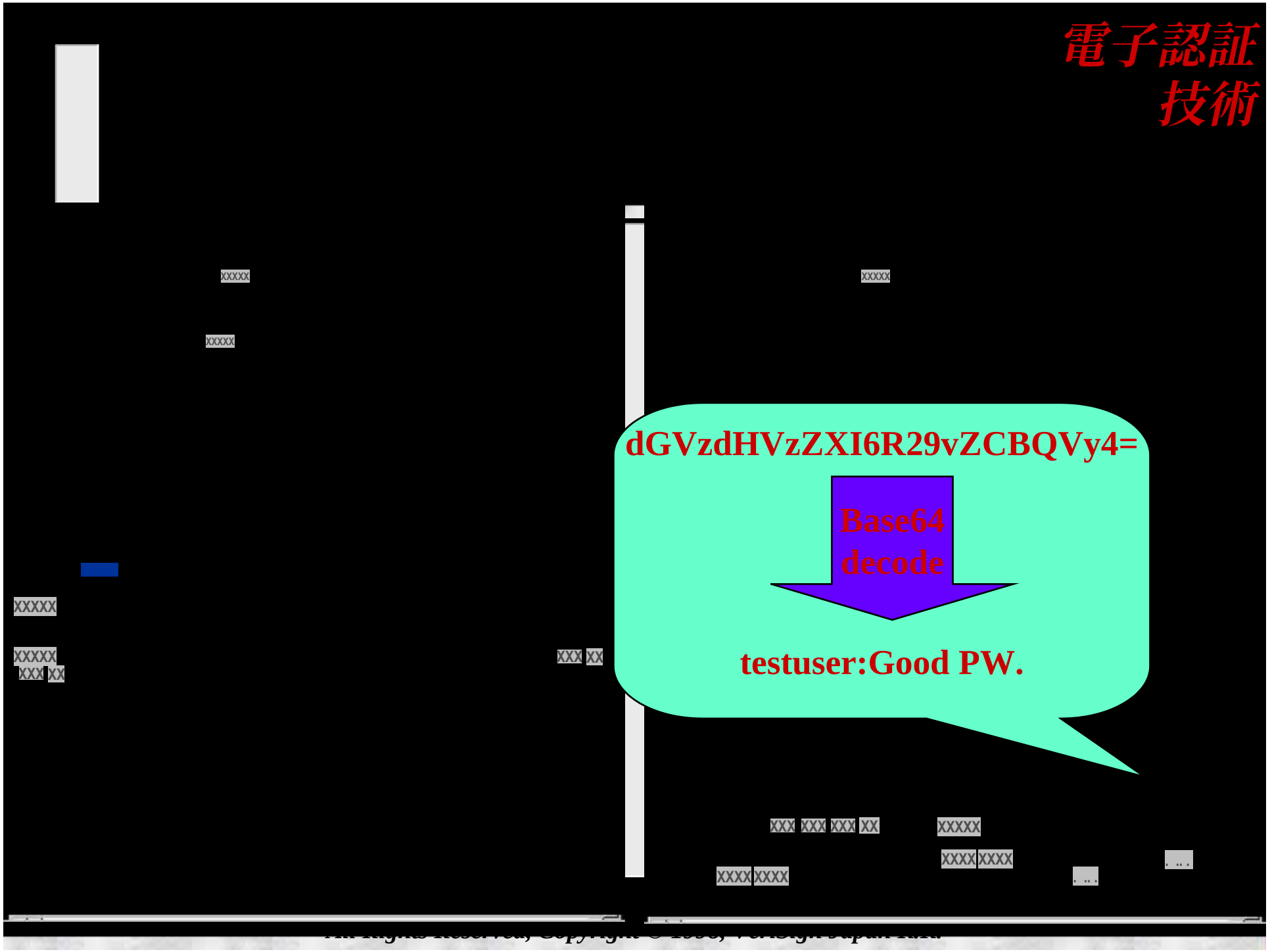
MyPassWd



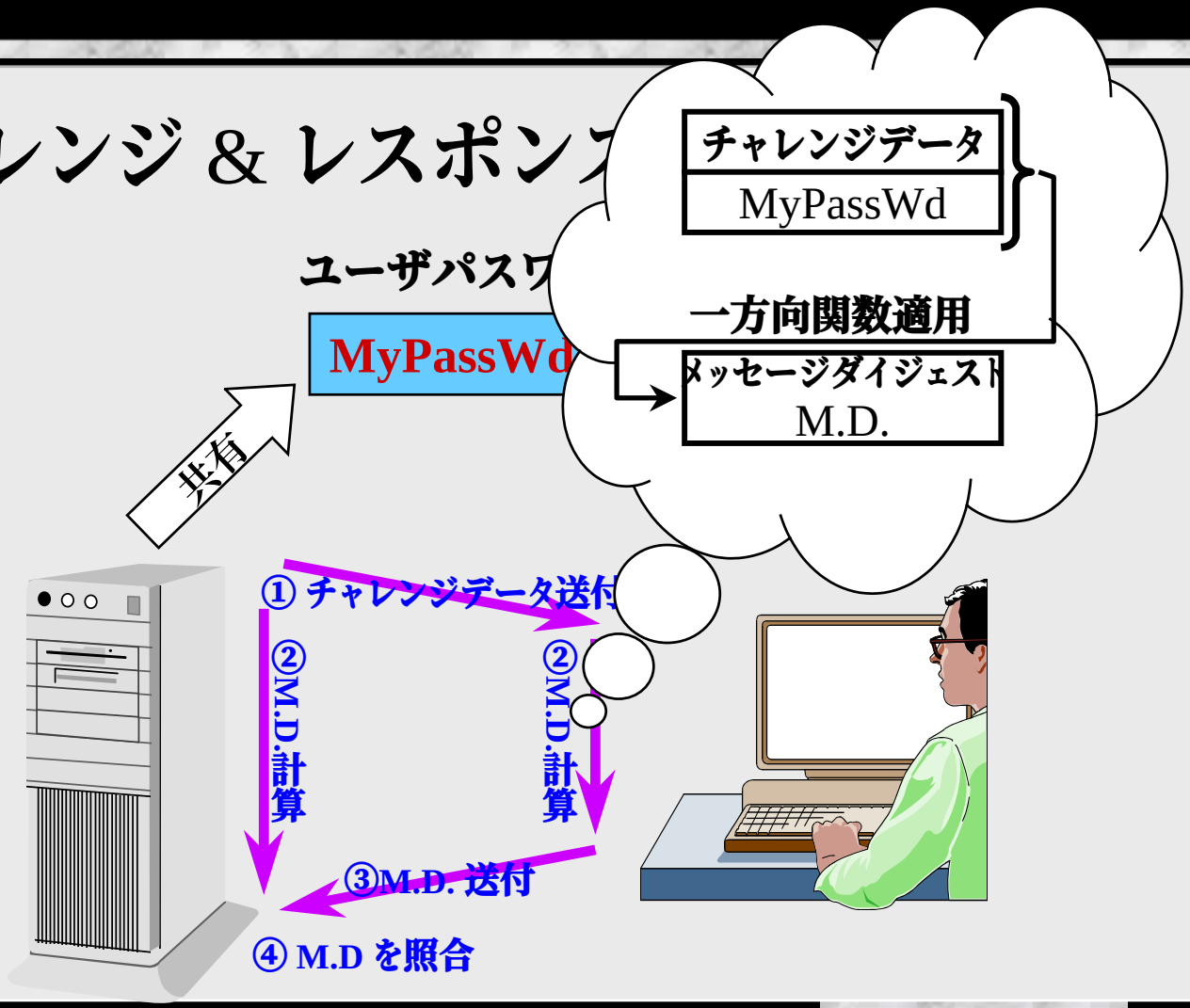
① パスワード送付

② 暗号化と照合

③認証結果通知



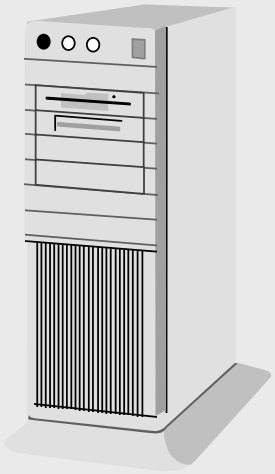
■ チャレンジ & レスポンス



■ ワンタイムパスワード方式 (S/Key 等)

» 前準備 (1)

チャレンジデータ



チャレンジデータ
MyPassWd

一方向関数 *n* 回適用

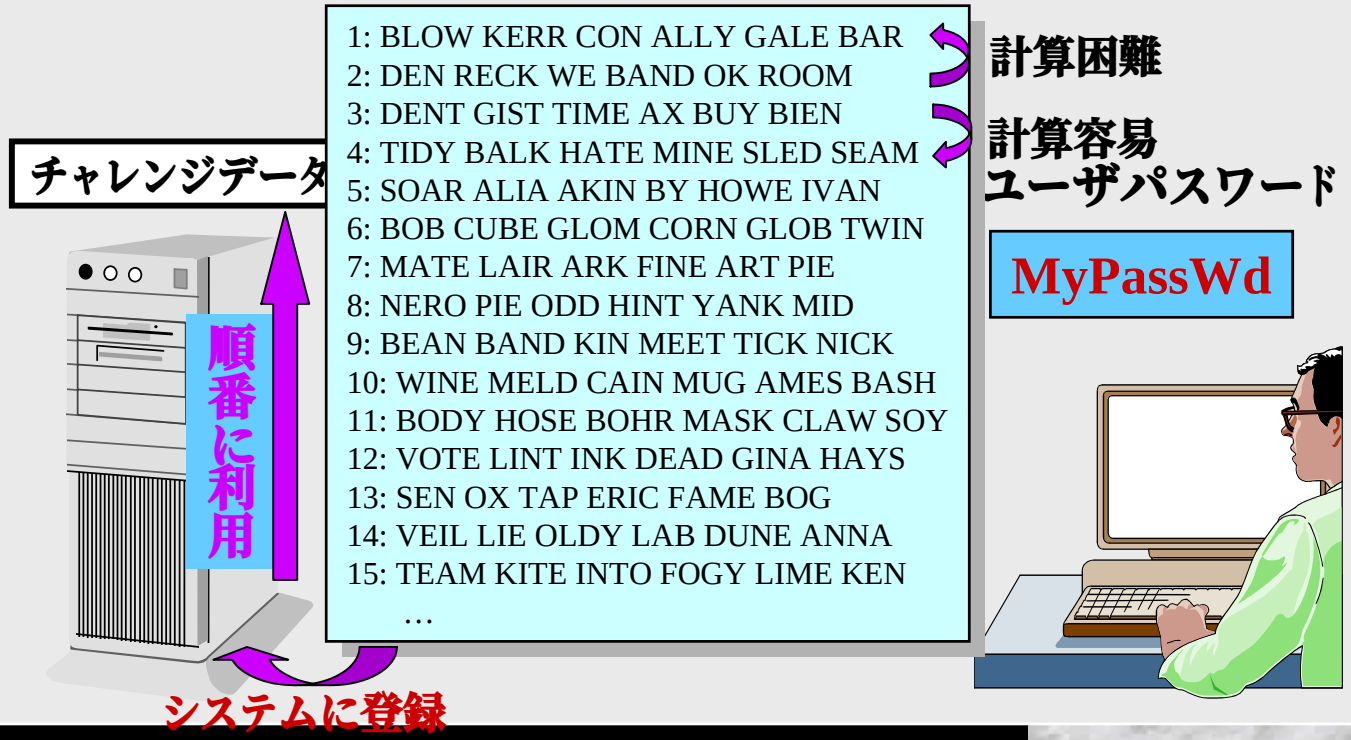
ワンタイムパスワード *n*
O.P.*n*

ユーザパスワード

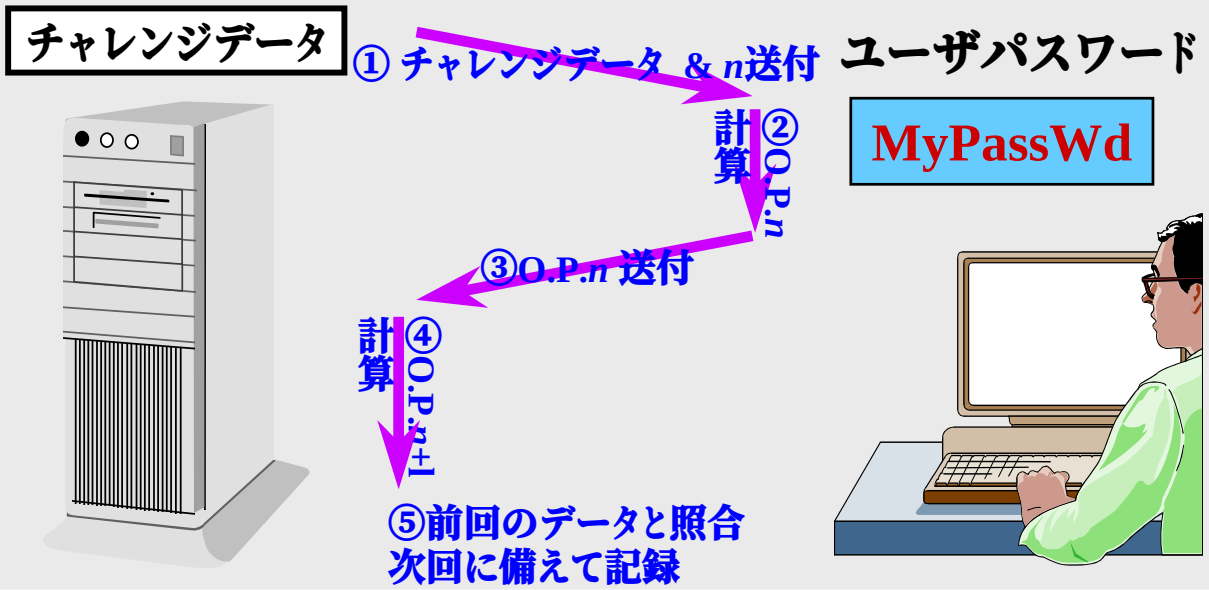
MyPassWd

ワンタイムパスワード方式 (S/Key 等)

