# 第7章　アクセスルール処理と認証を融合するプロトコルの提案

## 7.1 アクセスルールの完全性

### 7.1.1 プロトコルの定義

これまでに見た Unlink-Verify, Link-Verify, Unlink-Authenticate-SA, Link-Authenticate-SA, Unlink-Key-Transfer, Link-Key-Transfer の全てのプロトコルにおいて、アクセスルールの完全性は保証されている。

アクセスルールは、*aid* の発行時に *SP* が生成するが、その暗号学的ハッシュ等を $*$ 認証子に含める。従って、アクセスルールの完全性は、$*$ 認証子の完全性に帰着される。

### 7.1.2 安全性の評価

**定理 15.** *Unlink-Verify, Link-Verify, Unlink-Authenticate-SA, Link-Authenticate-SA, Unlink-Key-Transfer, Link-Key-Transfer* の各プロトコルにおいて、*SA* は $*$ 認証子の改竄を検知できる。

**証明** 以下では、*SA* が

$$rG_S \stackrel{?}{=} \omega(\pi(W) \mid c \mid *)(S - anmG_S) + W \tag{7.1}$$

の検証に成功すれば、$*$ が *UA* にとって Authentic な *SpA* に帰属することを証明する。

定理 9 の証明で示したように、*UA* にとって Authentic な *SpA* が存在し、$c \mid *$ に対して署名を生成している。

まず、*SpA* が Authentic であることから、*SpA* に入力された $*$ は *aid* に関連して *SP* が生成した $*$ であることが分かる。

更に、*SpA* は、入力された $*$ に対して署名を生成して返すので、*SA* による検証式 (7.1) が成立すれば、検証式中の $*$ は Authentic な *SpA* が計算に用いた $*$ に一致することが示される。 □

## 7.2 アクセスルールの認証付き更新

### 7.2.1 プロトコルの定義

権利に付随するアクセスルールは、自律的に自身を変化させるルールを含む可能性がある。例えば、回数券のように同権利を根拠として実行できる回数の上限を定める場合には、アクセスルールには上限値が指定されるが、権利を行使するたびに上限値を減算し、アクセスルールを更新する必要がある。

上記の要求に基づくアクセスルールの更新をサポートするために、Unlink-Rights-Update と Link-Rights-Update の 2 組のプロトコルを以下に定める。これらのプロトコルでは、信頼された *SA* が、*SP* に代わってアクセスルールを更新し、更に、*SpA* と協力してアクセスルールの更新に即して *aid* を更新する。

Unlink-Rights-Update と Link-Rights-Update は、それぞれ、Unlink-Verify と Link-Verify をベースとして、以下の処理を付加的に実行する。

1. *SA* は更新する新たなアクセスルールを生成し、それに伴って新たな $*$ 認証子 $\bar{*}$ に反映させる。例えば、新たなアクセスルールの暗号学的ハッシュを計算し、$\bar{*}$ に指定する。
2. *SpA* は *SA* を認証する。特に、*SA* がアクセスルールを更新する権限を有することを認証し、$c \mid * \mid \bar{*}$ に署名し Response とする。
3. *SA* は、Response を含む *UA* からの出力を検証し、検証が成功した場合は、*SpA* に対して、$e_1 = \mu(key, 1)$ を送付する。$e_1$ は、*SpA* が検証可能な MAC である。
4. *SpA* は $e_1$ の検証に成功した場合に限り、以下の手順を実行する。
   (a) $k$ と同じ長さを有する乱数 $\bar{k}$ を新たに生成する。
   (b) 現在処理中の鍵レコードに指定される $k$ を $\bar{k}$ で置き換える。
   (c) 下式で $\Delta$ を計算する。

   $$\Delta = \mu(k, *) - \mu(\bar{k}, \bar{*})$$

   (d) 更に、$\Delta$ への署名を更新した $\mu(\bar{k}, \bar{*})$ に基づいて計算する。生成する署名は、追跡不能認証・追跡可能認証の別により、$t'$ 或いは $t$ のいずれかとなる。

   $$t' = \omega(\Delta \mid \pi(W) \mid c)\mu(\bar{k}, \bar{*}) + w' + w'' \bmod n_S$$

   $$t = \omega(\Delta \mid \pi(W) \mid c)\big(\mu(\bar{k}, \bar{*}) + \rho\big) + w' + w'' \bmod n_S$$

   $t'$ 或いは $t$ の生成では、$\pi(W)$ をメッセージの途中に挿入して、暗号学的ハッシュを計算する。即ち、$r$ 或いは $s$ を計算した場合の署名スキームと異なる第三の署名スキームに従って、$t'$ 或いは $t$ を計算する。
   (e) $(\Delta, t')$ 或いは $(\Delta, t)$ を *UA* に送付する。
5. *UA* は、追跡不能性或いは Consensual Disclosure を満足することを目的に $(\Delta, t')$ 或いは $(\Delta, t)$ を下式により検証する。

   $$t'G_S \stackrel{G_S}{=} \omega(\Delta \mid \pi(W) \mid c)\big(S - (aid + \Delta)G_S\big) + W$$

$$tG_S \overset{G_S}{=} \omega(\Delta \mid \pi(W) \mid c)\big(S - (anm + \Delta)G_S\big) + W$$

そして、検証が成功した場合に限り、現在処理中の権利レコードに指定される *aid* を $aid + \Delta \bmod n_S$ で置き換える。必要であれば、署名 $t'$ は

$$t = t' + \omega(\Delta \mid \pi(W) \mid c)\rho \bmod n_S$$

により変換し、$(\Delta, t)$ を *SA* に送付する。

6. *SA* は、$(\Delta, t)$ を下式により検証することで、*SpA* で正しく更新処理が行われたことを確認する。

$$tG_S \overset{G_S}{=} \omega(\Delta \mid \pi(W) \mid c)\big(S - (anm + \Delta)G_S\big) + W$$

#### 7.2.1.1 Unlink-Rights-Update の定義

| *SA* | | *UA* | | *SpA* |
|---|---|---|---|---|
| | | | | $w', z' \in_{\mathcal{R}} [0, n_S)$ |
| | | | $\xleftarrow{W'}$ | $W' \overset{G_S}{=} w'G_S,\ Z' \overset{G_S}{=} z'G_S$ |
| | | $\rho, w'', z'' \in_{\mathcal{R}} [0, n_S)$ | | |
| | | $anm = aid - \rho \bmod n_S$ | | |
| | | $W \overset{G_S}{=} w''G_S + W'$ | | |
| | | $Z \overset{G_S}{=} z''G_S + Z'$ | | |
| | $\xleftarrow{W, anm}$ | | | |
| $c \in_{\mathcal{R}} [0, n_S)$ | | | | |
| $key = \omega(\pi(\alpha W)\|c)$ | | | | |
| $e_1 = \omega(key)$ | $\xrightarrow{c,*,\bar{*}}$ | | $\xrightarrow{c,*,\bar{*},w'',z''}$ | $key = \omega(\pi((w' + w'')A)\|c)$ |
| | | | | Unless $e_1$ is valid, abort the session. |
| | | | | $a = \omega(\pi((w' + w'')G_S) \mid c \mid * \mid \bar{*})$ |
| | | | $\xleftarrow{r}$ | $r' = a\mu(k, *) + w' + w'' \bmod n_S$ |
| | | Unless $(W, r')$ is valid, abort the session. | | |
| | | $a = \omega(\pi(W) \mid c \mid * \mid \bar{*})$ | | |
| | | $r = r' + a\rho \bmod n_S$ | | |
| Unless $(W, r)$ is valid, abort the session. | $\xleftarrow{r}$ | | | |
| $e_2 = \mu(key, 1)$ | $\xrightarrow{e_2}$ | $\longrightarrow$ | $\xrightarrow{e_2}$ | Unless $e_2$ is valid, reject the session. |
| | | | | Replace $k$ with new random $\bar{k}$. |
| | | | | $\Delta = \mu(k, *) - \mu(\bar{k}, \bar{*}) \bmod n_S$ |
| | | | | $d = \omega(* \mid \bar{*} \mid \pi((z' + z'')G_S) \mid c)$ |
| | | | $\xleftarrow{\Delta, t'}$ | $t' = d\mu(\bar{k}, \bar{*}) + z' + z'' \bmod n_S$ |
| | | Unless $\Delta, t'$ are valid, abort the session. | | |
| | | Replace $aid$ with $aid + \Delta$ | | |
| | | $d = \omega(* \mid \bar{*} \mid \pi(W) \mid c)$ | | |
| | | $\rho' \in_{\mathcal{R}} [0, n_S)$ | | |
| | | $anm' = aid + \rho'$ | | |
| | | $t = t' + d\rho' \bmod n_S$ | | |
| | $\xleftarrow{\Delta, t}$ | | | |
| Unless $anm', t$ are valid, abort the session. | | | | |

図 7.1: 追跡不能権利更新プロトコル: Unlink-Rights-Update

***UA* による検証式**

$$r'G_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid * \mid \bar{*})(S - aid \cdot G_S) + W \tag{7.2}$$

$$t'G_S \overset{G_S}{=} \omega(* \mid \bar{*} \mid \pi(Z) \mid r)\big(S - (aid + \Delta)G_S\big) + W \tag{7.3}$$

***SA* による検証式**

$$rG_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid *)(S - anm \cdot G_S) + W \tag{7.4}$$

$$tG_S \overset{G_S}{=} \omega(\Delta \mid \pi(Z) \mid r)\big(S - anm')G_S\big) + W \tag{7.5}$$

### 7.2.1.2 Link-Rights-Update の定義

| *SA* | *UA* | *SpA* |
|---|---|---|
| | | $w', q', z' \in_{\mathcal{R}} [0, n_S)$<br>$W' \overset{G_S}{=} w' G_S,\ Z' \overset{G_S}{=} z' G_S$ |
| | $\xleftarrow{W', Q', Z'}$ | $Q' \overset{G_S}{=} q' G_S$ |
| | $\rho, w'', q'', z'', x, y \in_{\mathcal{R}} [0, n_S)$<br>$anm = aid - \rho \bmod n_S$<br>$W \overset{G_S}{=} w'' G_S + W'$<br>$Q \overset{G_S}{=} q'' G_S + Q'$<br>$Z \overset{G_S}{=} z'' G_S + Z'$<br>$U \overset{G_S}{=} xG_S + yQ$ | |
| $c \in_{\mathcal{R}} [0, n_S)$<br>$key = \omega(\pi(\alpha W)\|c)$ | $\xleftarrow{anm, W}$ | |
| $e_1 = \omega(key)$ | $\xrightarrow{c, *, \bar{*}}$ | $\xrightarrow{c, *, \bar{*}, w''}$ $key = \omega(\pi((w' + w'')A)\|c)$ |
| | | $\xrightarrow{q'', U, \rho}$ Unless $e_1$ is valid, abort the session.<br>$a = \omega(\pi((w' + w'')G_S) \mid c \mid * \mid \bar{*})$<br>$r = a(\mu(k, *) + \rho) + w' + w'' \bmod n_S$<br>$Q \overset{G_S}{=} q'' G_S + Q'$<br>$e_P = \pi((\mu(k, *) + \rho)Q) \oplus \rho$<br>$V \overset{G_S}{=} (\mu(k, *) + \rho)U$<br>$b = \omega(r \mid e_P \mid \pi(Q))$ |
| | $\xleftarrow{r, s, e_P, V}$ | $s = b(\mu(k, *) + \rho) + q' + q'' \bmod n_S$ |
| | Unless $W, r, Q, s, e_P, V$ are valid, abort the session. | |
| Unless $W, r, Q, s, e_P$ are valid, abort the session. | $\xleftarrow{r, Q, s, e_P}$ | |
| $e_2 = \mu(key, 1)$ | $\xrightarrow{e_2}$ $\longrightarrow$ | $\xrightarrow{e_2}$ Unless $e_2$ is valid, reject the session.<br>Replace $k$ with new random $\bar{k}$.<br>$\Delta = \mu(k, *) - \mu(\bar{k}, \bar{*}) \bmod n_S$<br>$d = \omega(* \mid \bar{*} \mid \pi((z' + z'')G_S) \mid c)$ |
| | | $\xleftarrow{\Delta, t'}$ $t' = d\mu(\bar{k}, \bar{*}) + z' + z'' \bmod n_S$ |
| | Unless $\Delta, t'$ are valid, abort the session.<br>Replace $aid$ with $aid + \Delta$<br>$d = \omega(* \mid \bar{*} \mid \pi(W) \mid c)$<br>$\rho' \in_{\mathcal{R}} [0, n_S)$<br>$anm' = aid + \rho'$<br>$t = t' + d\rho' \bmod n_S$ | |
| Unless $anm', t$ are valid, abort the session. | $\xleftarrow{anm', t}$ | |

図 7.2: 追跡可能権利更新プロトコル: Link-Rights-Update

#### *UA* による検証式

$$rG_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid *)\,(S - anm \cdot G_S) + W \tag{7.6}$$

$$sG_S \overset{G_S}{=} \omega(r \mid e_P \mid \pi(Q))\,(S - anm \cdot G_S) + Q \tag{7.7}$$

$$V \overset{G_S}{=} x(S - anm \cdot G_S) + y\pi^{-1}(e_P \oplus \rho) \tag{7.8}$$

$$t'G_S \overset{G_S}{=} \omega(* \mid \bar{*} \mid \pi(Z) \mid r)\big(S - (aid + \Delta)G_S\big) + W \tag{7.9}$$

***SA* による検証式** (7.6)、(7.7)、及び、(7.10)

$$tG_S \stackrel{G_S}{=} \omega(* \mid \bar{*} \mid \pi(Z) \mid r)\big(S - anm' \cdot G_S\big) + W \tag{7.10}$$

***SP* による *aid* の復号** *SP* は、*SA* から $(c, *, anm, W, r, Q, s, e_P)$ を受け取り、以下の手順で *aid* を復号する。

1. 検証式 (5.5) 及び (5.6) を検査する。
2. 下式により $\rho$ 及び *aid* を計算する。

$$\rho = \pi((\sigma - aaid)Q) \oplus e_P$$

$$aid = anm + \rho \bmod n_S$$

### 7.2.2 安全性の検証

Unlink-Rights-Update 及び Link-Rights-Update の完全性の証明は容易である。

また、Unlink-Rights-Update 及び Link-Rights-Update の前半部分は、それぞれ、Unlink-Verify 及び Link-Verify にそれぞれ同一であるので、プロトコルの健全性は、Unlink-Verify 及び Link-Verify から継承する。

また、Unlink-Rights-Update 及び Link-Rights-Update の後半部分では、署名 $(anm', Z, t)$ を出力しているが、$z'', \rho' \in [0, n_S)$ を一様にランダムに選択しているため、$(anm', Z, t)$ の分布は一様であり、かつ、$(anm, W, r, Q, s, e_P)$ とは独立である。従って、Unlink-Rights-Update の追跡不能性と、Link-Rights-Update の Consensual Disclosure のサポートは、それぞれ、Unlink-Verify と Link-Verify の対応する性質に帰着される。

従って、Unlink-Rights-Update 及び Link-Rights-Update の安全性は、以下の定理 16 及び定理 17 によって示される。

**定理 16.** *Unlink-Rights-Update* 及び *Link-Rights-Update* において、*SP* は更新した $*$ 認証子 $\bar{*}$ の改竄を検知できる。

**証明** 以下では、*SA* が

$$rG_S \overset{?_S}{=} \omega(\pi(W) \mid c \mid * \mid \bar{*})(S - anmG_S) + W \tag{7.11}$$

の検証に成功すれば、*SA* が送付した $\bar{*}$ を *UA* にとって Authentic な *SpA* が受け取っていることを示す。

定理 9 の証明で示したように、*UA* にとって Authentic な *SpA* が存在し、$c \mid * \mid \bar{*}$ に対して署名を生成している。

*SpA* は、入力された L に対して署名を生成して返すので、*SA* による検証式 (7.11) が成立すれば、検証式中の $\bar{*}$、即ち、*SA* が *UA* に送付した $\bar{*}$ は、Authentic な *SpA* が計算に用いた $\bar{*}$ に一致することが示される。 □

**定理 17.** *Unlink-Rights-Update* 及び *Link-Rights-Update* において、*SA* が認証に成功する限りにおいては、*Authentic* な *SpA* が正しく更新処理を実行している。

**証明** *SA* は $(anm', Z, t)$ の検証に成功しているので、定理 9 の証明で示したように、*UA* にとって Authentic な *SpA* が存在して $* \mid \bar{*}, c$ に対して署名を生成している。署名 $(anm', Z, t)$ の署名は、$(anm, W, r)$ や $(anm, Q, s)$ の署名とは異なる署名スキームに基づいているので、Authentic な *SpA* は、$(c, *, \bar{*})$ を受け取って、更新処理を実行したことが分かる。 □

### 7.2.3 計算量の評価

Unlink-Rights-Update 及び Link-Rights-Update の計算量を表 7.1 に示す。

表 7.1: Unlink-Rights-Update 及び Link-Rights-Update の計算量

| | *SA* | *UA* | *SpA* | 合計 |
|---|---|---|---|---|
| Unlink-Rights-Update | 5 | 6 | 4 | 15 |
| Link-Rights-Update | 7 | 14 | 8 | 29 |

(楕円曲線上のスカラー倍演算の実行回数)

## 7.3 権限の無効化

### 7.3.1 プロトコルの定義

この節で提案するプロトコルは、4.5.1 で述べた 2 種の無効化のうち、Named-revocation を実現するものである。

権限の無効化では、*SP* が発行した権利無効化リスト (RRL) に従って、*SA* と *SpA* が協力して発行済みの権利を無効化する。権利無効化リストは、#識別子のリストであり、以下のように処理される。

- *SA* は *SP* から権利無効化リストを受け取り、*SP* が正しく生成した権利無効化リストであり、かつ、改竄されていないことを、署名等の手段を用いて検証する。

  権利無効化リストの形式及び検証の方法は、本プロトコルでは規定せず、*SP* と *SA* の間の合意に依存するものとする。

- *SA* は、権利無効化リストを *SpA* に送付する。

- *SpA* は現在のセッションで使用している鍵レコード中の#識別子が権利無効化リストに記載されている場合は、その鍵レコードを削除する。

- *SA* は、レスポンスの検証の成否によらず、サービスの提供は行わない。

Unlink-Revoke 及び Link-Revoke の定義は、それぞれ、図 7.3 及び図 7.4 で与えられる。

#### 7.3.1.1 Unlink-Revoke の定義

| *SA* | *UA* | *SpA* |
|---|---|---|
| | | $w' \in_{\mathcal{R}} [0, n_S)$ |
| | $\xleftarrow{W'}$ | $W' \overset{G_S}{=} w' G_S$ |
| | $\rho \in_{\mathcal{R}} [0, n_S)$<br>$anm = aid - \rho \bmod n_S$<br>$w'' \in_{\mathcal{R}} [0, n_S)$<br>$W \overset{G_S}{=} w'' G_S + W'$ | |
| | $\xleftarrow{W, anm}$ | |
| $c \in_{\mathcal{R}} [0, n_S)$<br>$key = \omega(\pi(\alpha W)\|c)$<br>$e_1 = \omega(key)$ | | |
| | $\xrightarrow{c, *, \mathrm{RRL}}$ | |
| | $\xrightarrow{c, *, \mathrm{RRL}, w''}$ | |
| | | $key = \omega(\pi((w' + w'')A)\|c)$<br>Unless $e_1$ is valid, abort the session.<br>$W \overset{G_S}{=} w'' G_S + W'$<br>$a = \omega(\pi(W) \mid c \mid *)$<br>$r' = a\mu(k, *) + w' + w'' \bmod n_S$<br>If RRL contains the relevant #-identifier, erase the relevant key record. |
| | $\xleftarrow{r'}$ | |
| | Unless $(W, r')$ is valid, abort the session.<br>$a = \omega(\pi(W) \mid c \mid * \mid \mathrm{RRL})$<br>$r = r' + a\rho \bmod n_S$ | |
| | $\xleftarrow{r}$ | |
| Unless $(W, r)$ is valid, abort the session. | | |

図 7.3: 追跡不能無効化プロトコル: Unlink-Revoke

***UA* による検証式**

$$r' G_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid *)(S - aid \cdot G_S) + W \tag{7.12}$$

***SA* による検証式**

$$r G_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid *)(S - anm \cdot G_S) + W \tag{7.13}$$

#### 7.3.1.2 Link-Revoke の定義

| *SA* | *UA* | *SpA* |
|---|---|---|
| | | $w', q' \in_{\mathcal{R}} [0, n_S)$<br>$W' \overset{G_S}{=} w'G_S$<br>$Q' \overset{G_S}{=} q'G_S$ |
| | $\xleftarrow{W', U'}$ | |
| | $\rho, w'', q'', x, y \in_{\mathcal{R}} [0, n_S)$<br>$anm = aid - \rho \bmod n_S$<br>$W \overset{G_S}{=} w''G_S + W'$<br>$Q \overset{G_S}{=} q''G_S + Q'$<br>$U \overset{G_S}{=} xG_S + yQ$ | |
| | $\xleftarrow{anm, W}$ | |
| $c \in_{\mathcal{R}} [0, n_S)$<br>$key = \omega(\pi(\alpha W)\|c)$<br>$e_1 = \omega(key)$ | | |
| | $\xrightarrow{c, *, \mathrm{RRL}}$ | |
| | $\xrightarrow{c, *, \mathrm{RRL}, w''}$ | |
| | $\xrightarrow{q'', U, \rho}$ | |
| | | $key = \omega(\pi((w' + w'')A)\|c)$<br>Unless $e_1$ is valid, abort the session.<br>$W \overset{G_S}{=} w''G_S + W'$<br>$a = \omega(\pi(W) \mid c \mid * \mid \mathrm{RRL})$<br>$r = a(\mu(k, *) + \rho) + w' + w'' \bmod n_S$<br>$Q \overset{G_S}{=} q''G_S + Q'$<br>$e_P = \pi((\mu(k, *) + \rho)Q) \oplus \rho$<br>$V \overset{G_S}{=} (\mu(k, *) + \rho)U$<br>$b = \omega(r \mid e_P \mid \pi(Q))$<br>$s = b(\mu(k, *) + \rho) + q' + q'' \bmod n_S$<br>If RRL contains the relevant #-identifier, erase the relevant key record. |
| | $\xleftarrow{r, s, e_P, V}$ | |
| | Unless $W, r, Q, s, e_P, V$ are valid, abort the session. | |
| Unless $(W, r, Q, s, e_P)$ is valid, abort the session. | $\xleftarrow{r, Q, s, e_P}$ | |

図 7.4: 追跡可能無効化プロトコル: Link-Revoke

***UA* による検証式**

$$rG_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid *)\,(S - anm \cdot G_S) + W \tag{7.14}$$

$$sG_S \overset{G_S}{=} \omega(r \mid e_P \mid \pi(Q))\,(S - anm \cdot G_S) + Q \tag{7.15}$$

$$V \overset{G_S}{=} x(S - anm \cdot G_S) + y \cdot \pi^{-1}(e_P \oplus \rho) \tag{7.16}$$

***SA* による検証式** (7.14) 及び (7.15)

***SP* による *aid* の復号** *SP* は、*SA* から $(c, *, anm, W, r, Q, s, e_P)$ を受け取り、以下の手順で *aid* を復号する。

1. 検証式 (7.14) 及び (7.15) を検査する。

***SP* による *aid* の復号** *SP* は、*SA* から $(c, *, anm, W, r, Q, s, e_P)$ を受け取り、以下の手順で *aid* を復号する。

1. 検証式 (5.5) 及び (5.6) を検査する。
2. 下式により $\rho$ 及び *aid* を計算する。

$$\rho = \pi((\sigma - aaid)Q) \oplus e_P$$

$$aid = anm + \rho \bmod n_S$$

### 7.3.2 安全性の評価

Unlink-Revoke、及び、Link-Revokeは、RRLに関する処理を除くと、それぞれ、Unlink-Authenticate-SA及び、Link-Authenticate-SAと同一である。