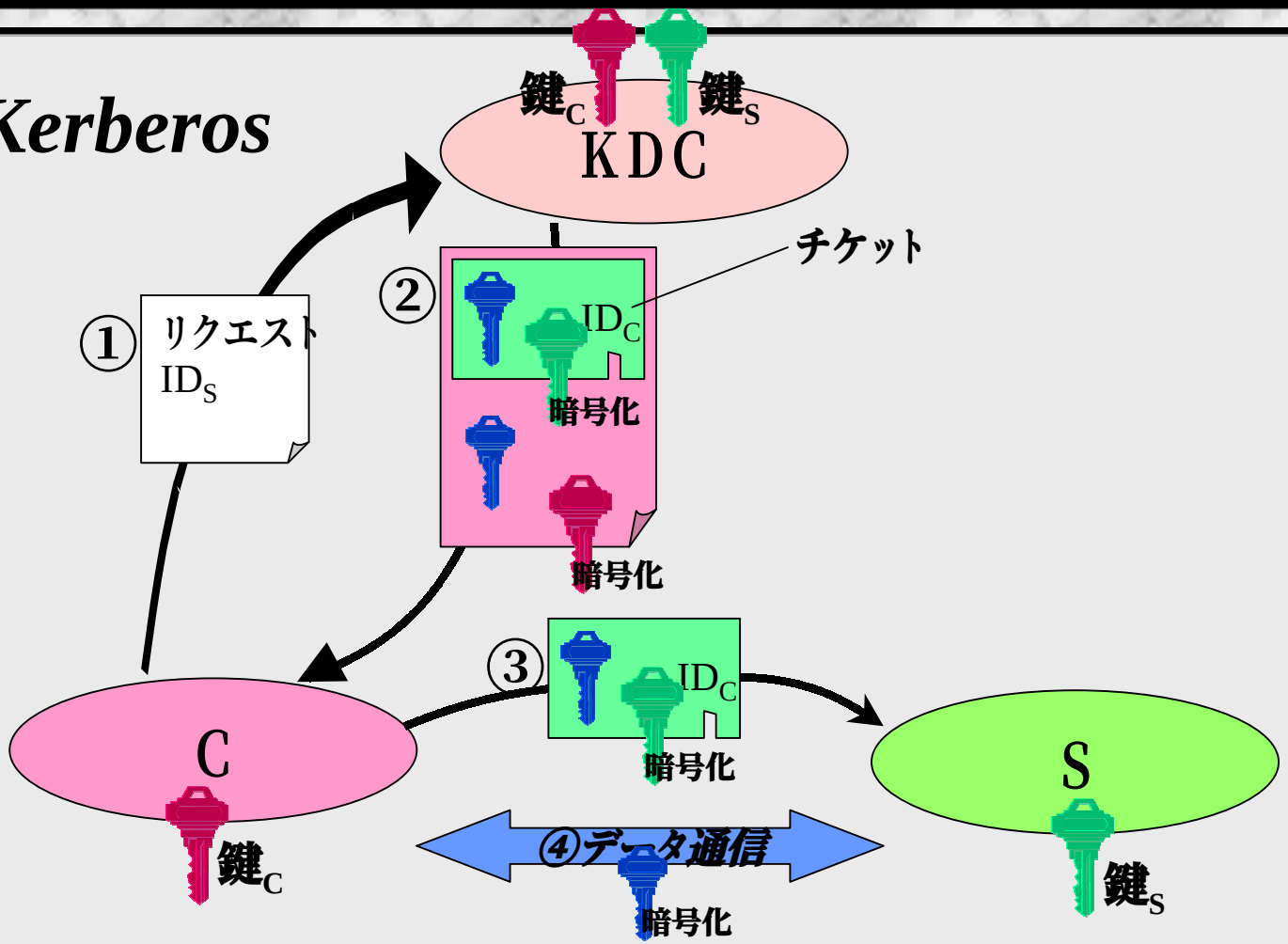
» 前準備 (2)



ワンタイムパスワード方式 (2)



Kerberos

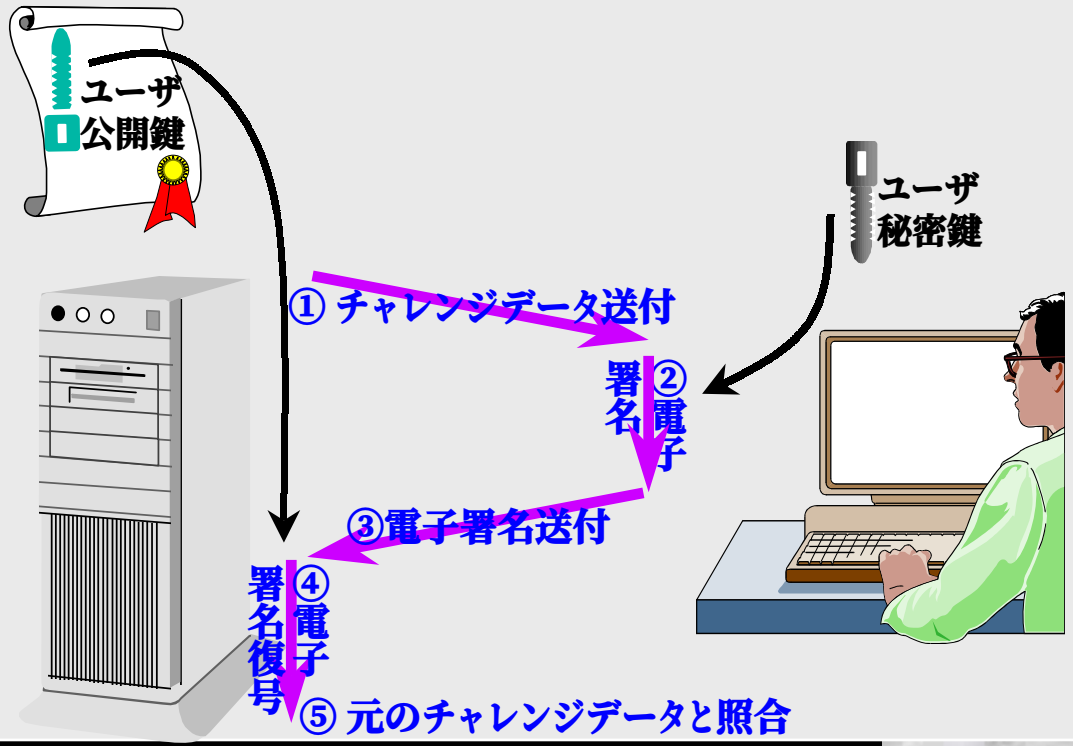


パスワード	秘密情報がそのままネットを流れる
チャレンジ&レスポンス	システム中のパスワード漏洩
ワンタイムパスワード	保護は認証時のみ
Kerberos	KDC（中心的な鍵管理センター）の存在

- いずれも、否認防止には役立たない

- 公開鍵証明書方式
  - 公開鍵証明書と秘密鍵を用いた電子署名処理とで認証を行なう
  - ユーザが署名したデータを公開鍵証明書中の公開鍵で復号できることでユーザの身元を確認

■ 公開鍵証明書方式 (2)

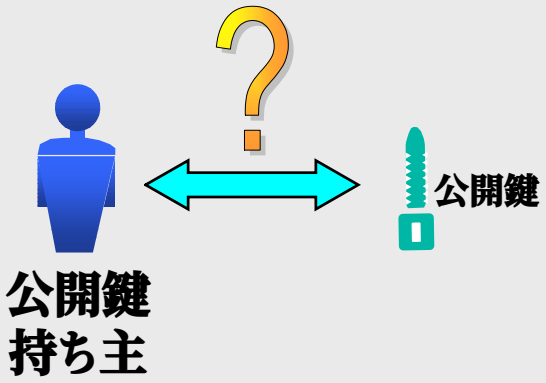


- 公開鍵証明書方式のその他の特徴
  - 非対称暗号技術を利用することで、認証処理後の通常の通信内容を保護することが可能
  - 否認の防止 (電子署名による)
  - ユーザの素性が CA による保証を経たデータとして証明書中に記載されているため、改めてユーザに入力を求めることなく、そのまま利用できる
- 現時点での**究極的な**認証方式

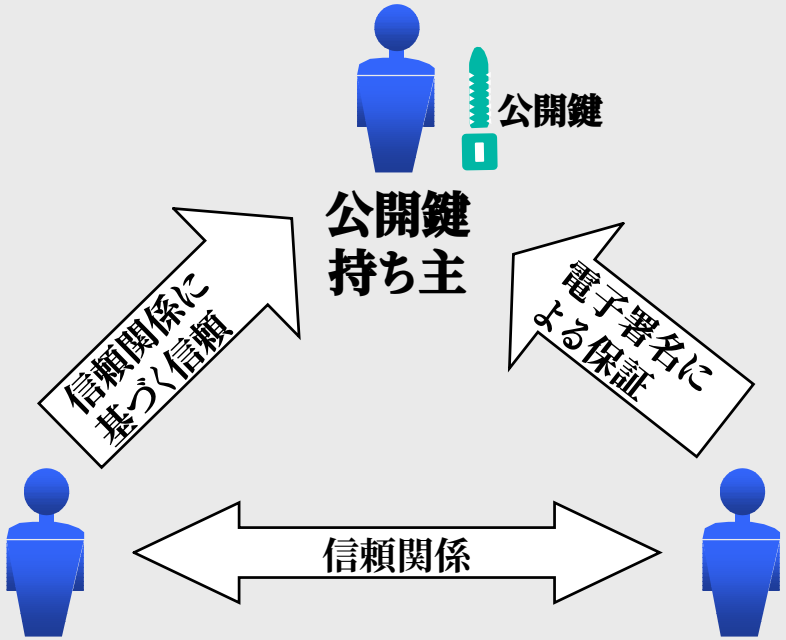
- 自己紹介
- 暗号化技術概説
- 認証技術の発展
- X.509 証明書と公開鍵基盤
- *The Internet Security Protocol*

証明書とPKI

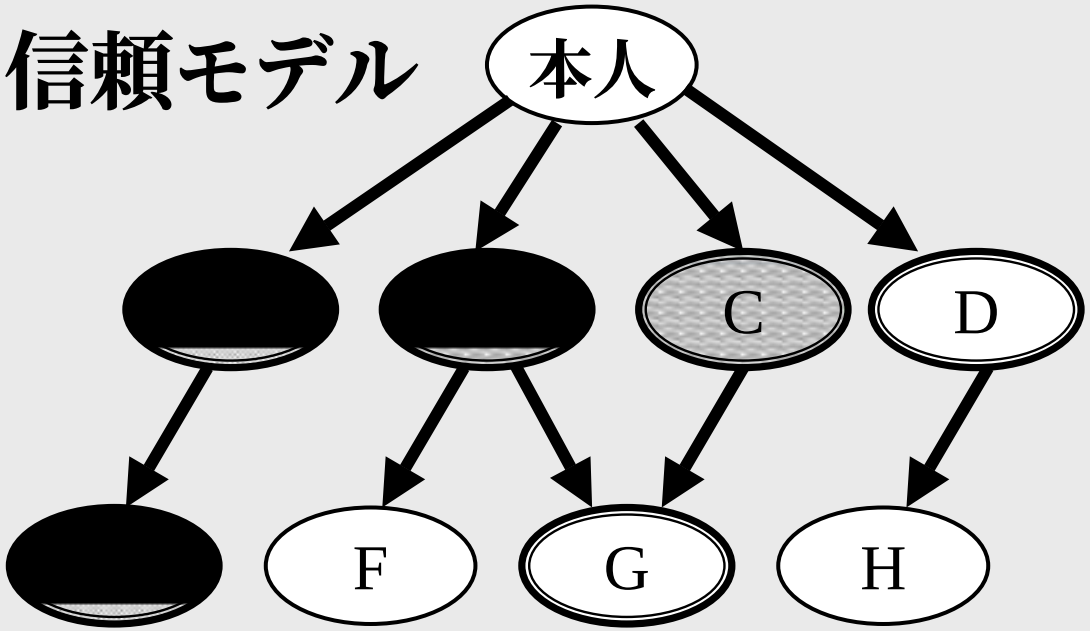
- 非対称暗号技術の補完的役割
- ある公開鍵の持ち主が本当に申告通りの人間であるかどうかを保証するための機構



- 1. **草の根的解決** (PGP)
  - 『友達の友達は友達』方式



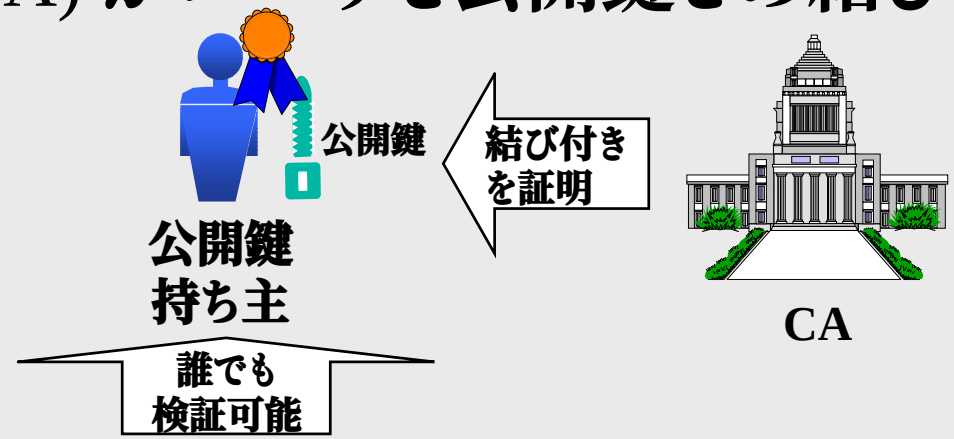
PGP の信頼モデル



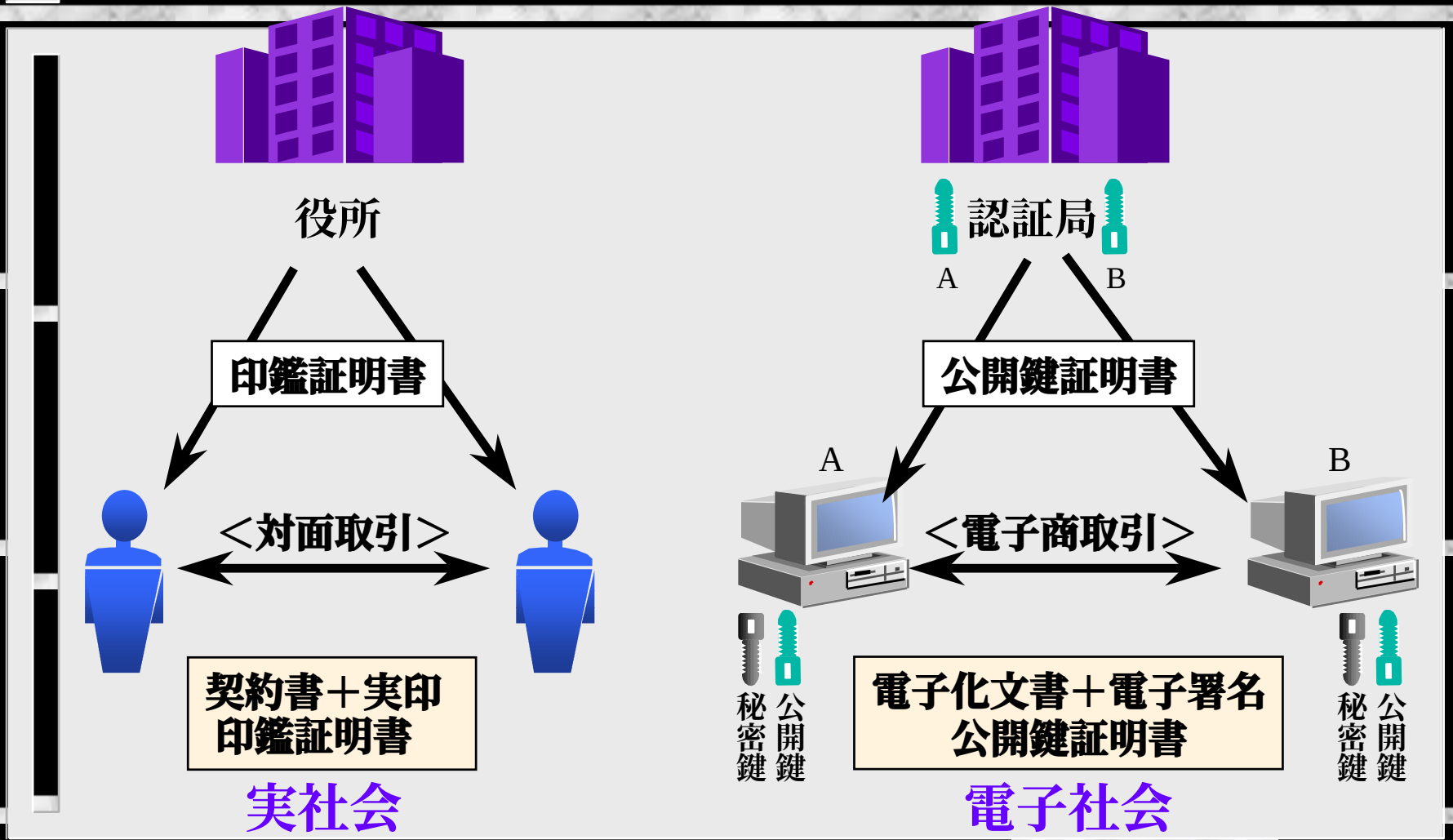
- “草の根的解決”の特徴
  - » とりあえず他の仕組が要らない
    - ➨ 初期段階での普及は容易
  - » ユーザ層が拡大すると、信頼性に不安が
    - ➨ 友達の友達の友達の…が信頼する鍵は信頼できる？
    - ➨ 個人への信頼と、その個人による保証への信頼とを区別する

2. 信頼された第三者機関による保証

- 公開鍵証明書認証局 (= Certificate Authority, CA) がユーザと公開鍵との結びつきを証明



証明書とPKI



公開鍵証明書

- CA が自らの責任のもとにユーザと公開鍵との結びつきを証明したもの
- ユーザの素性 (姓名、所属、電子メールアドレス等) と公開鍵データとに、CA 自らの電子署名を施したもの
- ITU-T 勧告 X.509 (およびその拡張) で定められた仕様に基づくものが主流

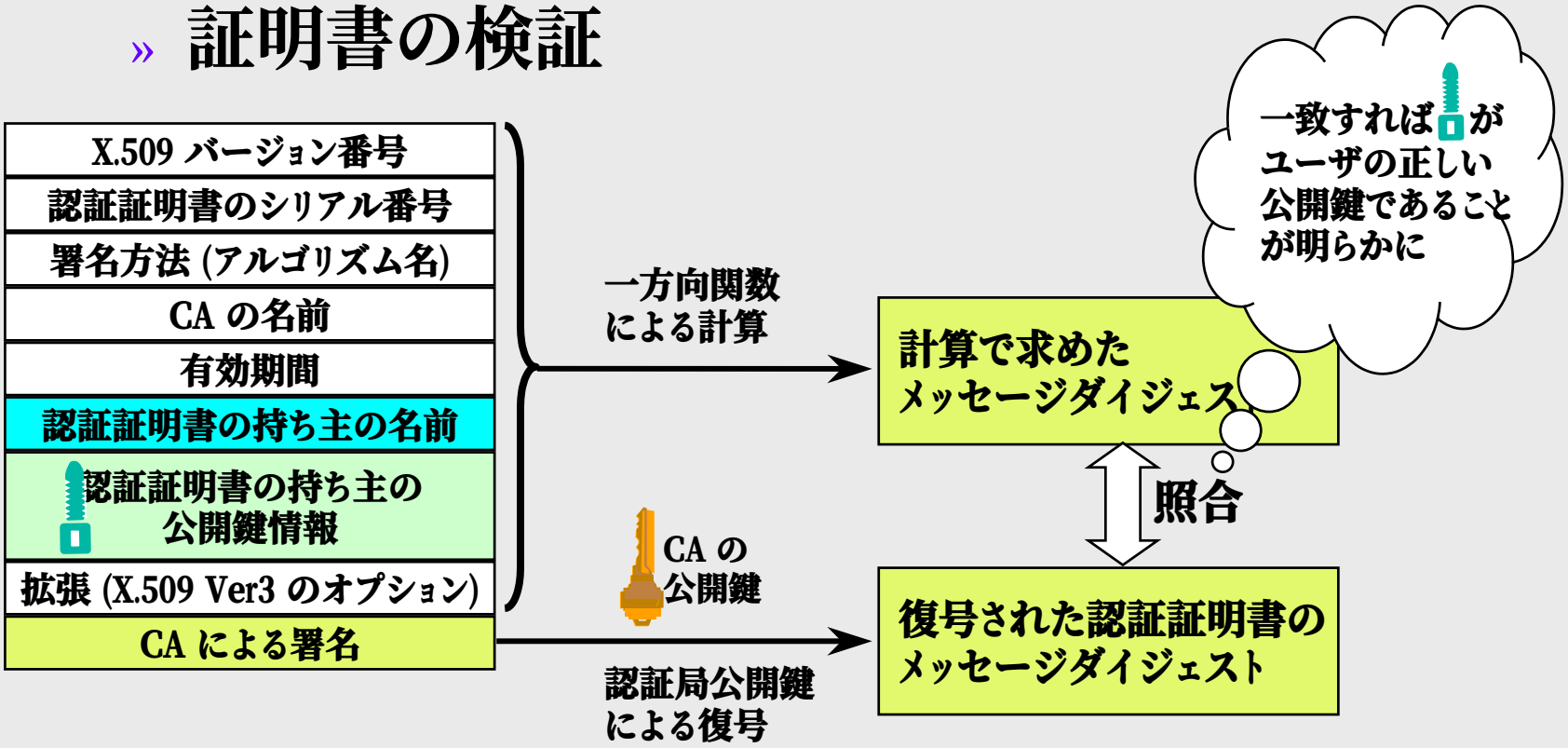
■ 公開鍵証明書 (続き)

» 証明書の内容

X.509 バージョン番号	・・・	X.509 のバージョン
認証証明書のシリアル番号	・・・	認証証明書ごとのユニークな番号
署名方法 (アルゴリズム名)	・・・	この認証証明書の署名方法
CA の名前	・・・	この認証証明書を発行した機関名
有効期間	・・・	この認証証明書の有効期間
認証証明書の持ち主の名前	・・・	登録された公開鍵の申請者の名前
 認証証明書の持ち主の公開鍵情報	・・・	登録された申請者の公開鍵
拡張 (X.509 Ver3 のオプション)	・・・	X.509 の拡張フィールド
CA による署名	・・・	上記全項目に対して一括して施した電子署名

■ 公開鍵証明書 (続き)

» 証明書の検証

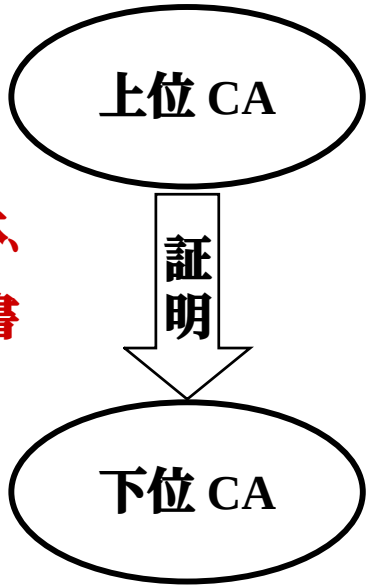


■ CA 自体の公開鍵の保証は？

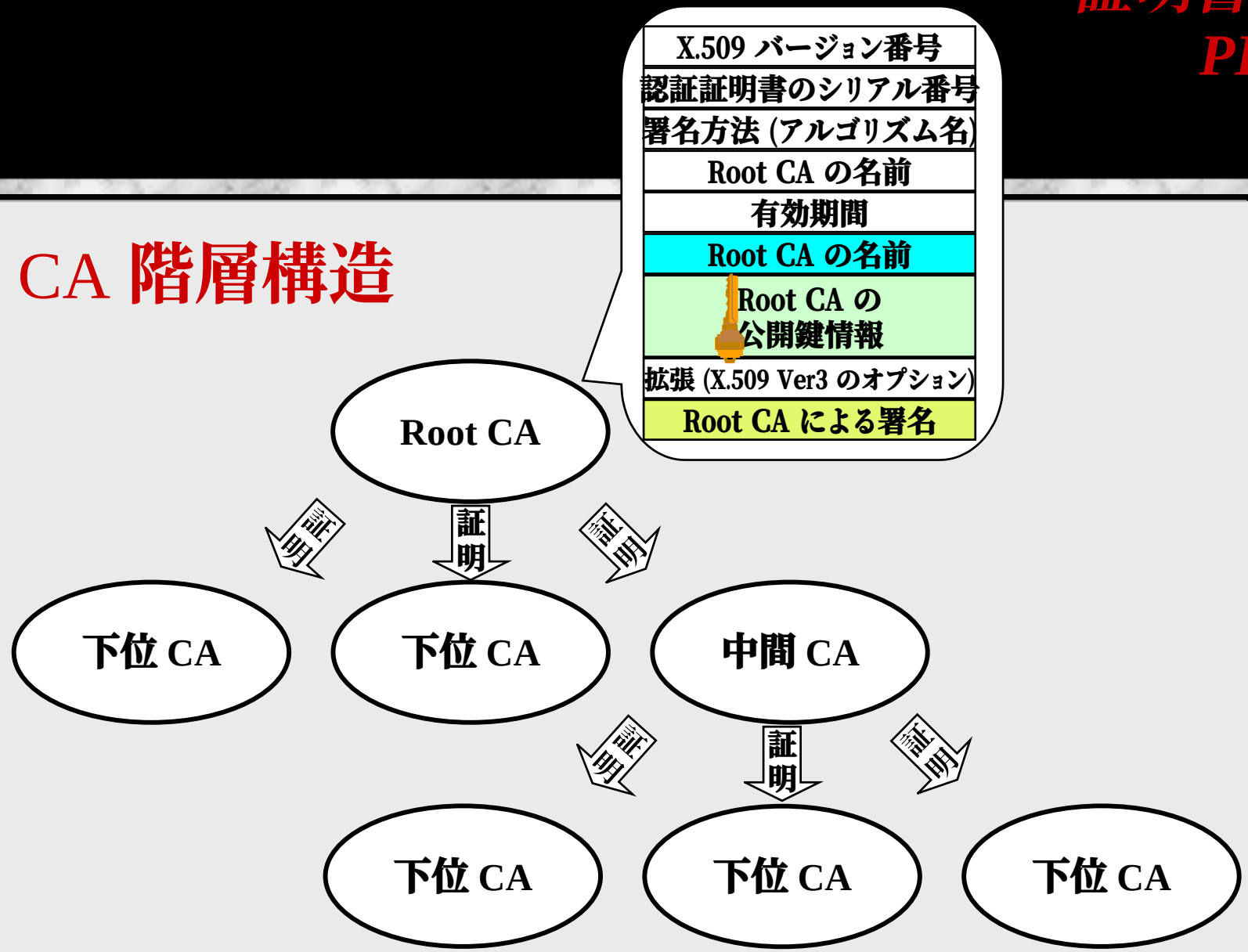
CA公開鍵証明書

X.509 バージョン番号
認証証明書のシリアル番号
署名方法 (アルゴリズム名)
上位 CA の名前
有効期間
下位 CA の名前
 下位 CA の 公開鍵情報
拡張 (X.509 Ver3 のオプション)
上位 CA による署名

CA の公開鍵自体、
上位 CA の証明書
で保証される



■ CA 階層構造

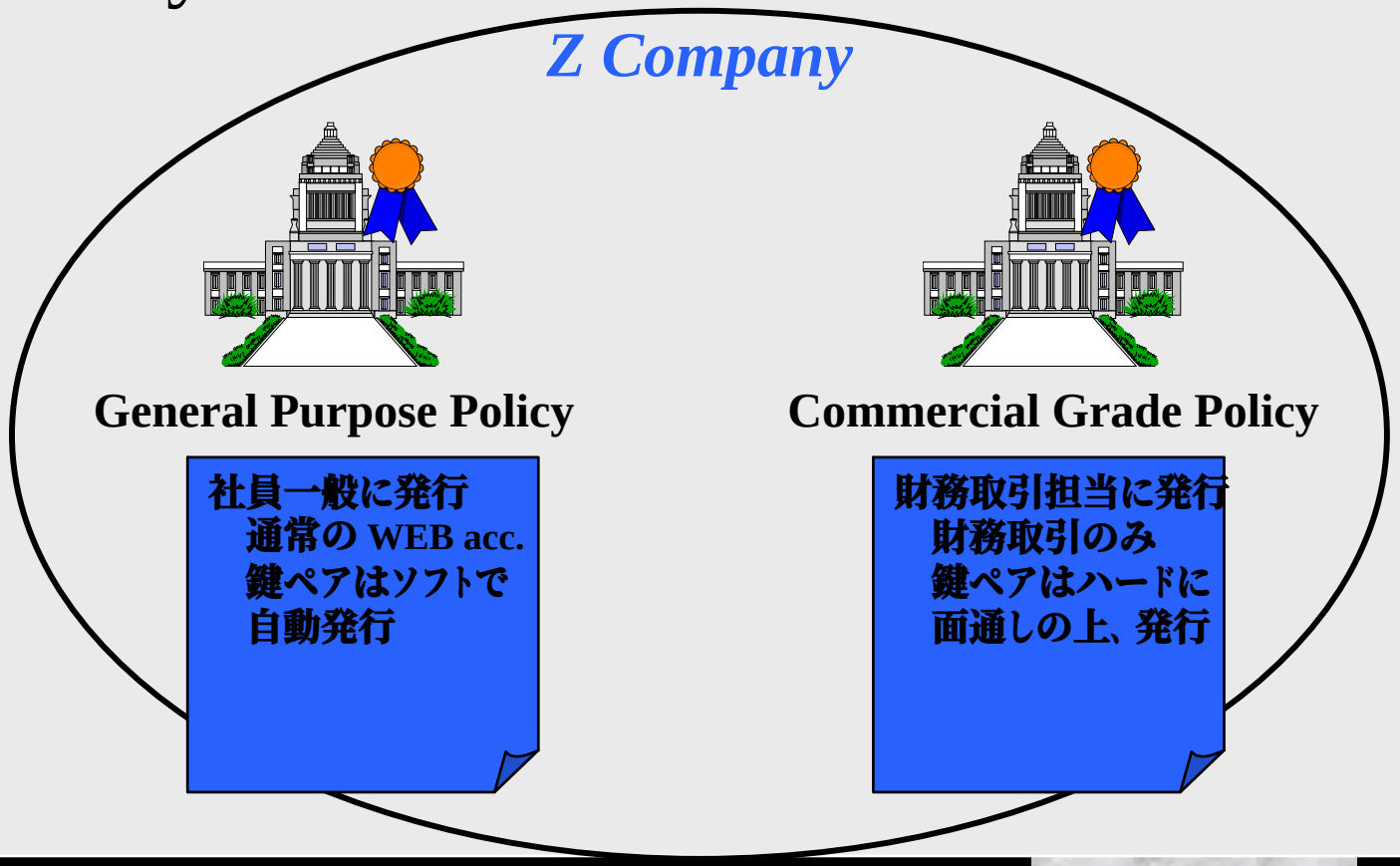


証明書とPKI

- = *Public Key Infrastructure*
- 非対称暗号技術を、広く利用可能とするための枠組
  - » 水道、電力、通信などと同じく、社会的基盤(*Infrastructure*)として

- Policy
  - 共通のセキュリティ要件を持った特定の共同体やアプリケーションのクラスに対して、証明書が適用可能であるかどうかを判断するための名前付けされた規則の集合
    (X.509 より)
  - ユーザが証明書を利用して署名検証などを行なう際に、発行者に対する信頼度を判断する基準となる