従って、Unlink-Revoke、及び、Link-Revokeに関する安全性として示さなければならない事実は、*SA*が*UA*に入力したRRLが、改竄されることなく*SpA*に入力されていることである。

実際、定理18が成立する。

**定理 18.** *Unlink-Revoke*、及び、*Link-Revoke*において、*SA*は権利無効化リスト*(RRL)*の改竄を検知できる。

**証明** 以下では、*SA*が

$$rG_S \stackrel{?}{=} \omega(\pi(W) \mid c \mid * \mid \mathrm{RRL})(S - anmG_S) + W \tag{7.17}$$

の検証に成功すれば、*SA*が送付したRRLを*UA*にとってAuthenticな*SpA*が受け取っていることを示す。

定理9の証明で示したように、*UA*にとってAuthenticな*SpA*が存在し、$c \mid * \mid \mathrm{RRL}$に対して署名を生成している。

*SpA*は、入力されたRRLに対して署名を生成して返すので、*SA*による検証式(7.17)が成立すれば、検証式中のRRL、即ち、*SA*が*UA*に送付したRRLは、Authenticな*SpA*が計算に用いたRRLに一致することが示される。 □

### 7.3.3 計算量の評価

Unlink-Revoke及びLink-Revokeの計算量を表7.2に示す。

表 7.2: Unlink-Revoke及びLink-Revokeの計算量

| | *SA* | *UA* | *SpA* | 合計 |
|---|---|---|---|---|
| Unlink-Revoke | 2 | 3 | 2 | 7 |
| Link-Revoke | 4 | 11 | 6 | 21 |

(楕円曲線上のスカラー倍演算の実行回数)

## 7.4 権限の譲渡

### 7.4.1 プロトコルの定義

権限の譲渡は、譲渡元ユーザの *UA* と *SpA*、仲介を行う *SA*、及び、譲渡先ユーザの *UA* と *SpA* の 5 者間のプロトコルである点で、これまで見てきたプロトコルと相違する。

仲介を行う *SA* の役割は、譲渡を発行者の意図を反映して適切に実行する点にあり、特に以下の 2 項目の安全性を提供するためである。

- 譲渡元ユーザのアクセス権限に付随するアクセスルールを確認して、譲渡が許可されていることを検証する。
- オリジナルのアクセスルールに基づいて、譲渡先ユーザに許すアクセスルールを適正に作成する。

譲渡を、*SA* を介さずに譲渡元と譲渡先のユーザの間で直接に実行し、それでも、発行者の意図を反映させることも可能であるが、この場合には、ユーザの *SpA* においてアクセスルールを解釈させる必要がある。しかし、アクセスルールの解釈は、IC チップ等、処理能力が制限される実装が想定される *SpA* で実行するには、荷が重い。また、アクセスルールの記述文法としては、XrML [31] や OMA [53] 等、複数の標準が並行して開発されている事情から、アクセスルールの記述はプロトコルの規定から切り離しておきたい。以上の理由から、本論文で提案するプロトコルでは、仲介者である *SA* が、比較的重いアクセスルールの処理を行い、譲渡を適正に維持するように設計する。

仲介を行うエンティティを、権限の発行者である *SP* ではなく、*SA* とすることで、仲介者を仮定することによる不自由さを、実用上問題のない程度まで、緩和することができると考えている。例えば、ドメインにおいて、信頼のできる *SA* を取得して、ドメイン内から自由にアクセスできるように運用すればよく、場合によっては、*SA* のプログラムを各ユーザの端末にインストールすることも可能であろう。勿論、サービスが非常に重要である場合には、権限の発行者が *SA* として仲介を行うことにより、厳密な安全性を実現することもできる。

この節で提案するプロトコルは、譲渡先の *UA* 及び *SpA* と仲介役の *SA* との間のプロトコルである Delegate-Receive、譲渡元の *UA* 及び *SpA* と仲介役の *SA* との間のプロトコルである Unlink-Delegate(追跡不能) と Link-Delegate(追跡可能) とから、構成される。

#### 7.4.1.1 Delegate-Receive の定義

| *SA* | *UA* | *SpA* |
|---|---|---|
| | | $\epsilon_T \in_{\mathcal{R}} [0, n_T)$ |
| | $\xleftarrow{E_T}$ | $E_T \stackrel{\mathcal{G}_T}{=} \epsilon_T G_T.$ |
| | $\epsilon_U \in_{\mathcal{R}} [0, n_T)$ | |
| | $E_U \stackrel{\mathcal{G}_T}{=} \epsilon_U G_T + E_T$ | |
| | $\xleftarrow{E_U, T}$ | |
| Perform Unlink-Delegate or Link-Delegate. | | |
| | $\xrightarrow{anm, \bar{*}, E_P, e_T}$ | $\xrightarrow{E_P, e_T, \epsilon_U}$ |
| | | $E_U \stackrel{\mathcal{G}_T}{=} (\epsilon_U + \epsilon_T) G_T$ |
| | | $k' = \pi((\epsilon_U + \epsilon_T + \bar{E}_U \tau) E_P)$ |
| | | $(\rho + \Delta \bmod n_S) \mid \bar{\#} = e_T \oplus k'$ |
| | | $k = \mu(k', \bar{\#})$ |
| | | Store $k$ and $\bar{\#}$. |
| | | $\xleftarrow{\rho+\Delta}$ |
| | $aid = anm + \rho + \Delta \bmod n_S$ | |
| | Store *aid* and $\bar{*}$ | |

図 7.5: 譲渡受領プロトコル: Delegate-Receive

### 7.4.1.2 Unlink-Delegate の定義

| *SA* | *UA* | | *SpA* |
|---|---|---|---|
| | $\xrightarrow{E_U,T,*,\bar{*}}$ Authenticate $T$. | $\xrightarrow{E_U,T}$ | $w', q' \in_{\mathcal{R}} [0, n_S)$, $\epsilon'_P \in [0, n_T)$ |
| | | | $W' \overset{G_S}{=} w'G_S$ |
| | | | $Q' \overset{G_S}{=} q'G_S$ |
| | | $\xleftarrow{W',Q',E'_P}$ | $E'_P \overset{G_T}{=} \epsilon_T G_T$ |
| | $\epsilon''_P, \xi, \eta \in_{\mathcal{R}} [0, n_T)$ | | |
| | $E_P \overset{G_T}{=} \epsilon''_P G_T + E'_P$ | | |
| | $\Upsilon \overset{G_T}{=} \xi G_T + \eta(E_U + \bar{E}_U T)$ | | |
| | | $\xrightarrow{\epsilon''_P,\Upsilon,*,\bar{*},\bar{\#}}$ | $k' = \pi((\epsilon'_P + \epsilon''_P)(E_U + \bar{E}_U T))$ |
| | | $\xleftarrow{\Delta}$ | $\Delta = \mu(k, *) - \mu(\mu(k', \bar{\#}), \bar{*}) \bmod n_S$ |
| | $\rho, w'', q'', x, y \in_{\mathcal{R}} [0, n_S)$ | | |
| | $anm = aid - \rho \bmod n_S$ | | |
| | $W \overset{G_S}{=} w''G_S + W'$ | | |
| | $Q \overset{G_S}{=} q''G_S + Q'$ | | |
| | $U \overset{G_S}{=} xG_S + yQ$ | | |
| | $\xleftarrow{anm,W}$ | | |
| $c \in_{\mathcal{R}} [0, n_S)$ | | | |
| $key = \omega(\pi(\alpha W)\|c)$ | | | |
| $e_1 = \omega(key)$ | $\xrightarrow{c}$ | $\xrightarrow{c,w''}$ | |
| | | $\xrightarrow{q'',U,\rho}$ | $key = \omega(\pi((w' + w'')A)\|c)$ |
| | | | Unless $e_1$ is valid, abort the session. |
| | | | $W \overset{G_S}{=} w''G_S + W'$ |
| | | | $a = \omega(\pi(W) \mid c \mid *)$ |
| | | | $r = a(\mu(k, *) + \rho) + w' + w'' \bmod n_S$ |
| | | | $Q \overset{G_S}{=} q''G_S + Q'$ |
| | | | $e_T = k' \oplus ((\rho + \Delta \bmod n_S) \mid \bar{\#})$ |
| | | | $\Psi \overset{G_T}{=} (\epsilon'_P + \epsilon''_P)\Upsilon$ |
| | | | $e_P = \pi((\mu(k, *) + \rho)Q) \oplus \bar{\#}$ |
| | | | $V \overset{G_S}{=} (\mu(k, *) + \rho)U$ |
| | | | $b = \omega(r \mid e_P \mid e_T \mid (\epsilon'_P + \epsilon''_P)G_T \mid \pi(Q))$ |
| | | $\xleftarrow{r,s,e_P,V}$ | $s = b(\mu(k, *) + \rho) + q' + q'' \bmod n_S$ |
| | | $\xleftarrow{e_T,\Psi}$ | |
| | Unless $W, r, Q, s, e_P, V, e_T, \Delta, \Psi$ are valid, abort the session. | | |
| | $\xleftarrow{r,Q,s,e_P}$ | | |
| | $\xleftarrow{e_T,E_P}$ | | |
| Unless $W, r, Q, s, e_P$ $e_T, E_P$ are valid, abort the session. | | | |

図 7.6: 追跡不能譲渡プロトコル: Unlink-Delegate

#### *UA* による検証式

$$rG_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid *)S\,(S - anm \cdot G_S) + W \tag{7.18}$$

$$sG_S \overset{G_S}{=} \omega(r \mid e_T \mid \Delta \mid E_P \mid \pi(Q))\,(S - anm \cdot G_S) + Q \tag{7.19}$$

$$V \overset{G_S}{=} x(S - anm \cdot G_S) + y \cdot \pi^{-1}(e_P \oplus \bar{\#}) \tag{7.20}$$

$$\Psi \overset{G_T}{=} \xi E_P + \eta \cdot \pi^{-1}(e_T \oplus \rho) \tag{7.21}$$