Policy の例

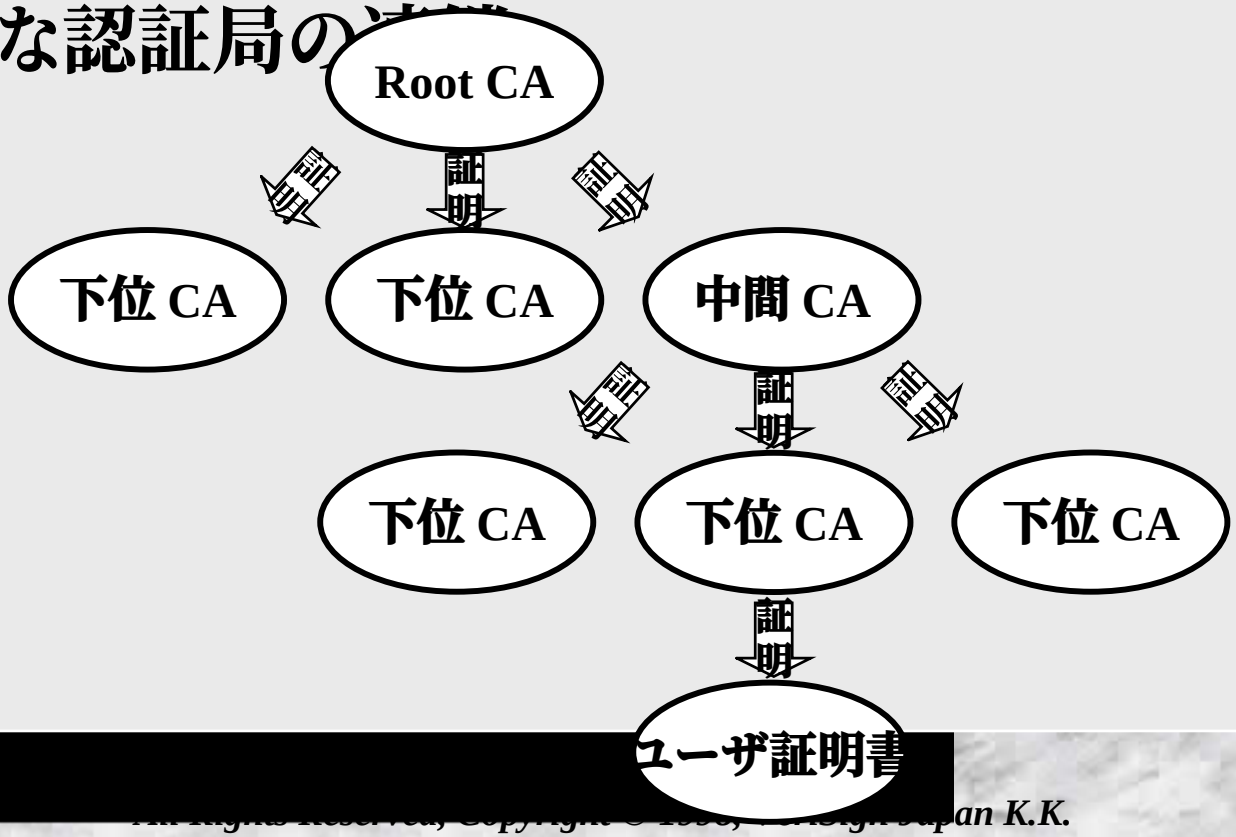


- Policy として定めるべき内容例:
  - 発行対象
    - どのような範囲に証明書を発行するか
      - 従業員
      - 顧客
  - 鍵管理
    - 認証局鍵情報
    - ユーザ鍵情報

- Policy として定めるべき内容例 (続き):
  - 身元確認
    - ユーザの身元を確認する方法
      - E-mail アドレス
      - 社員証の提示
      - 公的資料の確認
  - 運用方針
    - CRL 発行周期
    - 監査手順

- Policy として定めるべき内容例 (続き):
  - 証明書有効期限
  - ユーザ D.N. の一意性保証手段
  - 法的要件
    - 認証局の負うべき責任範囲
  - Policy 管理
    - Policy 決定権限者
    - Policy 維持管理と発行方法

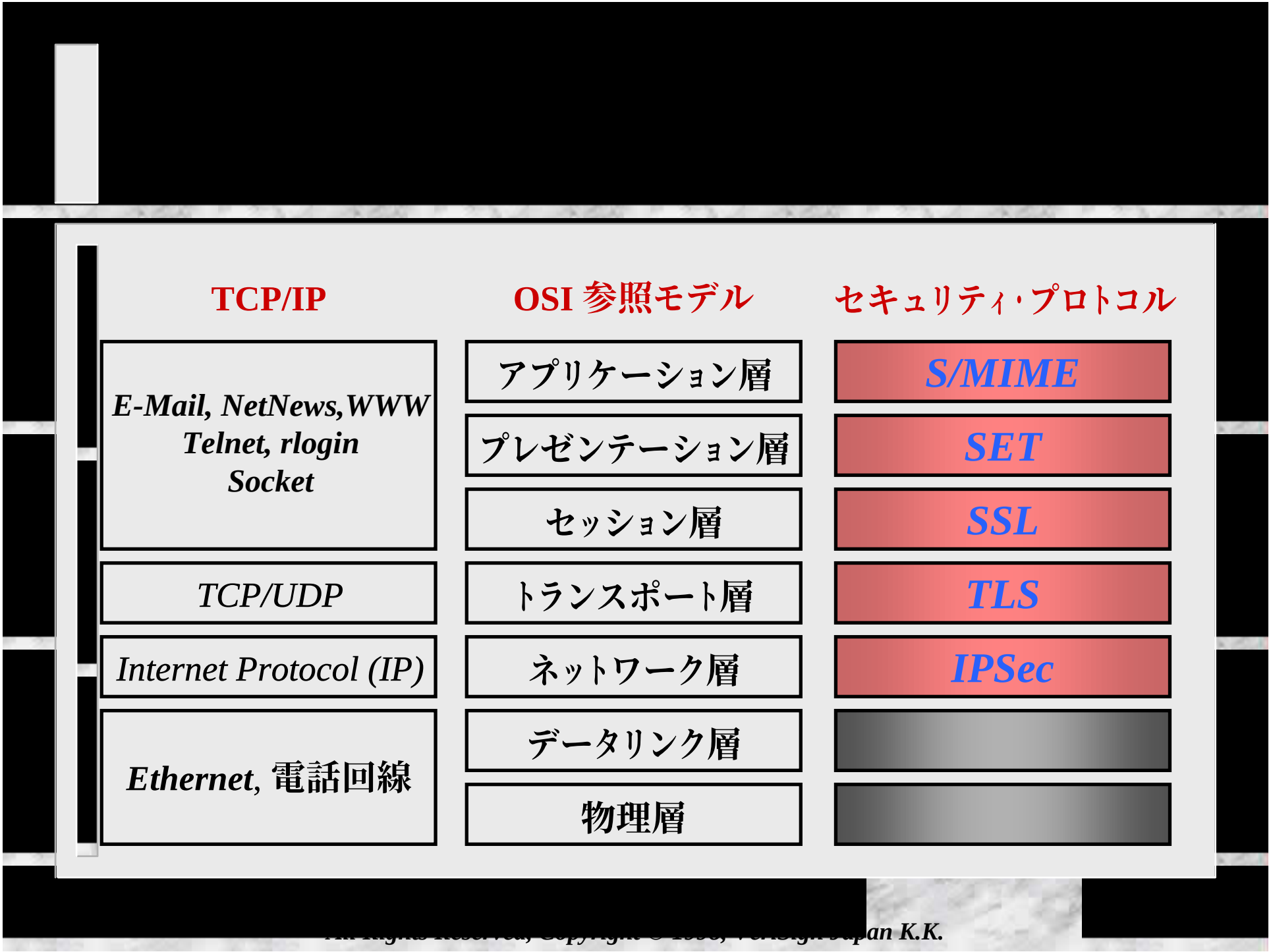
- 証明書パス
  - ある証明書の有効性をたしかめるために必要な認証局の連鎖



証明書パスの探索

- 以下のいずれかが可能ならば、証明書パスを検出できる
  - 認証局の名前から、その認証局公開鍵に対して証明書を発行した親認証局の証明書を見付けられる
  - 認証局の名前から、その認証局が他の認証局公開鍵に対して発行した証明書を見付けられる
- 両方が可能ならば、より効率的な探索が実現

- 自己紹介
- 暗号化技術概説
- 認証技術の発展
- X.509 証明書と公開鍵基盤
- *The Internet Security Protocol*



S/MIME

- RSADSI 社が提唱
- アプリケーションではなく、最初から標準規格となるべく誕生
- MIME *(Multipurpose Internet Mail Extensions)* に準拠
  - 可視コード化処理などは MIME に任せる
  - MIME 形式ならばどこでも利用可能 (WEB, EDI, インターネット FAX なども視野に)

S/MIME

- 米国民間暗号技術研究／開発の総本山
- PKCS *(Public-Key Cryptography Standards)* の電子メールへの適用
  - 特に PKCS#7 と PKCS#10
- Netscape Messenger、Microsoft Outlook Express に採用されたため、一気にシェアを拡大

■ 公開鍵暗号のデファクト標準 (1993 〜)

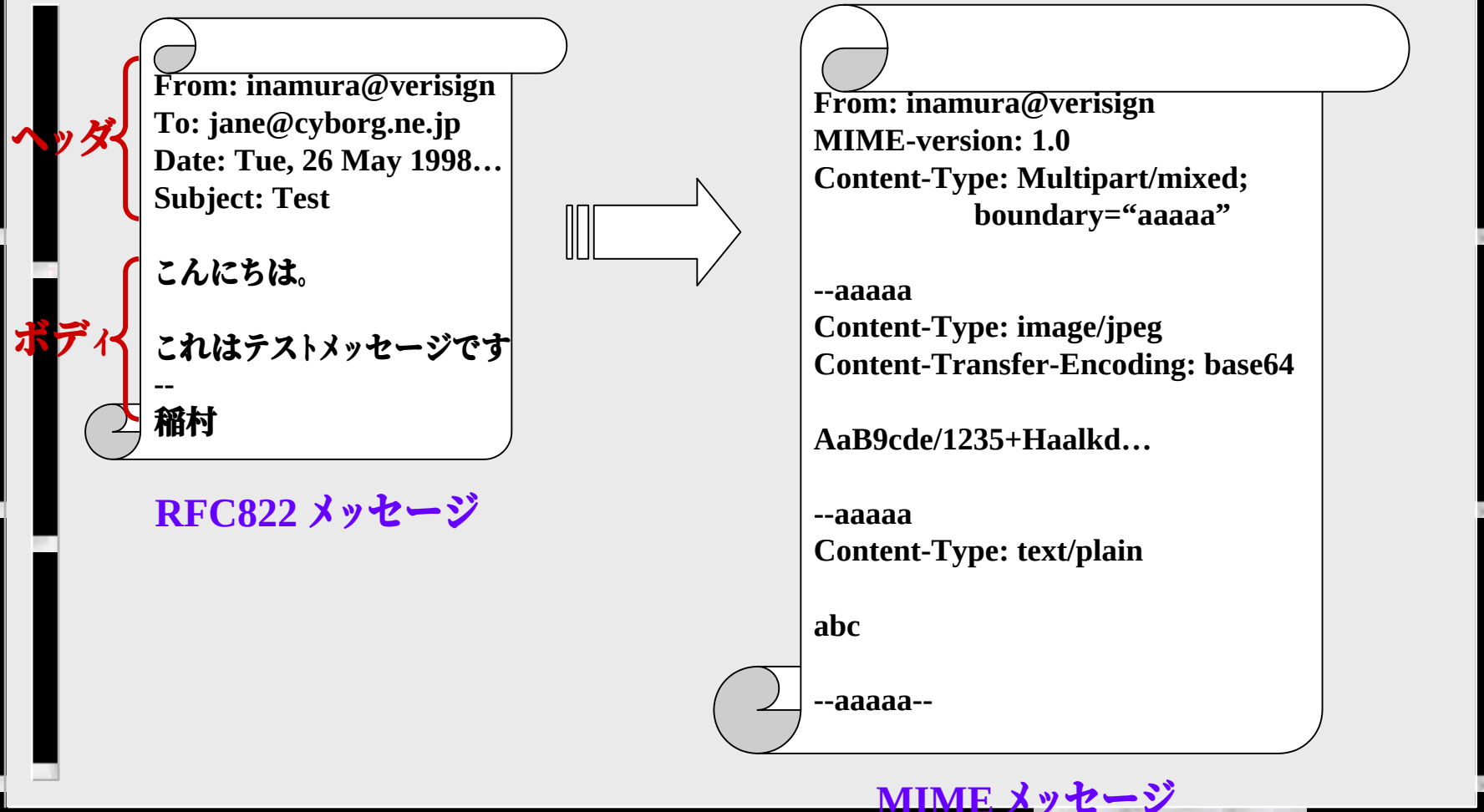
PKCS #1	RSA Encryption
PKCS #3	Diffie-Hellman Key-Agreement
PKCS #5	Password-Based Encryption
PKCS #6	Extended-Certificate Syntax
PKCS #7	Cryptographic Message Syntax
PKCS #8	Private-Key Information Syntax
PKCS #9	Selected Attribute Types
PKCS #10	Certificate Request Syntax
PKCS #11	Cryptographic Token Interface
PKCS #12	Personal Information Exchange Syntax
PKCS #13	Elliptic Curve Cryptography

S/MIME

- インターネットメール標準 (RFC822)
  - 基本的にフラットなテキストデータのみ
  - 7bit データ (特に、US-ASCII と呼ばれる)
- MIME = RFC822 に対する機能拡張
  - マルチメディアデータ
  - US-ASCII 以外の文字セット
  - 構造を持ったデータの通信
    - マルチパートメッセージ
    - 部分メッセージ

- いくつかのメッセージヘッダを追加
  - *MIME-Version:*
    - MIME 規格のバージョンを指定 (今後の拡張)
  - *Content-Type:*
    - 含まれるデータの形式を指定
      - *text, image, audio, video, application, message, multipart*
  - *Content-Transfer-Encoding:*
    - データの符号化方式の指定
      - *base64, quoted-printable, 7bit, 8bit, binary*
    - 7bit データしか送れない配送系を通過可能に

S/MIME



Security Multiparts for MIME (RFC1847)

```
From: inamura@verisign
MIME-version: 1.0
Content-Type: Multipart/signed;
              boundary="aaaaa"

--aaaaa
Content-Type: text/plain;
              charset=iso2022-jp

これは署名メッセージの例です。

--aaaaa
Content-Type: application/x-signature;
Content-Transfer-Encoding: base64

AaB9cde/1235+Haalkd…
--aaaaa--
```

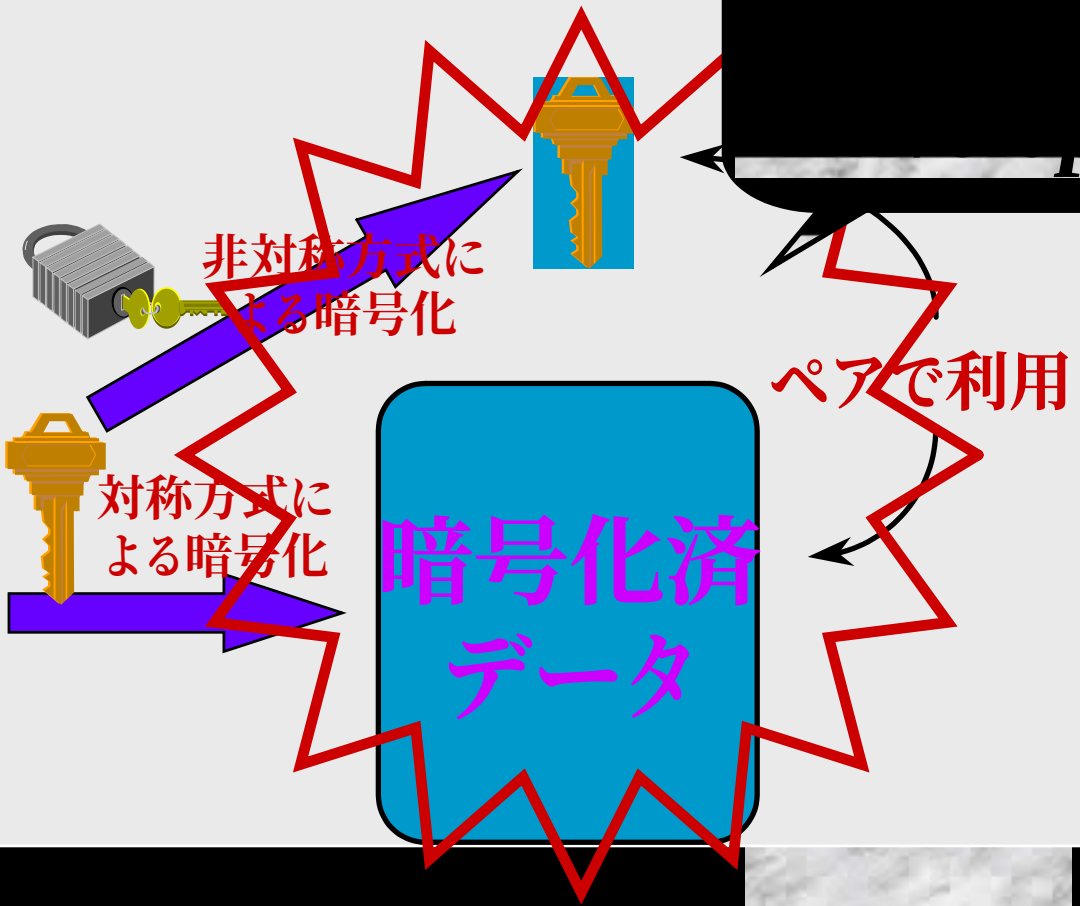
署名メッセージ

```
From: inamura@verisign
MIME-version: 1.0
Content-Type: Multipart/encrypted;
              boundary="aaaaa"

--aaaaa
Content-Type: application/x-key;
Content-Transfer-Encoding: base64

AaB9cde/1235+Haalkd…

--aaaaa
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64

AaB9cde/1235+Haalkd…
--aaaaa--
```

暗号化メッセージ

```
EnvelopedData ::= SEQUENCE {
    version Version,
    receipientInfos RecipentInfos,
    encryptedContentInfo
        EncryptedContentInfo}
```

EnvelopedData



MIME-Version: 1.0
Content-Type: text/plain

This is a test Mail for testing S/MIME....

データ

MIME 形式として準備

S/MIME

```
SignedData ::= SEQUENCE {
  version Version,
  digestAlgorithms DigestAlgorighmIdentifiers,
  contentInfo ContentInfo,
  certificates
    [0] IMPLICIT ExtendedCertificatesAndCertificates
          OPTIONAL,
  crls
    [1] IMPLICIT CertificateRevocationLists OPTIONAL,
  signerInfos SignerInfos }
```

- 署名付きデータ形式
  - *SignedData (PKCS オリジナル形式)*
    - 元データと署名データ両者をまとめて PKCS 可
    - 一般に PKCS 非互換のツールでは非可読
  - *Multipart/signed (MIME 標準形式)*
    - 元データ部
    - 署名
      - →

```
Content-Type: multipart/signed;
     protocol="application/pkcs7-signature"; micalg=sha-1;
     boundary=0000000001boundary

-- 0000000001boundary
Content-Type: text/plain

This is a test.

-- 0000000001boundary
Content-Type: application/pkcs7-signature; name=test.p7s
Content-Transfer-Encoding: base64

ghyHhHUujhJhjH77...
```

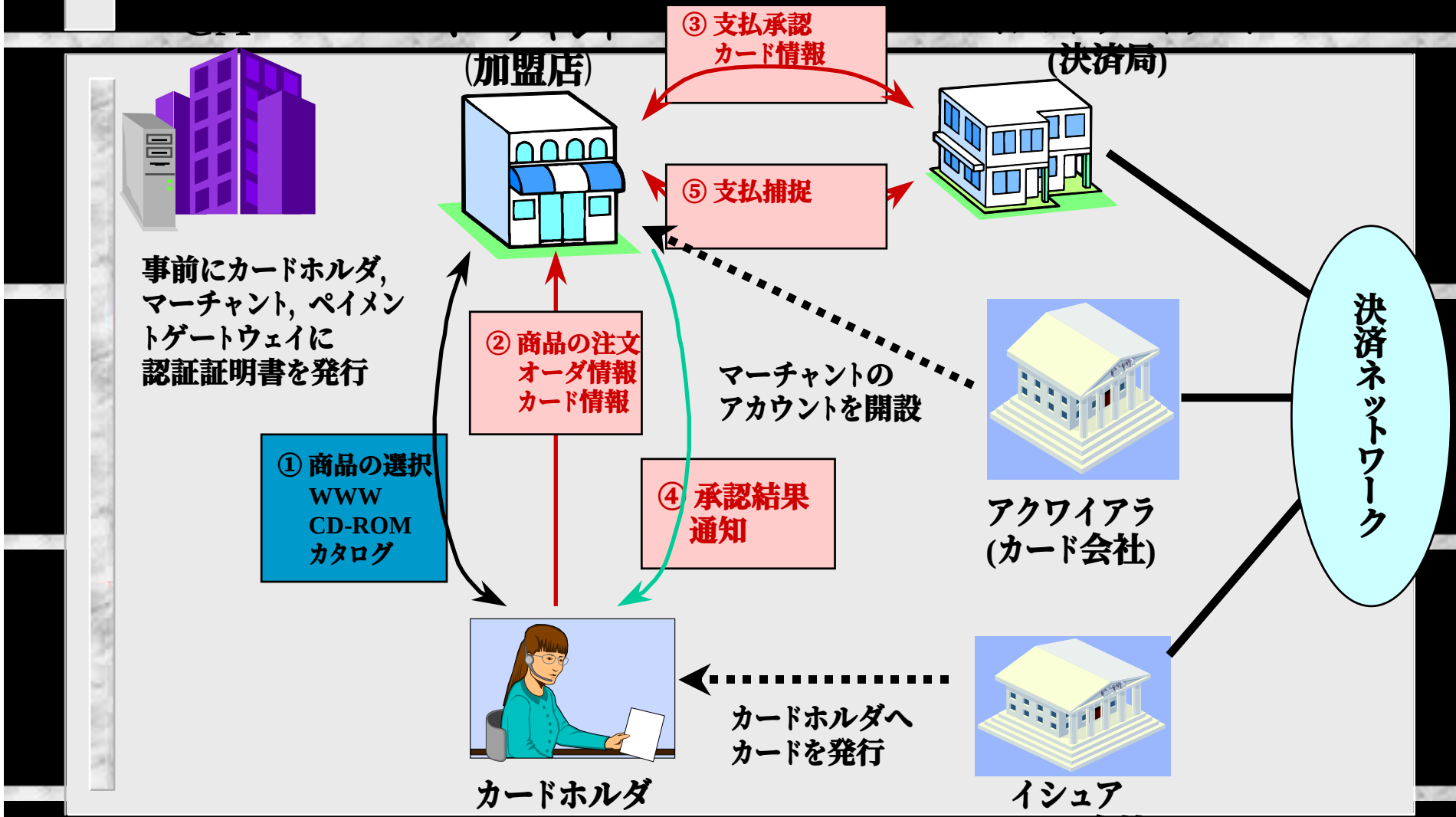
S/MIME

- ハイブリッド暗号方式
  - 対称暗号　　→ ***RC4/40*** **,** ***DES-EDE3*** (Min. Req.)
  - 非対称暗号　→ ***RSA, D-H***
- 電子署名
  - 一方向関数　→ ***MD2, MD5, SHA-1***
  - 非対称暗号　→ ***RSA, DSA***
- 可視コード化
  - MIME に一任

SET

- クレジットカードを決済手段として用いる電子商取引の規格
- Visa と MasterCard が規格提案
  - Version 1.0 @ 1997 年 5 月
- クレジットカードを用いた決済をネットワーク経由で安全に行なうことが目的

SET

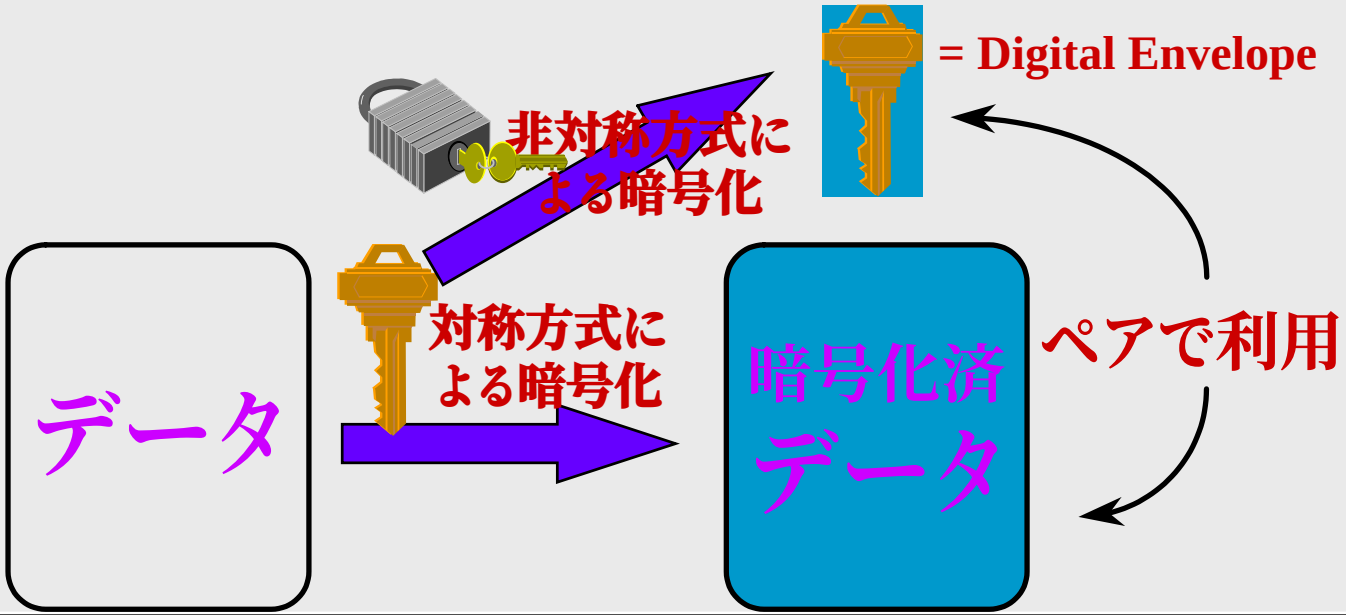


SET

- 対称／非対称暗号技術の共用
  - *Digital Enveloping*
  - *56bit DES & 1024bit RSA*
- 二種類の公開鍵／秘密鍵ペア
  - 署名用と鍵配送用
  - カード保持者は署名用の鍵だけで OK
- *OAEP*
- 二重署名

OAEP

- = Optimal Asymmetric Encryption Padding
- Digital Envelope 部分をより安全に

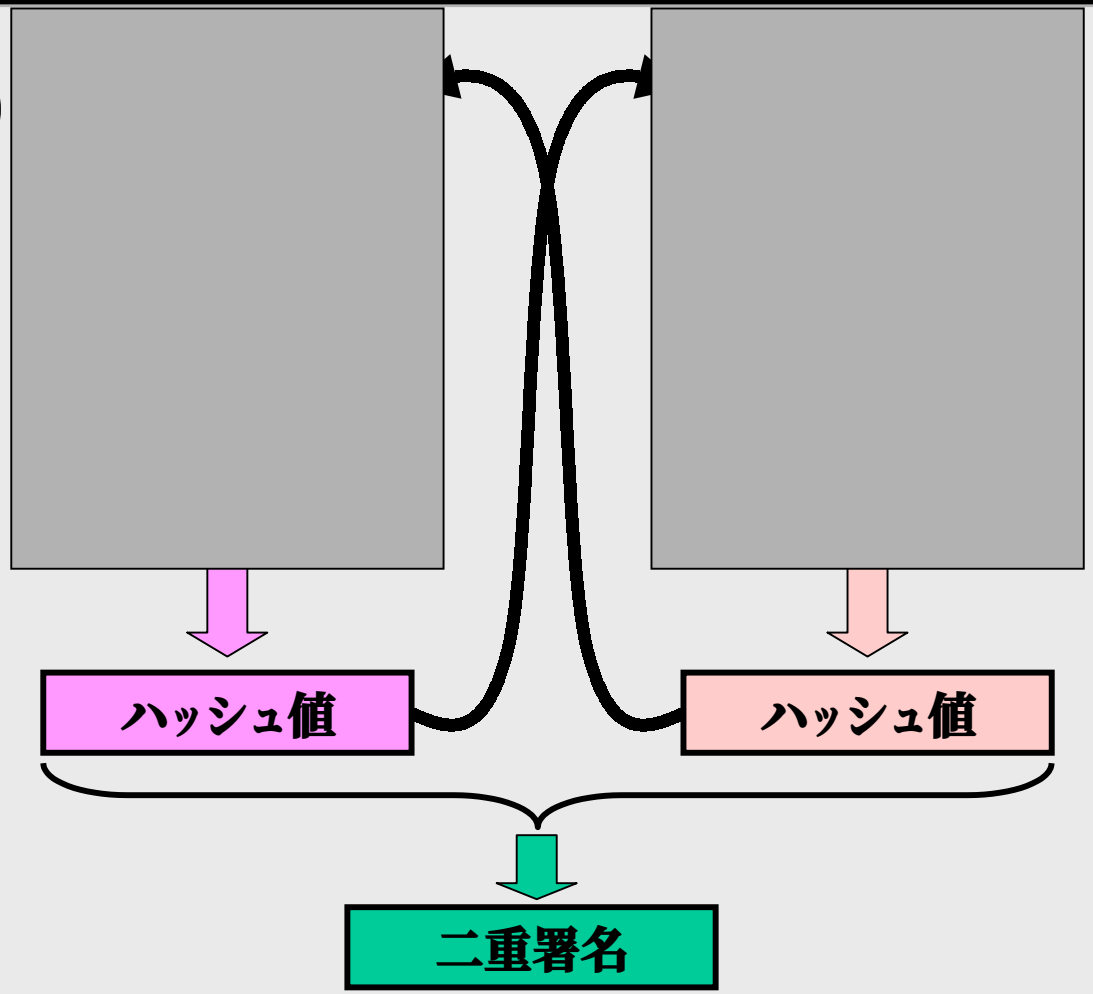


OAEP (2)

- 一般に *Digital Envelope* のサイズは非対称方式 (この場合は RSA) の鍵長程度
- 対称方式の鍵長 ≪ 非対称方式の鍵長
- 対称鍵の他に種々のデータを詰める (Pad)。適切な *Padding* により、平文を知らないままで *Envelope* 部を推測することが不可能に
  - カード情報
  - 暗号化されたデータのハッシュ値
  - ランダム値 *etc.*

- 二重署名
  - 取り引きの機密性を高めるため
  - カード保持者は:
    - 商人には自らの口座情報を知られたくない
    - 銀行には購入した商品の内容を知られたくない
    - 指定した以外の取り引き内容では金が動いて欲しくない
  - 解決策が二重署名

■ 二重署名 (2)