***SA* による検証式** (7.18) 及び (7.19)

***SP* による $\bar{\#}$ の復号** *SP* は、*SA* から $(c, *, anm, W, r, Q, s, e_P)$ を受け取り、以下の手順で $\bar{\#}$ を復号する。

1. 検証式 (7.18) 及び (7.19) を検査する。
2. 下式により $\rho$ 及び *aid* を計算する。

$$\bar{\#} = \pi((\sigma - aaid)Q) \oplus e_P$$

### 7.4.1.3 Link-Delegate の定義

| *SA* | | *UA* | | *SpA* |
|---|---|---|---|---|
| | $\xrightarrow{E_U,T,*,\bar{*}}$ | Authenticate $T$. | $\xrightarrow{E_U,T}$ | $w', q' \in_{\mathcal{R}} [0, n_S)$, $\epsilon'_P \in [0, n_T)$ |
| | | | | $W' \overset{G_S}{=} w'G_S$ |
| | | | | $Q' \overset{G_S}{=} q'G_S$ |
| | | | $\xleftarrow{W',Q',E'_P}$ | $E'_P \overset{G_T}{=} \epsilon_T G_T$ |
| | | $\epsilon''_P, \xi, \eta \in_{\mathcal{R}} [0, n_T)$ | | |
| | | $E_P \overset{G_T}{=} \epsilon''_P G_T + E'_P$ | | |
| | | $\Upsilon \overset{G_T}{=} \xi G_T + \eta(E_U + \bar{E}_U T)$ | | |
| | | | $\xrightarrow{\epsilon''_P,\Upsilon,*,\bar{*},\bar{\#}}$ | $k' = \pi((\epsilon'_P + \epsilon''_P)(E_U + \bar{E}_U T))$ |
| | | | $\xleftarrow{\Delta}$ | $\Delta = \mu(k, *) - \mu(\mu(k', \bar{\#}), \bar{*}) \bmod n_S$ |
| | | $\rho, w'', q'', x, y \in_{\mathcal{R}} [0, n_S)$ | | |
| | | $anm = aid - \rho \bmod n_S$ | | |
| | | $W \overset{G_S}{=} w''G_S + W'$ | | |
| | | $Q \overset{G_S}{=} q''G_S + Q'$ | | |
| | | $U \overset{G_S}{=} xG_S + yQ$ | | |
| $c \in_{\mathcal{R}} [0, n_S)$ | $\xleftarrow{anm,W}$ | | | |
| $key = \omega(\pi(\alpha W)\|c)$ | | | | |
| $e_1 = \omega(key)$ | $\xrightarrow{c}$ | | $\xrightarrow{c,w''}$ | |
| | | | $\xrightarrow{q'',U,\rho}$ | $key = \omega(\pi((w' + w'')A)\|c)$ |
| | | | | Unless $e_1$ is valid, abort the session. |
| | | | | $W \overset{G_S}{=} w''G_S + W'$ |
| | | | | $a = \omega(\pi(W) \mid c \mid *)$ |
| | | | | $r = a(\mu(k, *) + \rho) + w' + w'' \bmod n_S$ |
| | | | | $Q \overset{G_S}{=} q''G_S + Q'$ |
| | | | | $e_T = k' \oplus ((\rho + \Delta \bmod n_S) \mid \bar{\#})$ |
| | | | | $\Psi \overset{G_T}{=} (\epsilon'_P + \epsilon''_P)\Upsilon$ |
| | | | | $e_P = \pi((\mu(k, *) + \rho)Q) \oplus (\rho \mid \bar{\#})$ |
| | | | | $V \overset{G_S}{=} (\mu(k, *) + \rho)U$ |
| | | | | $b = \omega(r \mid e_P \mid e_T \mid (\epsilon'_P + \epsilon''_P)G_T \mid \pi(Q))$ |
| | | | $\xleftarrow{r,s,e_P,V}$ | $s = b(\mu(k, *) + \rho) + q' + q'' \bmod n_S$ |
| | | | $\xleftarrow{e_T,\Psi}$ | |
| | | Unless $W, r, Q, s, e_P, V, e_T, \Delta, \Psi$ are valid, abort the session. | | |
| | $\xleftarrow{r,Q,s,e_P}$ | | | |
| Unless $W, r, Q, s, e_P$ $e_T, E_P$ are valid, abort the session. | $\xleftarrow{e_T,E_P}$ | | | |

図 7.7: 追跡可能譲渡プロトコル: Link-Delegate

#### *UA* による検証式

$$rG_S \overset{G_S}{=} \omega(\pi(W) \mid c \mid *)\,(S - anm \cdot G_S) + W \tag{7.22}$$

$$sG_S \overset{G_S}{=} \omega(r \mid e_P \mid e_T \mid E_P \mid \pi(Q))\,(S - anm \cdot G_S) + Q \tag{7.23}$$

$$V \overset{G_S}{=} x(S - anm \cdot G_S) + y \cdot \pi^{-1}(e_P \oplus (\rho \mid \bar{\#})) \tag{7.24}$$

$$\Psi \overset{G_T}{=} \xi E_P + \eta \cdot \pi^{-1}(e_T \oplus ((\rho + \Delta \bmod n_S) \mid \bar{\#}) \tag{7.25}$$

***SA* による検証式** (7.22) 及び (7.23)

***SP* による $\bar{\#}$ と *aid* の復号** *SP* は、*SA* から $(c, *, anm, W, r, Q, s, e_P)$ を受け取り、以下の手順で $\bar{\#}$ 及び *aid* を復号する。

1. 検証式 (7.22) 及び (7.23) を検査する。
2. 下式により $\rho$ 及び *aid* を計算する。

$$\rho \mid \bar{\#} = \pi((\sigma - aaid)Q) \oplus e_P$$

$$aid = anm + \rho \bmod n_S$$

### 7.4.2 安全性の評価

Unlink-Delegate 及び Link-Delegate を Link-Verify と比較した際の本質的な違いは、$\rho + \Delta \bmod n_S$ と譲渡先での#識別子 $\bar{\#}$ の暗号化

$$e_T = \pi((\epsilon'_P + \epsilon''_P)(E_U + \bar{\mathcal{E}}_U T)) \oplus ((\rho + \Delta \bmod n_S) \mid \bar{\#})$$

を出力する点である。

この点で、Unlink-Delegate 及び Link-Delegate の健全性は、Link-Verify の健全性に帰着する。

Unlink-Delegate 及び Link-Delegate の完全性は容易に確かめることができるので、追跡不能性と Consensual Disclosure のサポートについて見る。

$e_T$ は、$T$ に対応する個人鍵 $\tau$ を保持し、$E_U$ を構成する $E_P$(Delegate-Receive) を生成した *SpA* のみが復号できる暗号である。一方、*UA* は $\Delta$ を受け取った後に、$\rho$ を一様にランダムに生成するので、$\rho + \Delta \bmod n_S$ も一様に分布する。即ち、$\chi = \rho + \Delta \bmod n_S$ と表す時、Decisional Diffie-Hellman 問題の困難性を仮定すれば、$(anm, W, r, Q, s, e_P, \chi)$ の分布において、この認証イベントに依存する要素は、関係式

$$aid = anm + \chi - \Delta \bmod n_S \tag{7.26}$$

を満たす点だけである。従って、唯一可能性のある攻撃は、譲渡先の *UA* 或いは *SpA* が、譲渡元の *SpA* が計算する $\Delta$ を知っていて、関係式 7.26 から *aid* を計算するもののみである。

従って、以下の定理が成立する。

**定理 19.** $\Delta$ を予め知ることができる譲渡先の *UA* 或いは *SpA* が *aid* を計算する攻撃を除いて、*Unlink-Delegate* は追跡不能性をサポートする。

**定理 20.** $\Delta$ を予め知ることができる譲渡先の *UA* 或いは *SpA* が *aid* を計算する攻撃を除いて、*Link-Delegate* は *Consensual Disclosure* をサポートする。

詳細な証明は、定理 13、及び、定理 14 と同様であるので、ここでは省略する。

### 7.4.3 計算量の評価

Unlink-Delegate 及び Link-Delegate の計算量を表 7.3 に示す。

表 7.3: Delegate-Receive、Unlink-Delegate 及び Link-Delegate の計算量

| | *SA* | *UA* | *SpA* | 合計 |
|---|---|---|---|---|
| Delegate-Receive | — | 1 | 3 | 4 |
| Unlink-Delegate | 5 | 12 | 9 | 26 |
| Link-Delegate | 5 | 17 | 10 | 32 |

(楕円曲線上のスカラー倍演算の実行回数)

# 第8章　相互運用のためのUACML (Ubiquitous Access Control Message Layer) の提案

## 8.1 UACML の基本設計

4.6 で見たように、相互運用性の観点から、Message specification (メッセージの規定) と Lower-layer connectivity (下位通信層との接続性) の 2 項目の要件が存在する。本章で規定する UACML (Ubiquitous Access Control Message Layer) は、上記の 2 項目の要件のサポートを目的とする。

### 8.1.1 メッセージの規定

USCML は、Service Provider、Service Provider Agent、User Agent、及び、Service Appliance の間で交換されるメッセージに関して、構文・意味 (セマンティクス)・符号化方式を厳密に定義することで、ドメインを横断したシームレスなアクセス制御を可能とする。

USCML で規定するメッセージは、5 章から 7 章において定義したプロトコルを実行するためのものである。そのため、メッセージは、*SP/UA* 間、*SA/UA* 間、及び、*UA/SpA* 間のインタフェースに属するものに分類される。

本論文で提案するプロトコルの設計では、計算量の削減にひとつの焦点を当てており、一定の成果を挙げている。しかしながら、プロトコルにおいていくら実行速度に注意を払っても、メッセージ規定、特に、符号化方式が冗長であると大きな効果を期待できない。そこで、本章で述べるメッセージ規定では、通信のオーバーヘッドを小さくすることを目的に、TLV (Type-Length-Value) の独自の改良により、メッセージ長の短縮を図ると共に、メッセージ中のフィールドの処理順序を意識して、可能な限り一方向スキャンによって処理ができるように工夫している。

### 8.1.2 下位通信層との接続性

ユビキタスコンピューティングでは、下位通信層における通信手段はドメインに依存し、無線 LAN、赤外線通信、Bluetooth、非接触 IC カード、NFC 等、複数の選択肢が存在するのみならず、将来に新しい方式が実用化される蓋然性も高い。サービスへのシームレスなアクセスを実現するためには、アクセス制御のプロトコルは、下位通信層の手段に依存しないように設計されるべきである。

また、2.2 では、下位通信層における追跡不能性の実現方式によっては、下位層での通信手段への非依存に加えて、アクセス制御プロトコルが満足しなければならない要件があることを見た。例えば、IP 層においてブロードキャストにより追跡不能性を実現する場合、通信の信頼性を保証する TCP を上位で利用することはできないことを考慮しなければならない。

USCML では、位通信層との接続性の観点から、以下の性質を満足するように設計する。

- 下位通信層に通信の信頼性が期待できないケースであっても、メッセージを検査することにより、パケットの部分的喪失や順序の狂いを検知し、再送を要求できること。
- 異なるユーザに対するメッセージが混在し、通信が輻輳する場合においても、メッセージに記載された情報から自分宛のメッセージを確実に選択できること。

- 更に、複数のセッションが存在する場合においても、メッセージに記載された情報から、セッション毎に仕分けができること。

## 8.2 UAC_Message の概要

UAC_Message は、権利発行と権利認証を目的として交換されるメッセージであり、*SP/UA* 間、*SA/UA* 間、及び、*UA/SP-A* 間の各インタフェースで交換されるメッセージ群を規定する。

| インタフェース | 権利発行 | 権利認証 |
|---|---|---|
| *SP/UA* | UA_Request, SP_Issue | |
| *SA/UA* | | SA_Call, UA_AID, SA_Initiate, UA_Witness, SA_Challenge, UA_Response, SA_Confirm |
| *UA/SP-A* | UA_Setup, SPA_Request, UA_Issue | UA_Initiate, SPA_Witness, UA_Challenge, SPA_Response, UA_Confirm, SPA_Confirm |

UAC_Message は以下の基本的な構造を有する。

```
UAC_Message          ::= Type Message_Body Check_Sum?
Message_Body         ::= UA_Setup|
                         SPA_Request| SPA_Request_Err|
                         UA_Request|
                         SP_Issue| SP_Issue_Err|
                         UA_Issue| UA_Issue_Err|
                         SA_Call|
                         UA_AID|
                         SA_Initiate| SA_Initiate_Err|
                         UA_Initiate|
                         SPA_Witness| SPA_Witness_Err|
                         UA_Witness| UA_Witness_Err|
                         SA_Challenge| SA_Challenge_Err|
                         UA_Challenge| UA_Challenge_Err|
                         SPA_Response| SPA_Response_Err|
                         UA_Response| UA_Response_Err|
                         SA_Confirm| SA_Confirm_Err
                         UA_Confirm| UA_Confirm_Err
                         SPA_Confirm| SPA_Confirm_Err
```

[Type] は 1 バイトのフィールドであり、その上位 2 ビットは下表のようにインタフェースを指示する。

| 上位 2 ビット | 定義 |
|---|---|
| 0 | Reserved. |
| 1 | *SP/UA* 間インタフェース |
| 2 | *SA/UA* 間インタフェース |
| 3 | *UA/SP-A* 間インタフェース |

MSB から 3 ビット目は、Check_Sum の有無を指示する。

| 上位 3 ビット目 | 定義 |
|---|---|
| 0 | Check_Sum を伴わない |
| 1 | Check_Sum は、Message_Body の H バイトのハッシュ値 |