SSL

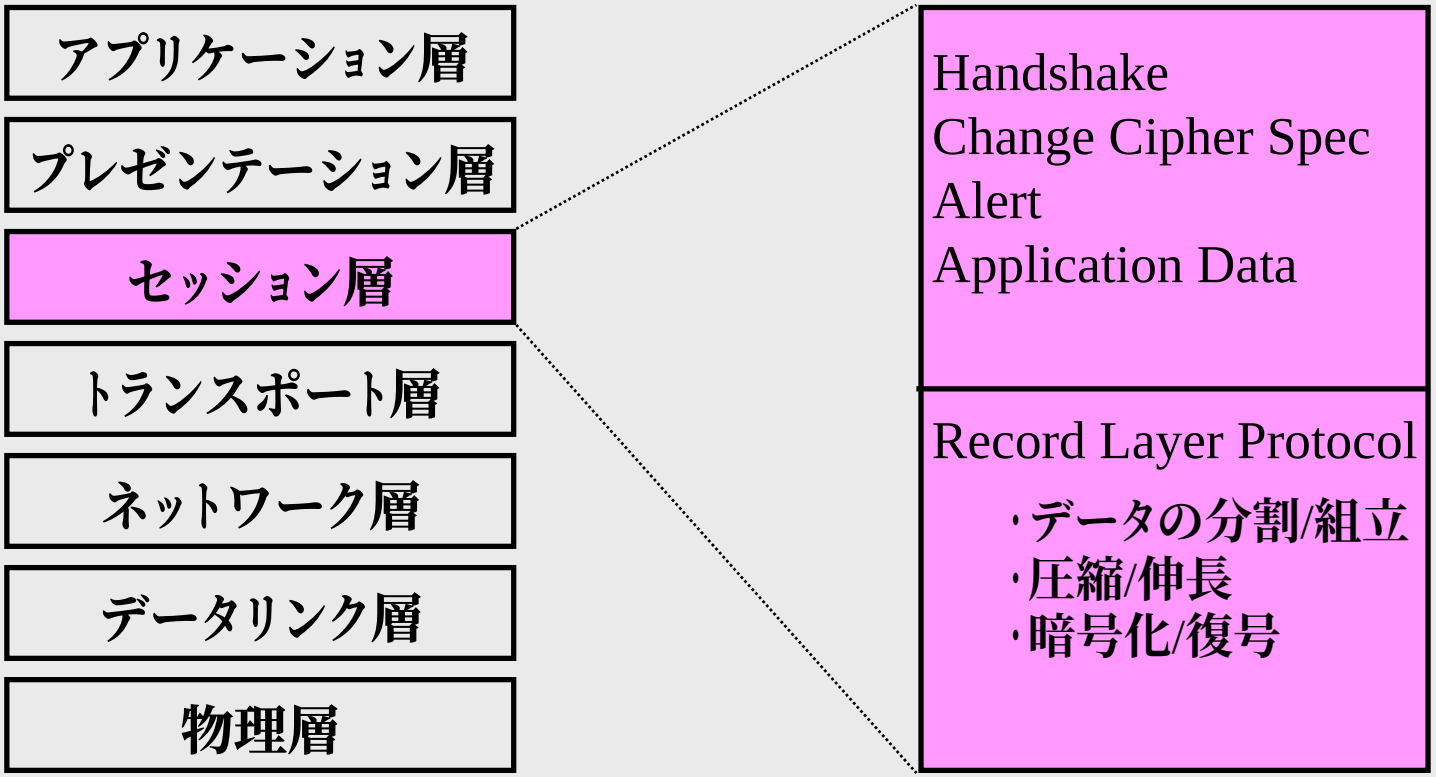
- Netscape Communications 社提唱
- 安全なソケット接続を提供
- 非常に多くの暗号化方式をサポート
- いまのところ、WWW での利用が主

SSL

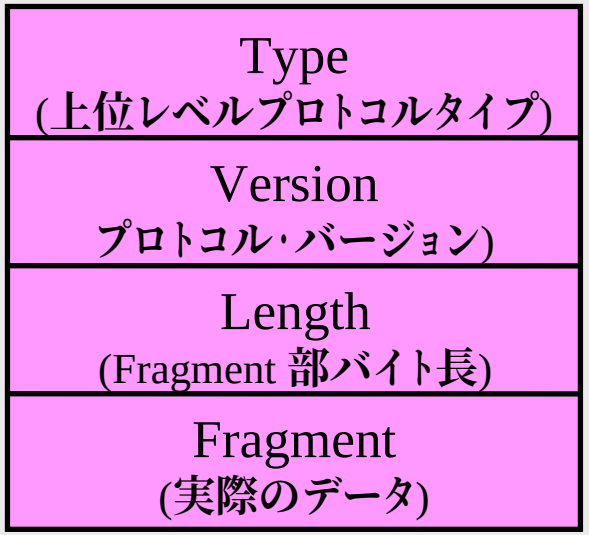
- Netscape Commerce Server, Netscape Navigator **などに搭載**
  - **ただし、日本で利用できるのは暗号強度の低い (鍵の秘密データは** 40bit) **輸出版のみ**
    - **金融機関などでは強力版も利用可能に**
    - ***Fortify for Netscape***
- **フリーなライブラリも存在**
  - ***SSLeay***
    - **最高強度の暗号機能が利用可能**

SSL

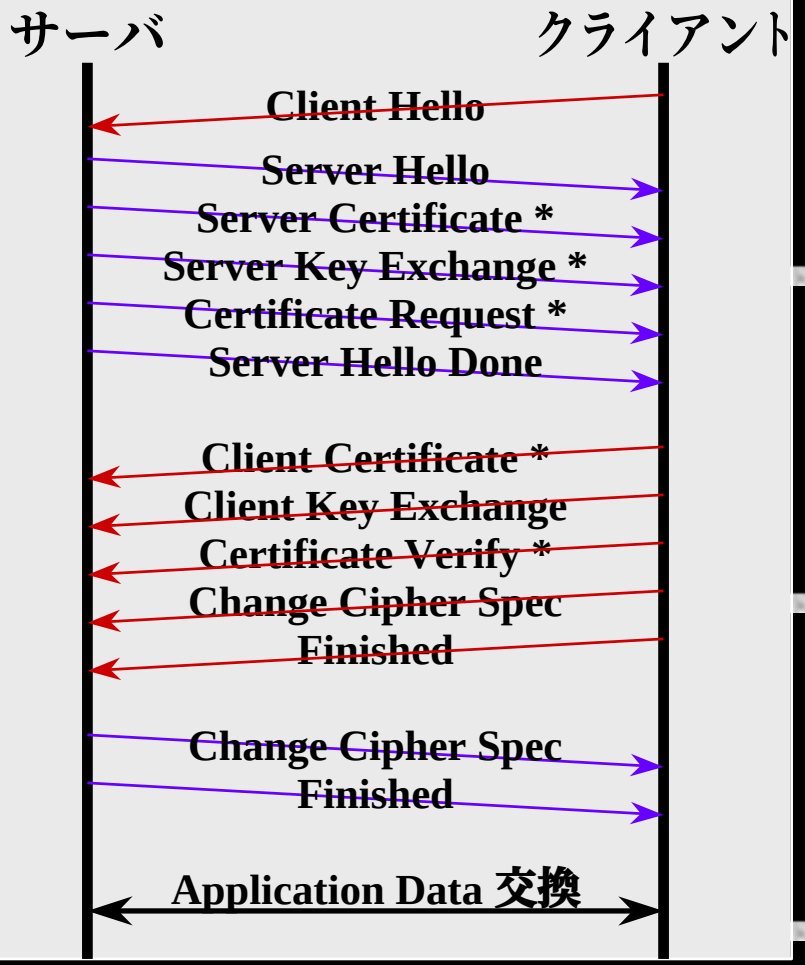
OSI 参照モデル



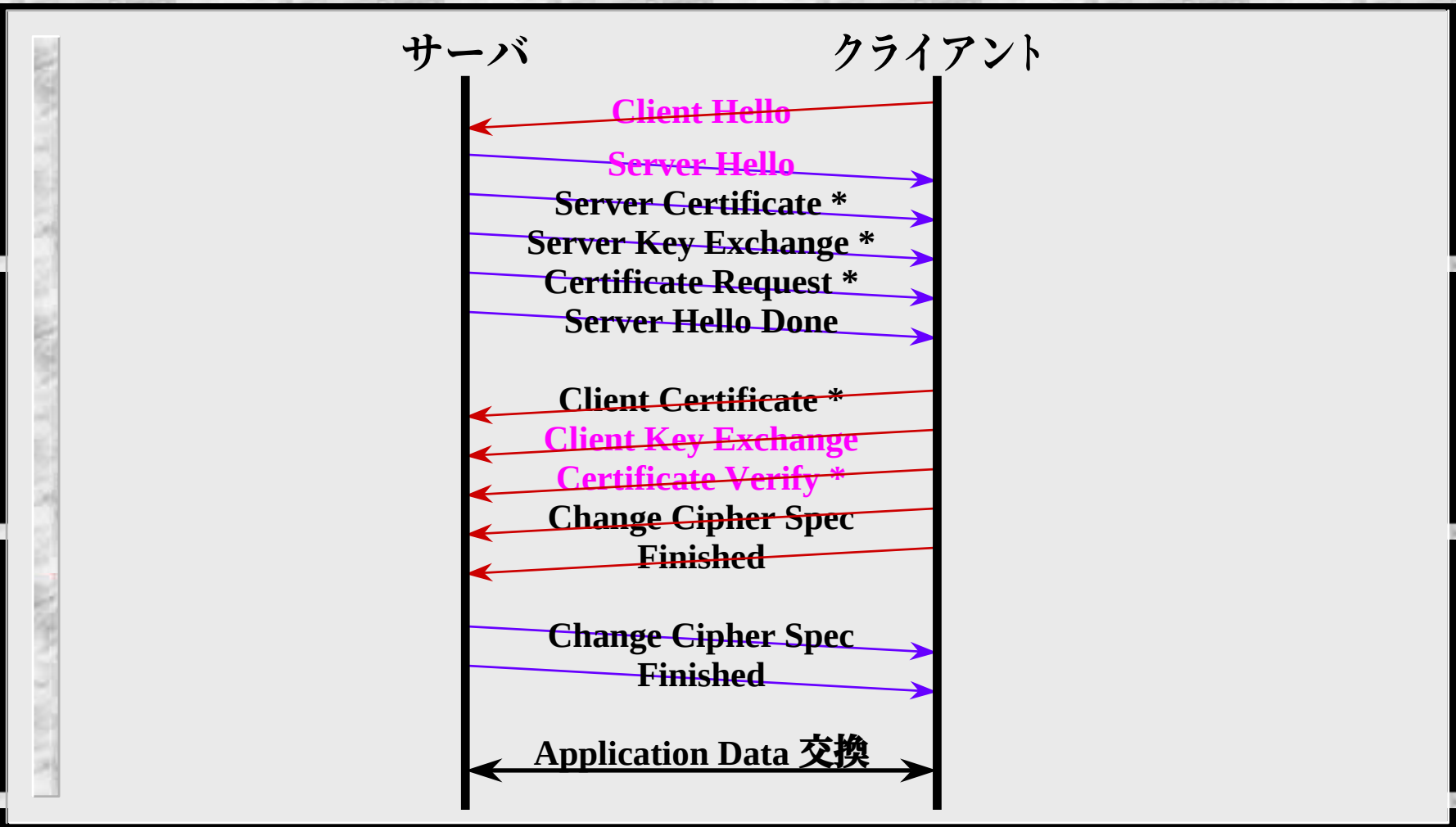
- Record Layer Protocol
  - 2^{14} Byte 以下にデータを分割
  - (必要なら) 圧縮処理
  - (必要なら) 認証データ付加
  - (必要なら) 暗号化処理
  - 右図のようなデータ構造体を生成して送受



- 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



SSL



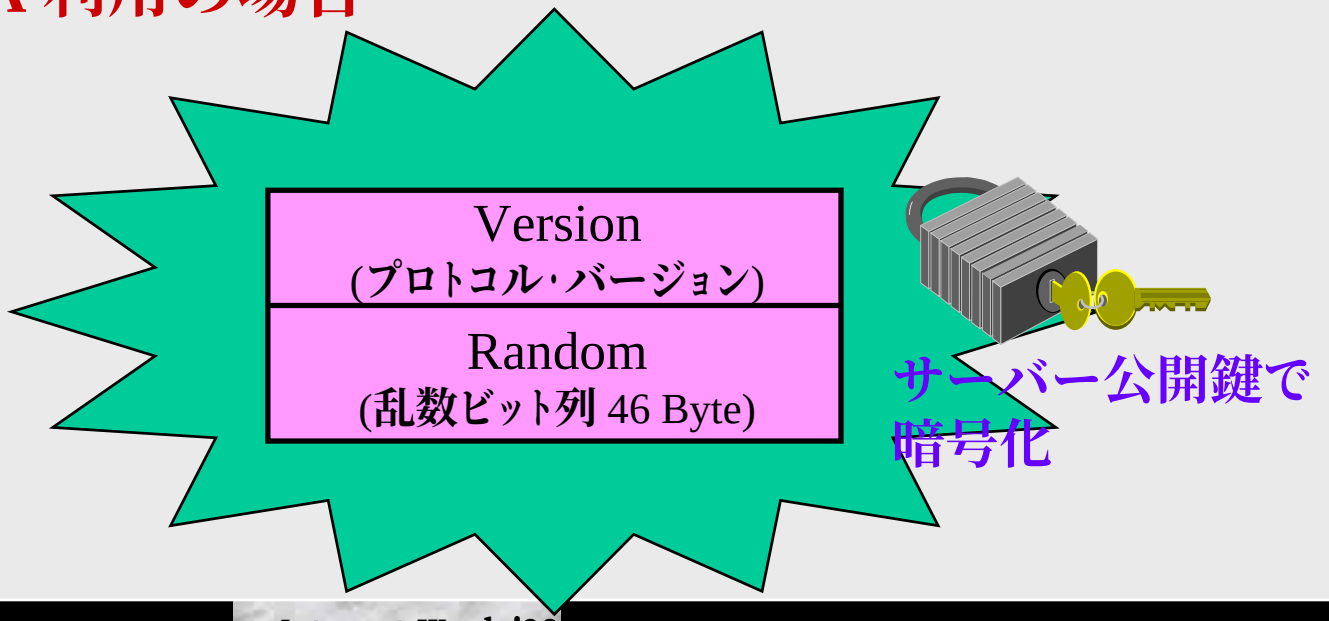
- C/S Hello
  - 暗号化／圧縮アルゴリズムの決定
  - 時間データを含み、接続のフレッシュさを保証