Type の値とメッセージとの対応は下表によって定義される。

| 上位 2 ビット | 下位 5 ビット | 定義 |
|---|---|---|
| 1 | 0 | UA_Request |
| | 1 | — |
| | 2 | SP_Issue |
| | 3 | SP_Issue_Err |
| | 4 – 31 | Reserved. |
| 2 | 0 | SA_Call |
| | 1 | UA_AID |
| | 2 | SA_Initiate |
| | 3 | SA_Initiate_Err |
| | 4 | UA_Witness |
| | 5 | UA_Witness_Err |
| | 6 | SA_Challenge |
| | 7 | SA_Challenge_Err |
| | 8 | UA_Response |
| | 9 | UA_Response_Err |
| | 10 | SA_Confirm |
| | 11 | SA_Confirm_Err |
| | 12 – 31 | Reserved. |
| 3 | 0 | UA_Setup |
| | 1 | SPA_Request |
| | 2 | UA_Issue |
| | 3 | UA_Initiate |
| | 4 | SPA_Witness |
| | 5 | SPA_Witness_Err |
| | 6 | UA_Challenge |
| | 7 | UA_Challenge_Err |
| | 8 | SPA_Response |
| | 9 | SPA_Response_Err |
| | 10 | UA_Confirm |
| | 11 | UA_Confirm_Err |
| | 12 | SPA_Confirm |
| | 13 | SPA_Confirm_Err |
| | 14 – 31 | Reserved. |

## 8.3 メッセージの文法と符号化規則

この節では、UAC_Message を具体的に規定するための構文文法と符号化規則について述べる。

*SP-A* は、IC カードなど通信能力及び演算能力において制限のある実装をとる可能性が高い。そのため、*SP-A* と *SA* の通信効率のボトルネックは、*SP-A* 内の通信バッファの容量制限と *SP-A* による通信メッセージの解釈能力の限界に起因する。

UAC_Message の文法と符号化規則は、メッセージサイズと解釈時の効率を最適化し、*SP-A* における通信・計算の負荷を軽減するように設計されている。

UAC_Message は基本的にはフィールド (Field) と呼ばれるバイト列の並びであるが、より正確には、次の BNF 記法によって構造が定義される。

```
UAC_Message            ::= EMPTY | Field+
Field                  ::= Fixed_Length_Field|Variable_Length_Field
                           |Repeat_Type_Field|Composite_Field
Fixed_Length_Field     ::= VALUE
Variable_Length_Field  ::= PREFIX VALUE
Repeat_Type_Field      ::= PREFIX (EMPTY | Fixed_Length_Field+ |
                           Variable_Length_Field+ | Composite_Field+)
Composite_Field        ::= EMPTY | Field+
PREFIX                 ::= 1-byte or 3-byte long string
VALUE                  ::= An arbitrary byte-string
EMPTY                  ::= The empty byte-string
```

### 8.3.1 フィールド (Field)

フィールドには、**固定長フィールド**、**可変長フィールド**、**複合フィールド**、及び、**繰り返し型フィールド**の 4 種類が存在し、それぞれ構造が異なる。

#### 8.3.1.1 固定長フィールド (Fixed-Length Field)

固定長フィールドは、UAC_Message の定義中で固定の長さを定められたフィールドであり、値となるバイト列 (VALUE) そのものである。

UAC_Message の定義中では、固定長フィールドは、$[\textit{Field\_Name}]^n$ と表記される。*Field_Name* は、UAC_Message 定義中で与えられるフィールド名であり、$n$ はこのフィールドのバイト長を表す。

**例** 西暦の年を、2 バイトの固定長フィールド $[\texttt{Year}]^2$ に指定する。

[Year] に指定される西暦年が 1960 年であれば、フィールドは 2 バイトのバイト列 0x07 0xA8 ($7 \times 256 + 10 \times 16 + 8 = 1960$) となる。

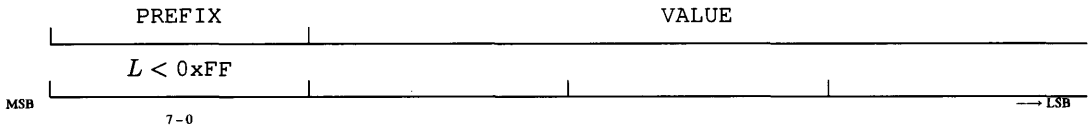
#### 8.3.1.2 可変長フィールド (Variable-Length Field)

可変長フィールドは、UAC_Message の定義において可変の長さを与えられたフィールドで、二つのバイト列 PREFIX と VALUE で構成される。

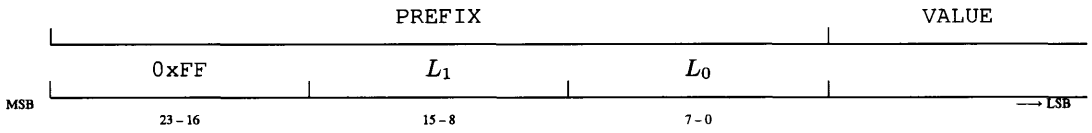
PREFIX は VALUE の前に位置し、VALUE のバイト長を指定する 1 バイトまたは 3 バイトのバイト列である。

PREFIX のが 1 バイトであるか 3 バイトであるかは、VALUE のバイト長によって異なる。

**VALUE のバイト長 $L$ が 255 未満の時:** PREFIX は $L$ を値として持つ 1 バイトデータである。




**VALUE のバイト長が 255 以上 65,535 以下の時:** PREFIX は 3 バイトのデータであり、上位 8 ビットは 1 に設定され (最上位バイトの値は 0xFF)、下位 16 ビット (下位 2 バイト) で $L$ を指定する。




$$L = 2^8 L_1 + L_0$$

VALUE の長さが 65,535 バイトを超えることはない。

可変長フィールドの値は、VALUE に指定される。

UAC_Message の定義中では、可変長フィールドは、(*Field_Name*) と表記される。*Field_Name* は、UAC_Message 定義中で与えられるフィールド名である。

**例** 人物の名前をアスキー文字に符号化して、可変長のフィールド (Family_Name) に指定する。

(Family_Name) に登録する名前が SHIN であれば、フィールド全体は 5 バイトのバイト列 0x04 'S' 'H' 'I' 'N' となる。

#### 8.3.1.3 複合フィールド (Composite Field)

複合フィールドは、空 (EMPTY)、或いは、一つ以上のフィールドから構成されるフィールドである。複合フィールドが (子) フィールドを構成要素として持つ場合、子フィールドには固定長フィールド、可変長フィールド、複合フィールド、繰り返し型フィールドのいずれを指定することも可能である。

UAC_Message の定義中では、複合フィールドは<*Field_Name*>と表記される。ただし、*Field_Name* は UAC_Message 定義中で与えられるフィールド名であるとする。

**例** <Personal_Data> ::= <Birthday>(Family_Name)(Given_Name)
<Birthday> ::= [Year]$^{2}$[Month]$^{1}$[Day]$^{1}$

Personal_Data をフィールド名にもつ複合フィールドは、Birthday、Family_Name、Given_Name の三つのフィールドから構成される。

このうち、Family_Name と Given_Name は、名前を与える可変長の文字列データであり、Birthday は生年月日を与える三つの固定長フィールドから構成される複合フィールドである。

**例** <Extended_Personal_Data> ::= <Personal_Data><Address>
<Address> ::= EMPTY | ((Country)(City)(Street))

Extended_Personal_Data は、Personal_Data に、Address フィールドを付け加えて拡張した複合フィールドである。

必ずしも全ての人物に対して住所が登録されるわけではないので、複合フィールド Address は空のデータ (EMPTY) が取れるように定義される。

Extended_Personal_Data は、以下のように定義しても同じである。

<Extended_Personal_Data> ::= <Personal_Data><Address>?
<Address> ::= (Country)(City)(Street)

#### 8.3.1.4 繰り返し型フィールド (Repeat-Type Field)

UAC_Message の定義中で名前が与えられた一つの固定長フィールド、可変長フィールド、或いは、複合フィールドの繰り返し構造を定義するフィールドである。

繰り返し型フィールドは、+を、子フィールドの後ろに記述することで、表記される。

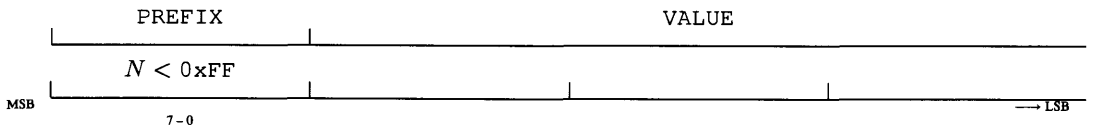
**例** <Name_List> ::= <Personal_Data>+

は、一つ以上の Personal_Data フィールドを含む名簿のデータ構造を定義する。

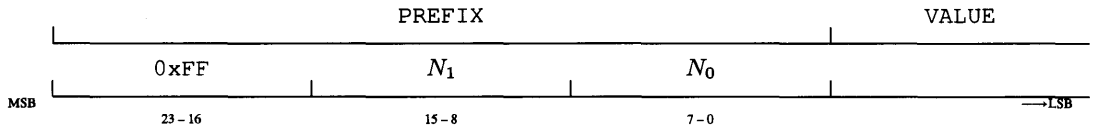
子フィールドの繰り返しの回数は、繰り返し型フィールドの先頭に指定される 1 バイト或いは 3 バイトの PREFIX で与えられる。

PREFIX が 1 バイトであるか 3 バイトであるかは、繰り返しの回数によって異なる。

**子フィールドの数 $N$ が 255 未満の時:** PREFIX は $N$ を値として持つ 1 バイトデータである。



**子フィールドの数が 255 以上 65,535 以下の時:** PREFIX は 3 バイトであり、上位 8 ビットは 1 に設定され (最上位 1 バイトの値は 0xFF)、下位 16 ビット (下位 2 バイト) が $N$ を与える。




$$N = 2^8 N_1 + N_0$$

繰り返し型フィールドは、65,535 個を超える子フィールドを含むことは出来ない。

### 8.3.2 フィンガープリント

フィールドを区別するためのタグやタイプがメッセージ中には指定されない点で、UAC_Message の定義は通常の TLV(Type-Length-Value) 記法と異なる。つまり、タグやタイプによってフィールドを識別する手段が提供されないので、UAC_Message の定義の中で、曖昧さを排除する指定がなされなければならない。

特に、UAC_Message 中においてフィールドを省略可能とすることができるが、タグやタイプの指定が存在しないため、メッセージ中でどのフィールドが存在し、どのフィールドが省略されているかを指定するための手段が必要となる。メッセージ長を増大させずに、この手段を提供するために用いられる手段がフィンガープリントである。

メッセージ中の任意の固定長フィールドをフィンガープリントに指定することが可能であり、その場合、フィンガープリントに指定された固定長フィールドの各ビットは、UAC_Message 中省略可能なフィールドに対応付けられる。すなわち、対応するフィールドがメッセージに指定されている時には 1 が、省略されているときには 0 がセットされる。

**例** 以下の定義は、<Extended_Personal_Data>+の解釈に曖昧さを残すため、正しい定義ではない。

```
<Extended_Name_List>      ::= <Extended_Personal_Data>+
<Extended_Personal_Data>  ::= <Personal_Data><Address>?
<Address>                 ::= (Country)(City)(Street)
```

すなわち、<Extended_Personal_Data>+の定義に従って符号化することは勿論可能であるが、各々の<Extended_Personal_Data>フィールドに<Address>フィールドが含まれないという情報が残らないので、解釈は一意に定まらない。

この問題を解決するために、Extended_Personal_Data の定義を、

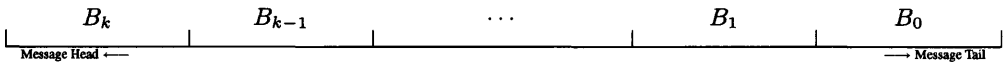
```
<Extended_Personal_Data> ::= [Finger_Print][1]<Personal_Data><Address>?
```

とし、Finger_Print の最上位ビット (MSB) を省略可能なフィールドである<Address>に対応付ける。このようにすれば、[Finger_Print] の最上位ビットを確かめることで、<Address>が指定されているか否かを確認できるため、先に述べたような曖昧さの問題はなくなる。