= Random
マスタ・シークレット生成に利用

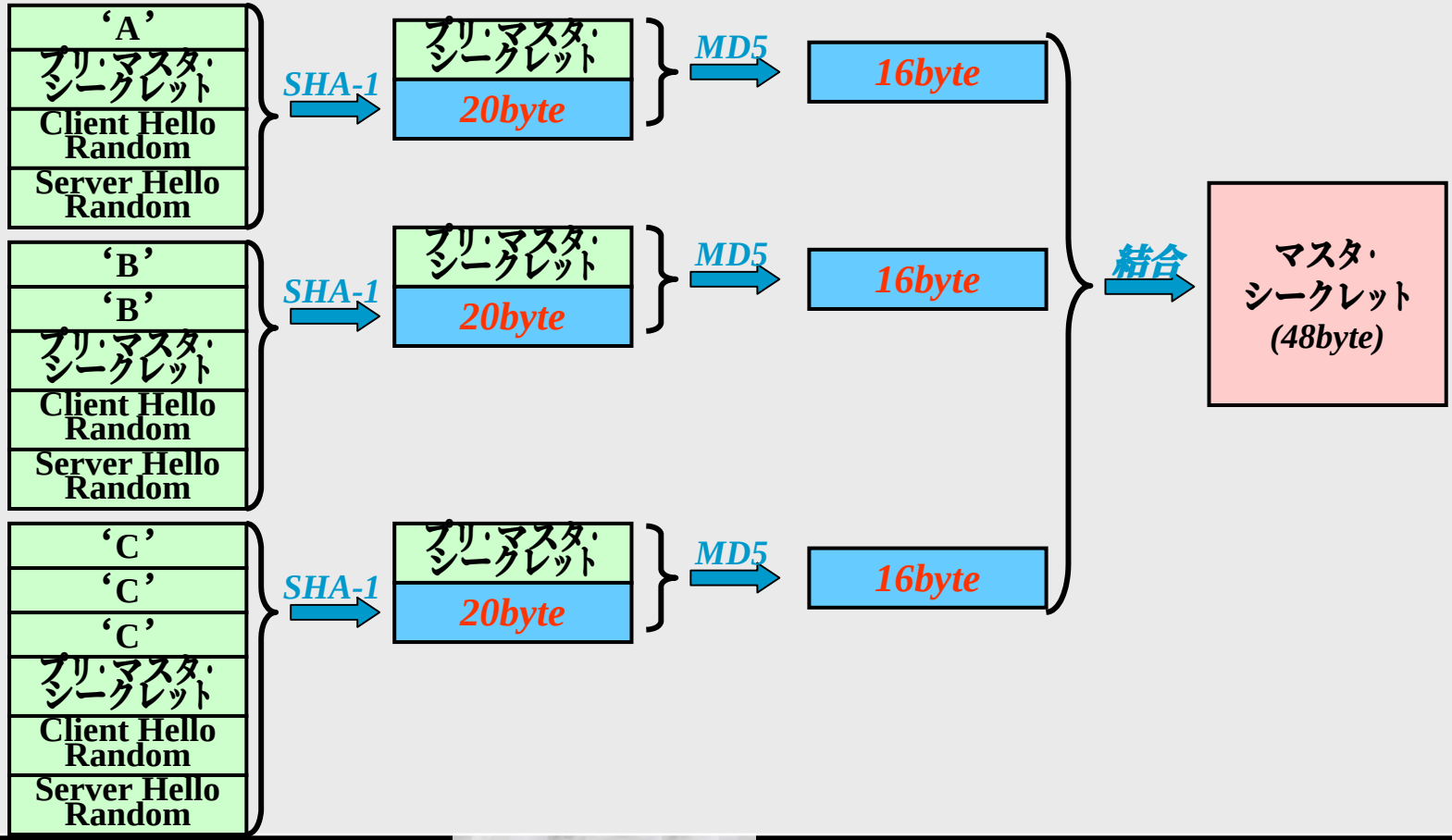
Client Key Exchange

- 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) をサーバーに送付

RSA 利用の場合



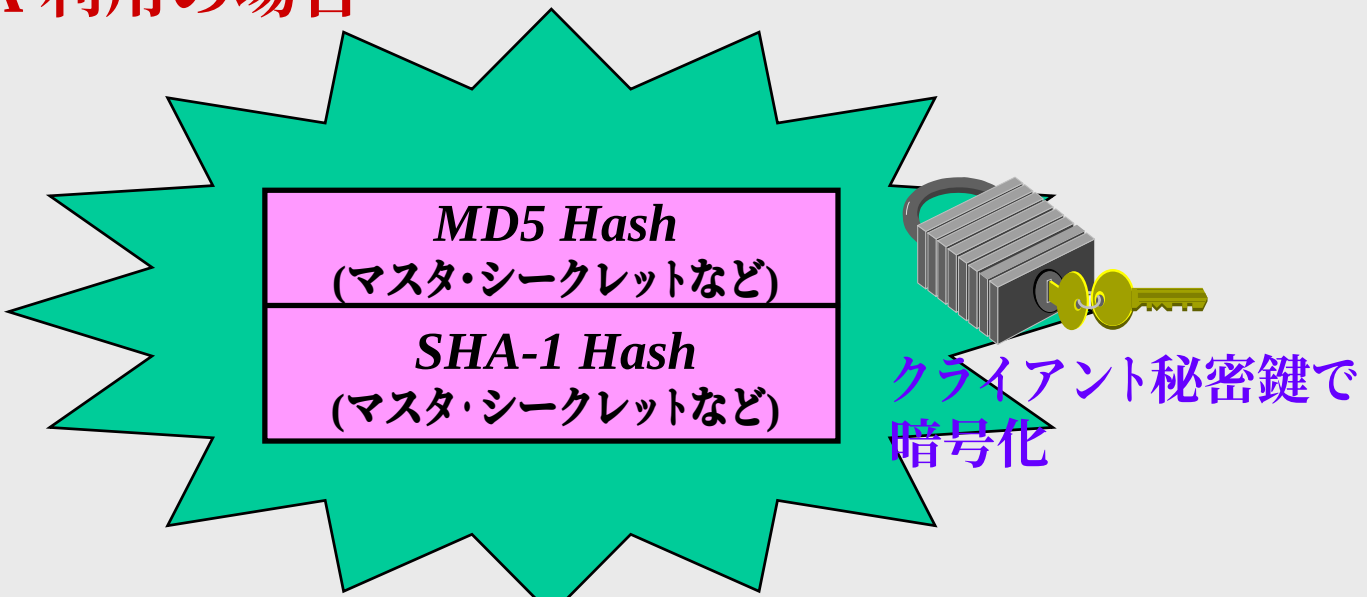
マスタ・シークレットの生成



Certificate Verify

- サーバーがクライアント証明書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

RSA 利用の場合



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

$$Key\_Block = MD5(MasterSecret + SHA(MasterSecret + ServerHelloRandom + ClientHelloRandom + 'A')) + MD5(MasterSecret + SHA(MasterSecret + ServerHelloRandom + ClientHelloRandom + 'BB')) + MD5(MasterSecret + SHA(MasterSecret + ServerHelloRandom + ClientHelloRandom + 'CCC')) + \cdots$$

+: 結合演算

- サポートする暗号化アルゴリズム等
  - 対称暗号
    - *RC2, RC4, IDEA, DES, 3DES, Fortezza*
  - 鍵交換
    - *RSA, Diffie-Hellman, Fortezza*
  - 一方向関数
    - *MD5, SHA-1*

- 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
  - 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

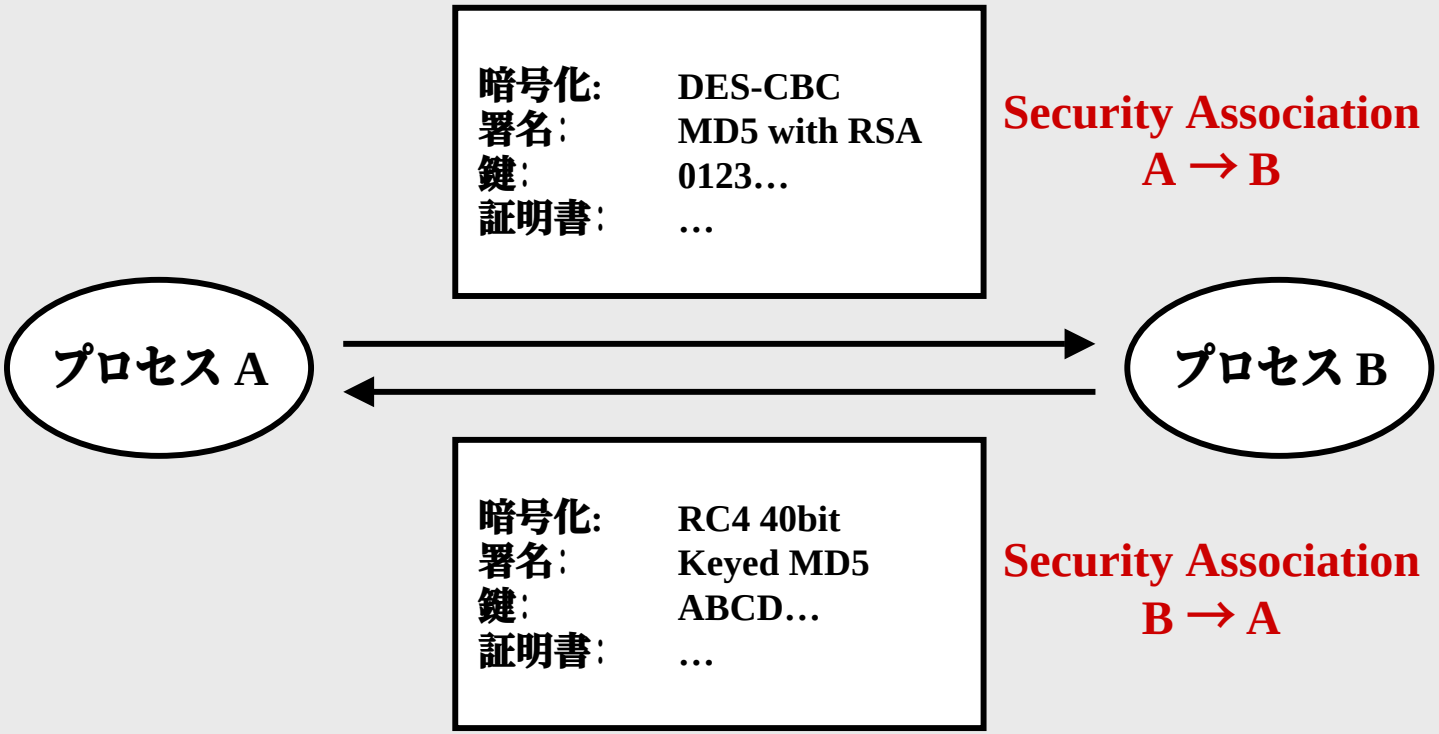
etc.

- *Internet Protocol* 自体へのセキュリティ機能付与
- 鍵管理方式との分離
- 認証と暗号化

- セキュリティ機能のない IP プロトコルとの共存が可能
  - 相互運用性 *(Interoperability)* の確保
- 現行 IP (IPv4) でも利用可能
  - 次世代 IP (IPv6) 向けのみではない

- 鍵管理方式は任意に利用可能
  - » 手動配布, *Diffie&Hellman*, *Kerberos*, etc.
- **Security Association** (SA) と **Security Parameters Index** (SPI) によって指定

■ Security Association (SA)



Security Parameters Index (SPI)

SPI for Host X

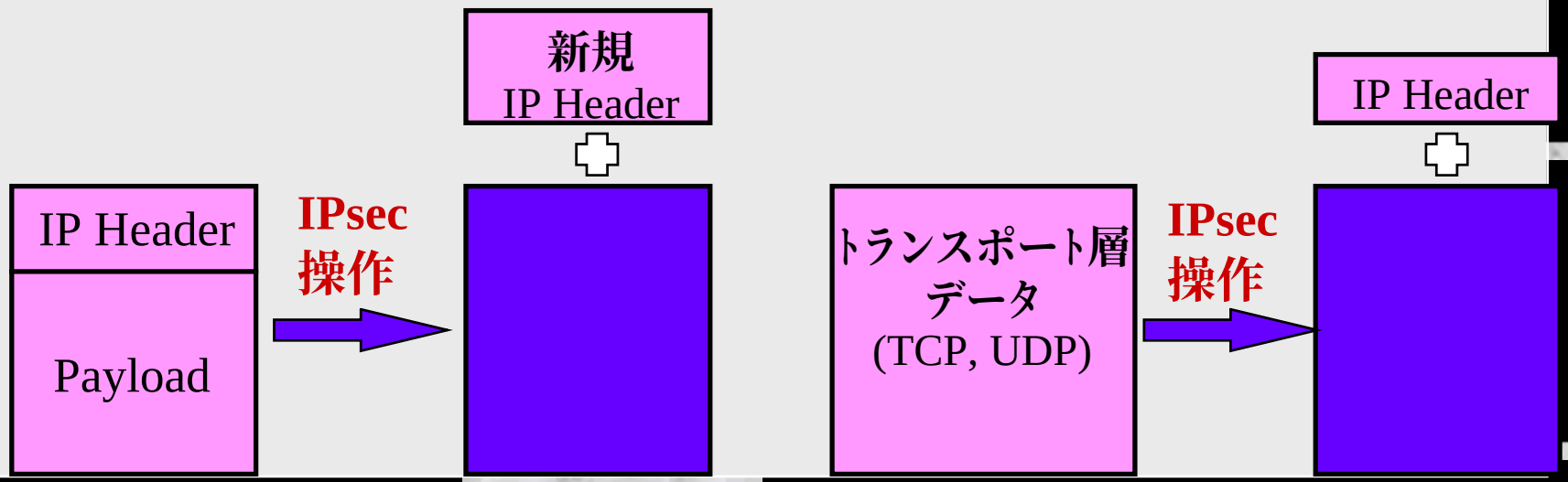
送信者	暗号方式	署名方式	鍵	…
Host A	DES-CBC	MD5 with RSA	0123	
Host B	IDEA-CBC	Keyed MD5	ABCD	
Host C	RC5-CFB	SHA with DSS	FECD	
…				
Host X	RC4	MD5 with RSA	5678	
…				

Security Association Table

- **Oakley**
  - *Diffie-Hellman* ベースの鍵配送プロトコル
- **ISAKMP**
  - *Internet Security Association and Key Management Protocol*
  - **Security Association** を構築･管理するためのメッセージフォーマットなどを定めたプロトコル

二つのモード

- トンネルモード
- トランスポートモード



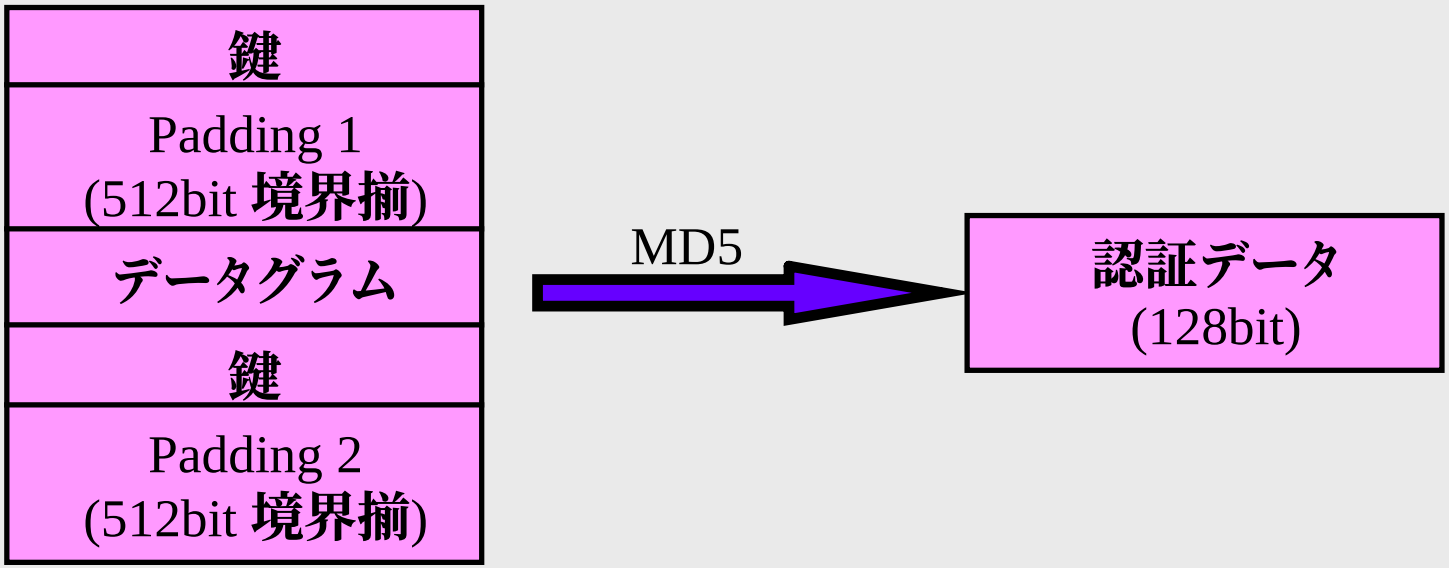
- 認証用データタイプ
  - IP Authentication Header (AH)
- 暗号化用データタイプ
  - IP Encapsulating Security Payload (ESP)
- と、RFC1825 では綺麗に分かれていたが…
  - 最新のモデルでは、ESP に認証機能を含められるようになってしまった

- IP データグラム全体から認証データ抽出
- 新しいヘッダタイプ (AH) の制定

AH 書式

Next Header	Length	RESERVED
Security Parameters Index (SPI)		
Sequence Number		
Authentication Data (可変長)		