### 8.3.3 数の符号化

数をバイナリデータとして UAC_Message 等に指定する時には、上位バイトから順に値を指定していく、いわゆる、ビッグエンディアン (Big Endian) を採る。

例えば、$n = 2^{8k}B_k + 2^{8(k-1)}B_{k-1} + \cdots + 2^8 B_1 + B_0, \quad B_i \in [0\text{x}00, 0\text{xFF}]$ を符号化した結果は、下図のようになる。




また、可変長フィールドに数を指定する時には、メッセージのサイズを小さくするために、$n \neq 0$ である限り、$B_k \neq 0$ となるように上位バイトを切り詰める。但し、0 は 1 バイトの 0x00 に符号化する。

### 8.3.4 符号化規則における記法

| 記号 | 定義 |
|---|---|
| a – z, A – Z, 0 – 9, <, >, / | 7 ビットで符号化された ASCII 文字。 |
| *Base64* | Base64 符号化方式に従う ASCII 文字列。 |
| $Base64^k$ | Base64 符号化方式に従う $k$ バイトの ASCII 文字列。 |
| *Base64*∗ | Base64 符号化方式に従う任意長の ASCII 文字列。 |
| $o$ | 任意のバイト。 |
| $o^k$ | $k$ バイトのバイト列。 |
| $o^*$ | 任意長のバイト列。 |
| $c$ | 表示可能文字 (ASCII コードと ISO 646 IRV コードの共通部分から、制御文字をのぞいた文字)。 |
| $c^k$ | $k$ 文字からなる表示可能文字列。 |
| $c^*$ | 任意長の表示可能文字列。 |
| $C$ | UTF-8 コード文字。 |
| $C^k$ | $k$ 文字からなる UTF-8 コード文字列。 |
| $C^*$ | 任意長の UTF-8 コード文字列。 |
| $d$ | 0 – 9 のいずれかに対応する ASCII 文字。 |
| $d^k$ | $k$ 桁の 10 進数を表現する 0 – 9 からなる ASCII 文字列。 |
| $d*$ | 0 – 9 からなる任意長の ASCII 文字列。 |
| $\emptyset$ | 空の文字列。 |

## 8.4 Service Provider/User Agent インタフェース

### 8.4.1 概要

*SP・UA* 間の権利発行は、以下のメッセージを以下の順序で交換することで実行される。SP_で始まるメッセージは *SP* が送信するメッセージ、UA_で始まるメッセージは *UA* が送信するメッセージを表す。

| メッセージ | 定義 |
|---|---|
| UA_Request | *UA* が *aid* の発行を要求する際に *SA* に送信する。UA_Request には、サービス公開鍵と *SP-A* の公開鍵証明書に加えて、*SP* と *SP-A* との間で鍵 $k$ を交換するための暗号データを記載する。 |
| SP_Issue | *SP* が生成した *aid*、*認証子、#識別子、及び、鍵 $k$ を交換するための暗号データを記載する。 |

## 8.4.2 UA_Request

### 8.4.2.1 規定

```
<UA_Request>          ::= [ServiceID]^H (Cert) (EUx)[EUy]^1
```

UA_Request の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [ServiceID] | 固定長 | $o^H$ | サービス公開鍵 $(\mathcal{G}_S, G_S, S)$ の X.509 証明書の ASN.1 符号化に対するハッシュ値 |
| (Cert) | 可変長 | $o^*$ | *SP-A* の公開鍵 $(\mathcal{G}_T, G_T, T)$ に対する X.509 証明書の ASN.1 符号化 |
| (EUx) | 可変長 | $o^*$ | $k$ を交換するために、*UA* と *SP-A* とが協調して生成した暗号化データ $E_U \overset{\mathcal{G}_T}{=} \epsilon_U G_T + E_T$ の $x$ 座標 |
| [EUy] | 固定長 | $o$ | $E_U \overset{\mathcal{G}_T}{=} \epsilon_U G_T + E_T$ の $y$ 座標の最下位ビット |

### 8.4.2.2 *UA* によるメッセージの生成手順

1. *UA* は UA_Setup を *SP-A* に送付し (8.6.2)、返信として SPA_Request を受け取る。
2. 乱数 $\epsilon_U \in [0, n_S)$ を生成し、

   $$E_U \overset{\mathcal{G}_T}{=} \epsilon_U G_T + E_T$$

   を計算する。

   $E_T$ は SPA_Request に指定される (ETx)[ETy] の $\mathcal{G}_T$ への逆射影である。

### 8.4.2.3 *SP* によるメッセージの処理手順

1. [ServiceID] が、取り扱うサービスの X.509 公開鍵証明書の ASN.1 符号化データのハッシュ値と一致することを検査する。
2. SP_Issue を *UA* に送付する (8.4.3)。

### 8.4.3 SP_Issue

#### 8.4.3.1 規定

```
<SP_Issue>              ::= [MessageID]^H (aid) (Rules) [HashedApplianceInfo]^H+ (Sx) [Sy]^1
                            [ID]^8 (EPx) [EPy]^1 <GroupDef>
<GroupDef>              ::= [Binary]^1(Field)(a)(b)(Gx)[Gy]^1(n)
[HashedApplianceInfo]::= Hash of <ApplianceInfo>
<ApplianceInfo>         ::= [Privileges]^1 (Ax)[Ay]^1
```

SP_Issue の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [MessageID] | 固定長 | $o^H$ | 受理した UA_Request のハッシュ値 |
| (aid) | 可変長 | $o^*$ | *SP* が生成した *aid* を指定 |
| (Rules) | 可変長 | $o^*$ | *aid* に付随するアクセスルール記述 |
| [HashedApplianceInfo] | 固定長 | $o^H$ | 以下に定義する<ApplianceInfo>のハッシュ値 |
| <ApplianceInfo> | 複合 | — | *aid* でアクセスできる *SA* に関する情報、*SP* に許されている認証モードと公開鍵を指定 |
| [Privileges] | 固定長 | o | 認証モードの実行権限の有無 |
| (Ax) | 可変長 | $o^*$ | *SA* の公開鍵 $A$ の $x$ 座標 |
| [Ay] | 固定長 | $o$ | *SA* の公開鍵 $A$ の $y$ 座標の最下位ビット |
| [ID] | 固定長 | $o^8$ | *SP* が *aid* の#識別子、サービス公開鍵毎に一意 |
| (EPx) | 可変長 | $o^*$ | $k$ を交換するために、*SP* が生成した暗号化データ $E_P \stackrel{G_T}{=} \epsilon_P G_T$ の $x$ 座標 |
| [EPy] | 固定長 | $o$ | $E_P \stackrel{G_T}{=} \epsilon_P G_T$ の $y$ 座標の最下位ビット |
| <GroupDef> | 複合 | – | サービス公開鍵の基礎となる群の定義 |
| [Binary] | 固定長 | $o$ | 基礎群が、標数 2 の楕円曲線の部分群である時に 1 を指定、奇標数の素体上で定義される時に 0 を指定 |
| (Field) | 可変長 | $o^*$ | 楕円曲線が標数 2 の素体の拡大体上で定義されている時には拡大次数を指定、標数が奇数である時には標数を指定 |
| (a) | 可変長 | $o^*$ | 楕円曲線の定義 $y^2 = x^3 + ax + b$(奇標数の場合)、或いは、$y^2 + xy = x^3 + ax^2 + b$ の係数 $a$(標数 2 の場合) |
| (b) | 可変長 | $o^*$ | 楕円曲線の定義 $y^2 = x^3 + ax + b$(奇標数の場合)、或いは、$y^2 + xy = x^3 + ax^2 + b$(標数 2 の場合) の係数 $b$ |
| (Gx) | 可変長 | $o^*$ | 楕円曲線上の基点の $x$ 座標 |
| [Gy] | 固定長 | $o$ | 楕円曲線上の基点の $y$ 座標の最下位ビット |
| (n) | 可変長 | $o^*$ | 基礎群中での基点の位数 |
| (Sx) | 可変長 | $o^*$ | サービスの公開鍵 $S$ の $x$ 座標 |
| [Sy] | 固定長 | $o$ | サービスの公開鍵 $S$ の $y$ 座標の最下位ビット |

Privileges の符号化規則

| ビット | 定義 |
|---|---|
| MSB | 0: 顕名認証の実行権限無　1: 顕名認証の実行権限有 |
| −1 | 0: 権利消費の実行権限無　1: 権利消費の実行権限有 |
| −2 | 0: 権利更新の実行権限無　1: 権利更新の実行権限有 |
| −3 | 0: 鍵転送の実行権限無　1: 鍵転送の実行権限有 |
| −4 | 0: 権利無効化の実行権限無　1: 権利無効化の実行権限有 |
| −5 – LSB | Reseved. 全てのビットがセットされる |

#### 8.4.3.2 *SP* によるメッセージの生成手順

1. アクセスルールの表現である (Rules) を特定する。
2. # 識別子 [ID](#) を特定する。
3. 特権を与える *SA* を表現するリスト [HashedAplliance Info]+を特定する。
4. サービスの公開鍵 $S$ とその基礎となる群 $(\mathcal{G}_S, G_S, n_S)$ を (Sx) [Sy] 及び<GroupDef>に符号化する。
5. 乱数 $\epsilon_P \in_R [0, n_T)$ を生成し、下式により $E_P$ 及び *aid* を計算し、(EPx) [EPy] 及び (aid) に符号化する。

$$
\begin{aligned}
E_P &\stackrel{\mathcal{G}_T}{=} \epsilon_P G_T \\
k' &= \pi(\epsilon_P(E_U + \bar{E}_U T)) \\
k &= \mu(k', \#) \\
aid &= \sigma - \mu(k, *) \bmod n_S
\end{aligned}
$$

- (Rules) のハッシュ値を [HashedRules] とあらわすとき、$*$ は以下の符号である。
  [HashedRules] [HashedApplianceInfo]+
- $E_U$ は UA_Request に指定される (EUx) [EUy] の $\mathcal{G}_T$ への逆射影である。

#### 8.4.3.3 *UA* によるメッセージの処理手順

1. [MessageID] が先に送信した UA_Request のハッシュ値と一致することを検証する。
2. 内部データベースに権限レコードを生成し、(aid)、(Rules)、[HashedApplianceInfo]+、$(\mathcal{G}_S, G_S, n_S)$ (<GroupDef>)、$S$ ((Sx) [Sy])、及び、SPA_Request に指定される [KeyID] とを記録する。

   この権限レコードは、SA_Call 受信時に、<GroupDef> (Sx) [Sy] のハッシュ値を鍵として検索される。
3. [ID] (EPx) [EPy] <GroupDef>を UA_Issue に指定して *SP-A* に送付する (8.6.4)。

## 8.5 Service Appliance/User Agent インタフェース

### 8.5.1 概要

*SA*・*UA* 間の権利認証は、以下のメッセージを以下の順序で交換することで実行される。SA_で始まるメッセージは *SA* が送信するメッセージであり、UA_で始まるメッセージは *UA* が送信するメッセージである。

| メッセージ | 定義 |
|---|---|
| SA_Call | *SA* がサポートするサービス公開鍵のリストを認証モードと共に送信する。*UA* は受信したリストからサービス公開鍵を選択する。 |
| UA_AID | *UA* は選択したサービス公開鍵をと共に、匿名の場合は *anm*、顕名の場合は *aid* を *SA* に送信する。 |
| SA_Initiate | *SA* は、チャレンジ $c \in [0, n_S)$ をランダムに生成し、そのハッシュ値をウィットネス $e_0$ として *UA* に送信する。*SA*・*UA* 間のセッションは SA_Initiate の送信 (受信) と共に開始され、SA_Confirm の送信 (受信) をもって終了する。*SA* は、このセッションで交換するメッセージの識別番号 MessageID の開始番号を生成し、SA_Initiate に付して *UA* に送信する。 |
| UA_Witness | *UA* と *SP-A* とが協調して生成するウィットネス $W$ を *SA* に送信する。 |
| SA_Challenge | *SA* が生成したチャレンジ $c$ を *UA* に送信する。権利更新、鍵転送、権利無効化において必要となる情報も、このメッセージに記載して送信する。 |
| UA_Response | *UA* と *SP-A* とが協調して生成するレスポンス $r \in [0, n_S)$ を *SA* に送信する。鍵転送のために *UA* が生成するデータ $R \in \mathcal{G}_S$ も、このメッセージに記載して送信する。 |
| SA_Confirm | *SA* が認証の結果を *UA* に通知する。権利消費、権利更新、権利無効化等による永続データへの変更は、*SP-A* がこのメッセージを確認してから確定する。 |