Keyed MD5 方式による認証データ抽出



IPv4

IPv4 ヘッダ

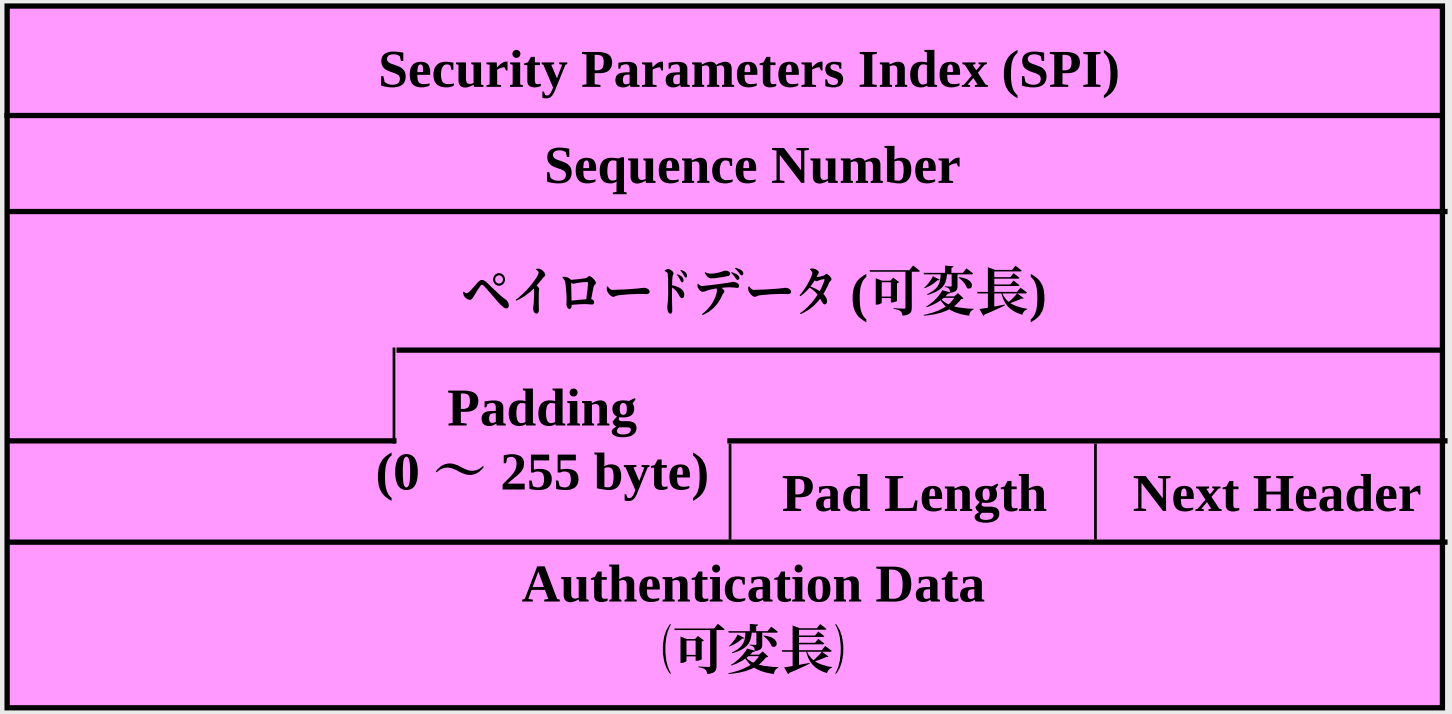
Version (= 4)	ヘッダ長	Type Of Service	Total Length
ID		フラグ	フラグメント・オフセット
Time To Live	Next Header (=51)	Header Checksum	
Source Address			
Destination Address			
Options (可変長)		Pad (0 ～ 24bit)	
Authentication Header (AH)			
上位プロトコルデータ			

IPv6

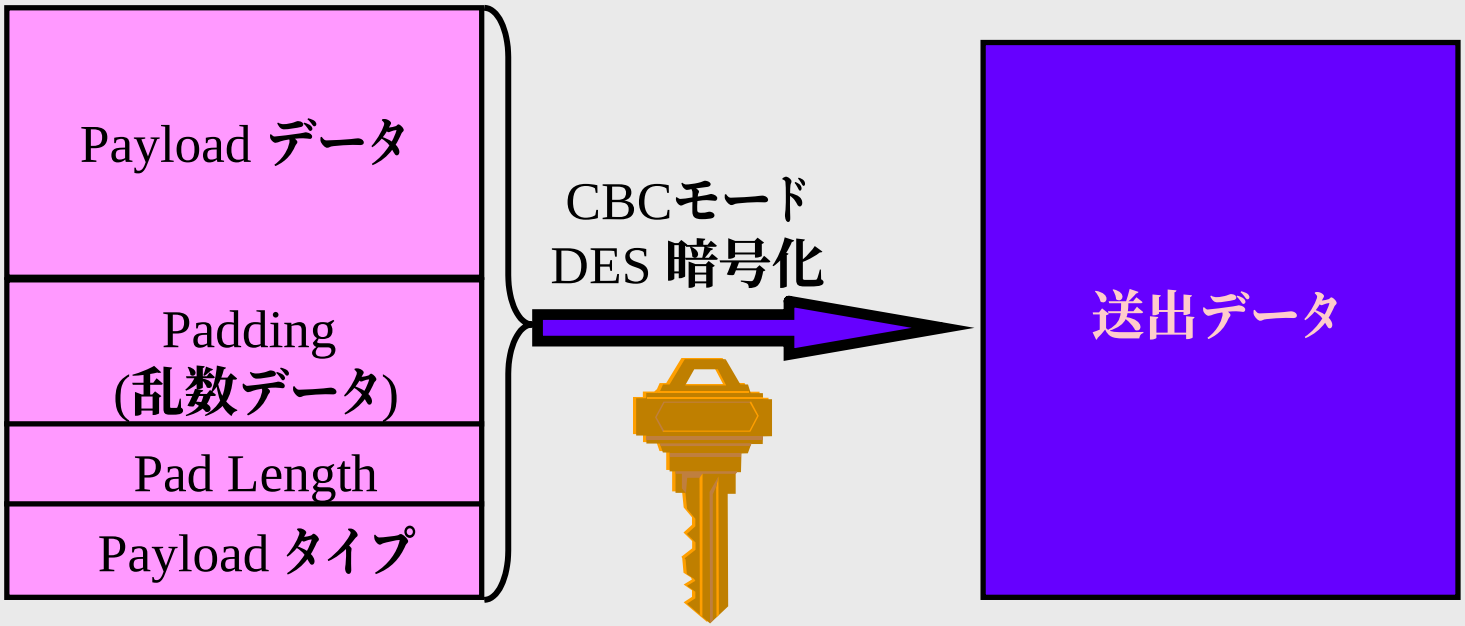
IPv6 ヘッダ

Version (= 6)	Prio	Flow Label	
Payload Length		Next Header (=51)	Hop Limit
Source Address (128bit)			
Destination Address (128bit)			
Authentication Header (AH)			
上位プロトコルデータ			

ESP 書式



DES CBC 方式による暗号化



IPv4

IPv4 ヘッダ

Version (= 4)	ヘッダ長	Type Of Service	Total Length
ID		フラグ	フラグメント・オフセット
Time To Live	Next Header (=50)	Header Checksum	
Source Address			
Destination Address			
Options (可変長)		Pad (0 〜 24bit)	
Encapsulating Security Payload (ESP)			

IPv6

IPv6 ヘッダ

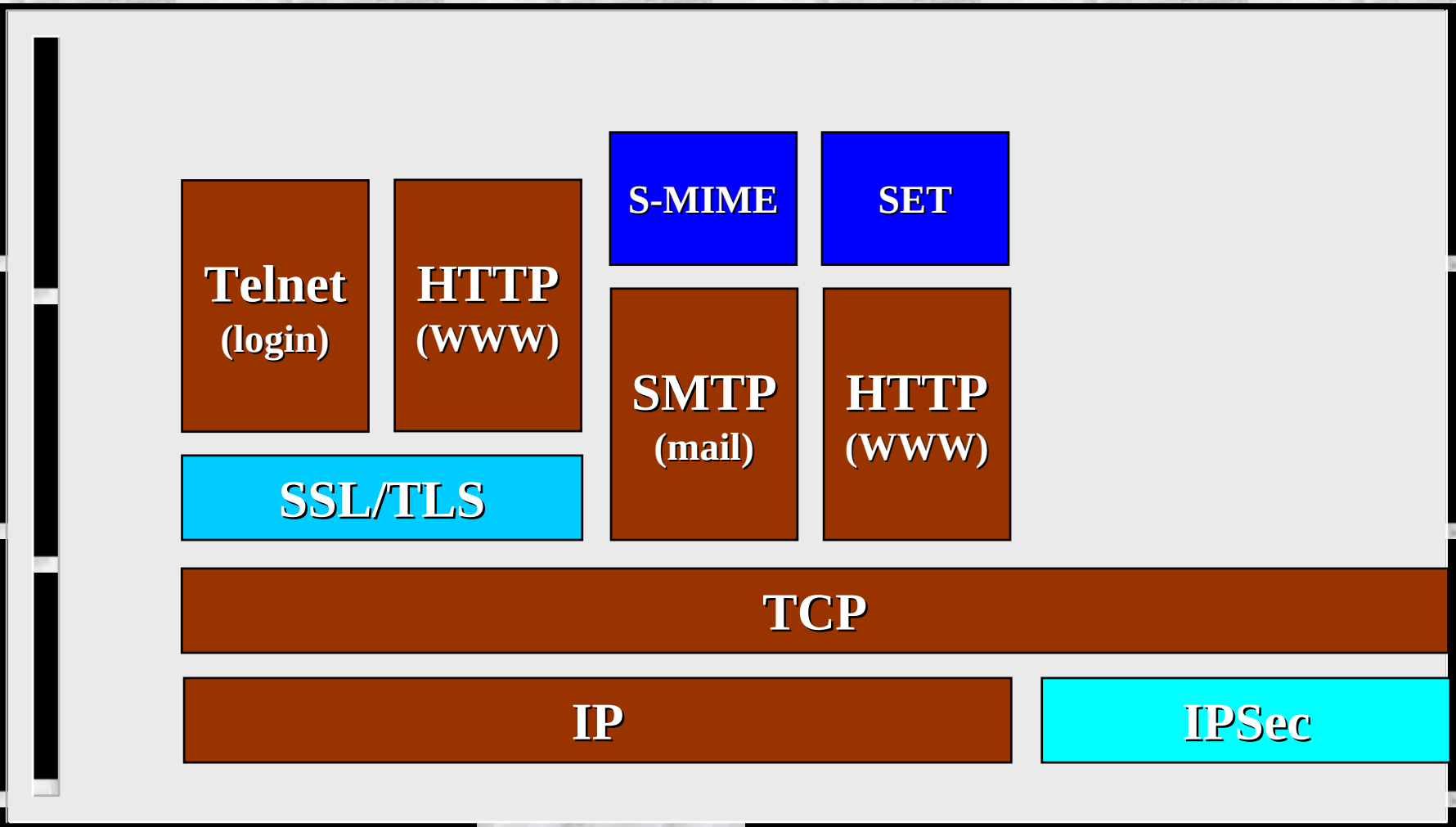
Version (= 6)	Prio	Flow Label	
Payload Length		Next Header (=50)	Hop Limit
Source Address (128bit)			
Destination Address (128bit)			
Encapsulating Security Payload (ESP)			

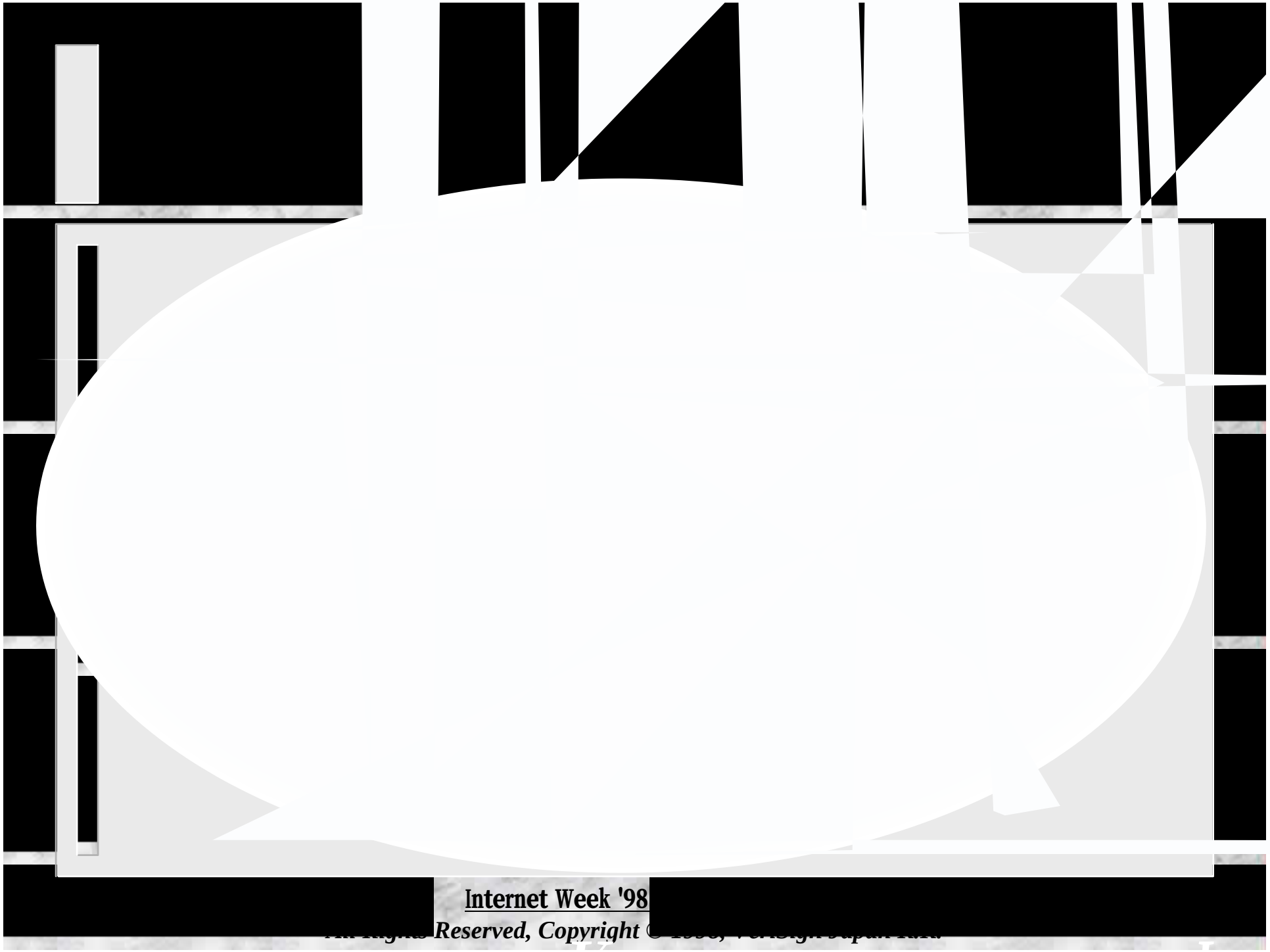
- インターネット全体での PKI 実現の試み
  - 証明書の内容
  - 証明書の管理･運用に関する各種プロトコル
  - ディレクトリ管理プロトコル

  etc.
- SPKI *(Simple Public Key Infrastructure)* などという WG もあり、今後どう動くかは不明だが…

PKIX

- *Certificate and CRL Profile*
- *Certificate Management Protocols*
- *Operational Protocols - LDAPv2*
- *Certificate Policy and Certification Practices Framework*
- *Representation of Key Exchange Algorithm (KEA) Keys in Internet X.509 Public Key Infrastructure Certificates*
- *Operational Protocols: FTP and HTTP*
- *Online Certificate Status Protocol - OCSP*
- *Representation of Elliptic Curve Digital Signature Algorithm (ECDSA) Keys and Signatures in Internet X.509 Public Key Infrastructure Certificates*
- *Certificate Request Message Format*
- *Certificate Management Message Formats*
- *Certificate Management Messages ov CMS*
- *LDAPv2 Schema*
- *Caching the Online Certificate Status Protocol*
- *WEB based Certificate Access Protoc - WebCAP/1.0 (56291 bytes)*
- *ENHANCED CRL DISTRIBUTION OPTIONS*
- *Time Stamp Protocols*
- *Data Certification Server Protocols*
- *PKIX Roadmap*





Internet Week '98