## 8.5.2 SA_Call

### 8.5.2.1 規定

```
<SA_Call>          ::= <Indicator>+
<Indicator>        ::= [Thumbnail]^H [AuthMode]^1 [HashedApplianceInfo]^H
```

SA_Call の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| <Indicator> | 複合 | — | *SA* が提供しえるサービスのインジケータ |
| [Thumbnail] | 固定長 | $o^H$ | 提供するサービスの公開鍵の<GroupDef>(Sx)[Sy] のハッシュ値 |
| [AuthMode] | 固定長 | o | 要求する認証モード |
| [HashedApplianceInfo] | 固定長 | $o^H$ | *SA* がこのサービスに関して有する認証モード特権 [Privilege] と公開鍵 (Ax)[Ay] に関して、[Privilege](Ax)[Ay] のハッシュ値 |

AuthMode の符号化規則

| ビット | 定義 |
|---|---|
| MSB | 0: 匿名認証 1: 顕名認証 |
| −1 | 0: 権利消費無 1: 権利消費有 |
| −2 | 0: 権利更新無 1: 権利更新有 |
| −3 | 0: 鍵転送無 1: 鍵転送有 |
| −4 | 0: 権利無効化無 1: 権利無効化有 |
| −5 - LSB | Reseved. 全てのビットがクリアされる |

### 8.5.2.2 *SA* によるメッセージの生成手順

<Indicator>毎に以下の手順を繰り返す。

1. サービス公開鍵 $(\mathcal{G}_S, G_S, n_S, S)$ に対して、<GroupDef>(Sx)[Sy] のハッシュ値を計算し、[Thumbnail] に符号化する。
2. サービス提供時の認証モードを決定し、[AuthMode] に符号化する。
3. *SA* の特権モード及び公開鍵 $A$ を [Privilege](Ax)[Ay] に符号化し、そのハッシュ値を [HassedApplianceInfo に符号化する。

### 8.5.2.3 *UA* によるメッセージの処理手順

1. SA_Call から<Indicator>を選択することで、アクセスするサービスを選択する。選択しない場合は、このメッセージを無視すればよいが、選択する場合には以下の条件を満足する権利レコードをデータベース中で特定する。

- 権利レコードに指定されるサービス公開鍵 $(\mathcal{G}_S, G_S, n_S, S)$ のハッシュ値が<Indicator>に一致する。
- 権利レコードに指定される [HashedApplianceInfo]+が、<Indicator>中の [HashedApplianceInfo と一致するエントリーを含む。
- [AuthMode] の内容が適切である。特に、「顕名認証」が要求されている際には、ユーザの同意を確認することが必要である。

2. UA_AID を *SP* に送付する (8.5.3)。

### 8.5.3 UA_AID

#### 8.5.3.1 規定

```
<UA_AID>              ::= [TempSessionID]^H <Indicator> (aid) (Rules)
```

UA_AID の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [TempSessionID] | 固定長 | o^H | SA_Call のハッシュ値、ブロードキャストによりメッセージが配信される場合においても *SA* が判別が可能 |
| <Indicator> | 複合 | — | 選択したサービスのインジケータ |
| [Thumbnail] | 固定長 | o^H | 提供するサービスの公開鍵の<GroupDef>(Sx)[Sy] のハッシュ値 |
| [AuthMode] | 固定長 | o | 要求する認証モード |
| [HashedApplianceInfo] | 固定長 | o^H | *SA* がこのサービスに関して有する認証モード特権 [Privilege] と公開鍵 (Ax)[Ay] に関して、[Privilege](Ax)[Ay] のハッシュ値 |
| (aid) | 可変長 | o* | *aid* を乱数化して得られる *anm* を指定 |
| (Rules) | 可変長 | o* | *aid* に付随して *UA* が保有するアクセスルール記述 |

#### 8.5.3.2 *UA* によるメッセージの生成手順

1. SA_Call の処理において、<Indicator>の選択に伴って特定される権利レコードから、基礎群 $((\mathcal{G}_S, G_S, n_S))$、アクセス ID (*aid*)、及び、アクセスルール記述 ((Rules)) を取得する。
2. $\rho \in_R [0, n_S)$ をランダムに選び、当該権利レコード中の *aid* から *anm* を下式により計算し、(aid) に符号化する。

$$anm = aid - \rho \bmod n_S$$

3. セッションの仮の識別番号 [TempSessionID] として、SA_Call のハッシュ値を計算する。

   権利認証においては、SA_Call と UA_AID を含めたメッセージがブロードキャストで配達されるケースも想定して、*SA* 同士が互いに調整することなくし、正しい当事者間での通信を保証するために、ハッシュ関数の非衝突性を利用してセッション識別番号を定める。

#### 8.5.3.3 *SA* によるメッセージの処理手順

1. [TempSessionID] が受信した SA_Call のハッシュ値と一致するかを検査する。
2. メッセージ中の<Indicator>が SA_Call で送付した<Indicator>+のエントリーに含まれることを検査する。
3. SA_Initiate を *UA* に送付する (8.5.4)。

### 8.5.4 SA_Initiate

#### 8.5.4.1 規定

```
<SA_Initiate>          ::= [SessionID]^H <ApplianceInfo>
<ApplianceInfo>        ::= [Privileges]^1 (Ax)[Ay]^1
```

SA_Initiate の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [SessionID] | 固定長 | $o^H$ | UA_AID のハッシュ値、以降このセッションで交換されるメッセージには全て同じ値を指定 |
| <ApplianceInfo> | 複合 | — | *SA* の認証モード特権と公開鍵を指定 |
| [Privileges] | 固定長 | o | 認証モードの実行権限の有無 |
| (Ax) | 可変長 | $o^*$ | *SA* の公開鍵 $S$ の $x$ 座標 |
| [Ay] | 固定長 | $o$ | *SA* の公開鍵 $S$ の $y$ 座標 |

Privileges の符号化規則

| ビット | 定義 |
|---|---|
| MSB | 0: 顕名認証の実行権限無　1: 顕名認証の実行権限有 |
| −1 | 0: 権利消費の実行権限無　1: 権利消費の実行権限有 |
| −2 | 0: 権利更新の実行権限無　1: 権利更新の実行権限有 |
| −3 | 0: 鍵転送の実行権限無　1: 鍵転送の実行権限有 |
| −4 | 0: 権利無効化の実行権限無　1: 権利無効化の実行権限有 |
| −5 − LSB | Reseved. 全てのビットがセットされる |

#### 8.5.4.2 *SA* によるメッセージの生成手順

1. セッション識別番号として、先に受信した UA_AID のハッシュ値を計算する。

   以降、本セッションの識別番号として一貫してこの値を利用する。ブロードキャストによるメッセージ配達において、かつ、*SA* 間の調整を仮定することなく、衝突を回避することを目的とする。

   特に、同一の *SA* に対して複数の *UA* がアクセスするケースを想定している。

2. *SA* は、UA_AID で要求されたサービスの基礎群上で定義される自身の公開鍵 $A$ と、*SP* から許可されている認証モードを [Privileges] に符号化する。

#### 8.5.4.3 *UA* によるメッセージの処理手順

1. メッセージ中の<ApplianceInfo>のハッシュ値と、SA_Call から選択した [HashedApplianceInfo] の一致を検査する。

2. 実行しようとしている認証モードが、*SA* の認証モード特権によってカバーされていることを確認する。具体的には、選択した<Indicator> 中の [AuthMode] のビット毎否定 (bit-wise negation) と、<ApplianceInfo>中の<Privilege>とのビット毎論理和 (bit-wise OR) が 255 になることを確認する。

3. `UA_Initiate` を *SP-A* に送付する (8.6.5)。

### 8.5.5 UA_Witness

#### 8.5.5.1 規定

```
<UA_Witness>          ::= [SessionID]^H <Witness>
<Witness>             ::= (Wx)[Wy]^1
```

UA_Witness の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [SessionID] | 固定長 | $o^H$ | セッションの識別番号、UA_AID のハッシュ値 |
| <Witness> | 複合 | — | *UA* と *SP-A* が協調して生成するウィットネス $W \stackrel{\mathcal{G}_S}{=} (w' + w'')G_S$ |
| (Wx) | 可変長 | $o^*$ | $W$ の $x$ 座標 |
| [Wy] | 固定長 | $o$ | $W$ の $y$ 座標の最下位ビット |

#### 8.5.5.2 *UA* によるメッセージの生成手順

1. UA_Initiate を *SP-A* に送付し、返信として SPA_Witness を受信する。
2. 乱数 $w'' \in_R [0, n_S)$ を生成し、下式によりウィットネス $W$ を計算し、<Witness>に符号化する。

$$W \stackrel{\mathcal{G}_S}{=} w''G_S + W'$$

$W'$ は SPA_Witness に指定される (W'x)[Wy] の $\mathcal{G}_S$ への逆射影である。

#### 8.5.5.3 *SA* によるメッセージの生成手順

1. SA_Challenge を *UA* に送付する (8.5.6)。

### 8.5.6 SA_Challenge

#### 8.5.6.1 規定

```
<SA_Challenge> ::= [SessionID]^H [AuthMode]^1(Challenge) [HashedRules]^H
                   <Seed>?  (RRL)? (NewRules)?  [KeyCert]^H
<Seed>         ::= (Cx)[Cy]^1
```

**[AuthMode] のビットと省略可能フィールドとの対応**

(MSB) (-1) (-2) (NewRules) (-3) <Seed>
(-4) (RRL)

getResponseInput の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [SessionID] | 固定長 | $o^H$ | セッションの識別番号、UA_AID のハッシュ値 |
| [AuthMode] | 固定長 | o | 認証モード |
| (Challenge) | 可変長 | $o^*$ | *SA* が *UA* に送るチャレンジ $c \in [0, n_S)$ |
| [HashedRules] | 固定長 | $o^H$ | この認証イベントで有効な*認証子 |
| <Seed> | — | — | 鍵転送における転送鍵計算のためのシード $C \in \mathcal{G}_S$ |
| (Cx) | 可変長 | $o^*$ | $C$ の $x$ 座標 |
| [Cy] | 固定長 | $o$ | $C$ の $y$ 座標の最下位ビット |
| (RRL) | 可変長 | $o^*$ | 権利無効化における無効化対象となる権利の#識別子のリスト (Rights Revocation List) |
| (NewRules) | 可変長 | $o^*$ | 権利更新における新しいアクセスルール記述 |
| [KeyCert] | 固定長 | $o^{\lvert h \rvert}$ | *SA* と *SP-A* で交換する *key* のハッシュ値 |

#### 8.5.6.2 *SA* によるメッセージの生成手順

1. 乱数 $c \in_R [0, n_S)$ を生成し、(Challenge) に符号化する。
2. (Rules) のハッシュ値を [HashedRules] に符号化する。
3. 認証モードに応じて、<Seed>、(RRL)、及び、(NewRules) を指定する。
4. 鍵転送を行うときには、オリジナルのシード $L$ に対して、乱数 $\lambda \in_R [0, n_S)$ を生成して、下式で計算する $C \in \mathcal{G}_S$ をシードとして<Seed>に符号化する。

   $$C \overset{\mathcal{G}_S}{=} \lambda L$$

5. 下式により、*SP-A* と共有する鍵 *key* を計算し、更に、そのハッシュ値を [KeyCert] に符号化する。

   $$key = \omega(\pi(\alpha W)|c)$$

   $$e_1 = \omega(key)$$

#### 8.5.6.3 *UA* によるメッセージの処理手順

1. UA_Challenge を *SP-A* に送付する (8.6.7)。

### 8.5.7 UA_Response

#### 8.5.7.1 規定

```
<UA_Response>          ::= [SessionID]^H [AuthMode]^1
                           (Signature)?<EncryptedSeed>?(EncryptedMask)?(Response)<QWitness>?
<EncryptedSeed>        ::= (Rx)[Ry]^1
<QWitness>             ::= (Qx)[Qy]^1
```

**[AuthMode] のビットと省略可能フィールドとの対応**

```
(MSB)(Signature), (EncryptedMask), <QWitness>          (-1)                    (-2)
(-3)<EncryptedSeed>
```

getResponseInput の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [SessionID] | 固定長 | $o^H$ | セッションの識別番号であり、UA_AID のハッシュ値 |
| [AuthMode] | 固定長 | o | 認証モード |
| (Signature) | 可変長 | $o^*$ | SPA_Response 中の (Signature) |
| <EncryptedSeed> | — | — | 転送鍵計算のためのシード $S$ の暗号化 $R \in \mathcal{G}_S$ |
| (Rx) | 可変長 | $o^*$ | <EncryptedSeed>の $x$ 座標 |
| [Ry] | 固定長 | $o$ | <EncryptedSeed>の $y$ 座標の最下位ビット |
| (EncryptedMask) | 可変長 | $o^*$ | SPA_Response 中の (EncryptedMask) |
| (Response) | 可変長 | $o^*$ | *UA* が *SA* に送るレスポンス $r \in [0, n_S)$、UA_Witness で送付する<Witness>と併せて、(Challenge)[HashedRules][Finger_Print] に対する Schnorr 署名を構成する |
| <QWitness> | — | — | 顕名認証において $s$ 及び $e_P$ とともに署名と暗号化を構成するデータ $Q \in \mathcal{G}_S$ |
| (Qx) | 可変長 | $o^*$ | $Q$ の $x$ 座標 |
| [Qy] | 固定長 | $o$ | $Q$ の $y$ 座標の最下位ビット |

#### 8.5.7.2 *UA* によるメッセージの生成処理

1. SPA_Response から以下のように生成する。
   - SPA_Response の2バイトの [SessionID] を、*SA/UA* インタフェースの **H** バイトの [SessionID] に置き換える。
   - SPA_Response の [EncryptedTest] を削除する。
   - 匿名認証の場合、SPA_Response の (Response) を置き換える。
   - 顕名認証の場合、<QWitness>を末尾に追加する。
2. 匿名認証の場合、(Response) は、SPA_Response 中の (Response) の数値化である $r'$ に対して、下式で定義される $r$ の符号化である。

$$r = r' + a\rho \bmod n_S$$

但し、$a$ は、$W$ の符号化 (Wx) [Wy]、SA_Challenge の内容、及び、(NewRules) のハッシュ値 [HashedNewRules] をこの順に連接して得られる下記の符号のハッシュ値とする。

(Wx) [Wy] [AuthMode] (Challenge) [HashedRules] <Seed> ? (RRL) ? [HashedNewRules]

3. 顕名認証の場合、下式により $Q$ を計算し、(Qx) [Qy] に符号化する。

$$Q \overset{\mathcal{G}_S}{=} Q' + q''G_S$$

#### 8.5.7.3 *SA* によるメッセージの処理手順

1. (Response) の数値化である $r$ が下式を満たすことを検査する。

$$rG_S \overset{\mathcal{G}_S}{=} a(S - anm \cdot G_S) + W$$

2. 顕名認証の場合は、$r$ の検証に加えて、Signature の数値化である $s$ が下式を満たすことを検査する。

$$sG_S \overset{\mathcal{G}_S}{=} b(S - anm \cdot G_S) + Q$$

但し、$Q$ は (Qx) [Qy] の $\mathcal{G}_S$ への逆射影であり、$b$ は UA_Response 中の下記の符号のハッシュ値とする。

(EncryptedMask) (Response) (Qx) [Qy]

3. SA_Confirm を *UA* に送付する (8.5.8)。

## 8.5.8 SA_Confirm

### 8.5.8.1 規定

<SA_Confirm> ::= [SessionID]$^{H}$ [MAC]$^{H}$

UA_Witness の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [SessionID] | 固定長 | $o^{H}$ | セッションの識別番号であり、UA_AID のハッシュ値 |
| [MAC] | 固定長 | $o^{H}$ | 鍵 *key* による ASCII 文字列 "SUCCESS" への HMAC |

### 8.5.8.2 *SA* によるメッセージの生成手順

1. *key* を鍵として、ASCII 文字列 SUCCESS の HMAC を計算し、[MAC] に符号化する。

### 8.5.8.3 *UA* によるメッセージの処理手順

1. UA_Confirm を *SP-A* に送付する (8.6.9)。

## 8.6 User Agent/Service Provider Agent インタフェース

### 8.6.1 概要

*UA・SP-A* 間のインタフェースは、権利発行を目的とするメッセージ群と権利認証を目的とするメッセージ群とから構成される。以下、UA_で始まるメッセージは *UA* が送信するメッセージであり、SPA_で始まるメッセージは *SP-A* が送信するメッセージである。

*UA・SP-A* 間の権利発行セッションは、以下のメッセージを以下の順序で交換することで実行される。

| メッセージ | 定義 |
|---|---|
| UA_Setup | *UA* が *aid* の発行を *SP* に要求するのに先立って、*SP-A* 側で新たな $k$ のための鍵レコードの識別子と、$k$ を交換するために *SP-A* が計算する暗号データ $E_T$ の出力を、*SP-A* に要求する。 |
| SPA_Request | *SP-A* は、内部データベースに新たに生成した鍵レコードの識別子と、鍵 $k$ を交換するための暗号データ $E_T$ を指定して送付する。 |
| UA_Issue | *SP* が生成した#識別子、及び、鍵 $k$ を交換するための暗号データ $E_P$ を指定して送信する。*SP-A* は、$E_P$ から $k$ を計算し、#識別子と併せて、鍵レコードに記録する。 |

*UA・SP-A* 間の権利認証セッションは、以下のメッセージを以下の順序で交換することで実行される。

| メッセージ | 定義 |
|---|---|
| UA_Initiate | *SP-A* において使用する鍵レコードの識別子と、*SP* の特権及び認証のための公開鍵を指定して送付する。また、現在のセッションの 2 バイトの識別番号 SessionID を指定する。 |
| SPA_Witness | *SP-A* 生成する仮のウィットネス $W'$ を *UA* に送付する。 |
| UA_Challenge | *SA* が生成したチャレンジ $c$ と、*SA* の認証のために *SP-A* が検証するデータとを送付する。顕名認証、権利更新、鍵転送、権利無効化において必要となる情報も、このメッセージに記載して送付する。 |
| SPA_Response | *SP-A* が生成する仮のレスポンス $r' \in [0, n_S)$ を *UA* に送付する。顕名認証或いは鍵転送のために *SP-A* が生成するデータも、このメッセージに記載して送付する。 |
| UA_Confirm | *SA* による権利認証の結果を *SP-A* に転送する。*SP-A* による、権利消費／権利更新／権利無効に伴う鍵レコードへの変更は、*SP-A* がこのメッセージを確認してから確定する。 |
| SPA_Confirm | *SP-A* は、権利消費／権利更新／権利無効に伴う永続データへの変更を行った後、確認のためにこのメッセージを送付する。更に、権利更新を行った場合は、*aid* の差分をこのメッセージに指定する。 |

## 8.6.2 UA_Setup

### 8.6.2.1 規定

```
<UA_Setup>              ::= EMPTY
```

### 8.6.2.2 *UA* によるメッセージの生成手順

1. *SP-A* から新たな *aid* の発行を要求する際に、*SP-A* から [KeyID] と鍵交換のためのデータ $E_T$ を取得することを目的に *SP-A* に送付する。

### 8.6.2.3 *SP-A* によるメッセージの処理手順

1. SPA_Request を *SP-A* に送付する (8.6.3)。

### 8.6.3 SPA_Request

#### 8.6.3.1 規定

<SPA_Request> ::= [KeyID]$^4$ (ETx)[ETy]$^1$

SA_Call の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [KeyID] | 固定長 | $o^4$ | *SP-A* がこのセッションで交換する鍵 $k$ を記録する鍵レコードの識別子 |
| (ETx) | 可変長 | $o^*$ | $k$ を交換するために *SP-A* が生成する暗号化データ $E_T \stackrel{G_T}{=} \epsilon_T G_T$ の $x$ 座標 |
| [ETy] | 固定長 | $o$ | $E_T$ の $y$ 座標の最下位ビット |

#### 8.6.3.2 *SP-A* によるメッセージの生成手順

1. 現時点で未使用の [KeyID] を選択する。
2. 乱数 $\epsilon_T \in_R [0, n_T)$ を生成し、下式により $E_T$ を計算し、(ETx)[ETy] に符号化する。

$$E_T \stackrel{G_T}{=} \epsilon_T G_T$$

#### 8.6.3.3 *UA* によるメッセージの処理手順

1. UA_Request を *SP* に送付する (8.4.2)。

### 8.6.4 UA_Issue

#### 8.6.4.1 規定

<UA_Issue> ::= [KeyID]$^4$ (Emask) [ID]$^2$ (EPx)[EPy]$^1$ <GroupDef>

SA_Call の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [KeyID] | 固定長 | o$^4$ | *SP-A* がこのセッションで交換する鍵 $k$ を記録する鍵レコードの識別子 |
| (Emask) | 可変長 | $o^*$ | *UA* が匿名化の目的のために生成した乱数 $\epsilon_U$ |
| [ID] | 固定長 | o$^2$ | *SP* が生成する *aid* の#識別子、サービス公開鍵をスコープとして一意に *aid* を決定する |
| (EPx) | 可変長 | $o^*$ | $k$ を交換するために *SA* が生成する暗号化データ $E_P$ の $x$ 座標 |
| [EPy] | 固定長 | $o$ | $E_P$ の $y$ 座標の最下位ビット |
| <GroupDef> | 複合 | – | サービス公開鍵の基礎となる群の定義 |
| [Binary] | 固定長 | $o$ | 基礎群が、標数 2 の楕円曲線の部分群である時に 1 を指定、奇標数の素体上で定義される時に 0 を指定 |
| (Field) | 可変長 | $o^*$ | 楕円曲線が標数 2 の素体の拡大体上で定義されている時には拡大次数を指定、標数が奇数である時には標数を指定 |
| (a) | 可変長 | $o^*$ | 楕円曲線の定義 $y^2 = x^3 + ax + b$(奇標数の場合)、或いは、$y^2 + xy = x^3 + ax^2 + b$ の係数 $a$(標数 2 の場合) |
| (b) | 可変長 | $o^*$ | 楕円曲線の定義 $y^2 = x^3 + ax + b$(奇標数の場合)、或いは、$y^2 + xy = x^3 + ax^2 + b$(標数 2 の場合) の係数 $b$ |
| (Gx) | 可変長 | $o^*$ | 楕円曲線上の基点の $x$ 座標 |
| [Gy] | 固定長 | $o$ | 楕円曲線上の基点の $y$ 座標の最下位ビット |
| (n) | 可変長 | $o^*$ | 基礎群中での基点の位数 |

#### 8.6.4.2 *UA* によるメッセージの生成手順

1. $E_U \overset{\mathcal{G}_T}{=} \epsilon_U G_T + E_T$ の計算に用いた $\epsilon_U$ を (Emask) に符号化する。
2. SP_Issue に指定された [ID](EPx)[EPy]<GroupDef>を接合して、メッセージを完成させる。

#### 8.6.4.3 *SP-A* によるメッセージの処理手順

1. 以下の手順で $k$ を計算する。

$$\begin{aligned} E_U &\overset{\mathcal{G}_S}{=} (\epsilon_U + \epsilon_T)G_S \\ k' &= \pi((\epsilon_U + \epsilon_T + \bar{E}_U\tau)E_P) \\ k &= \mu(k', [\mathrm{ID}]) \end{aligned}$$

$E_P$ は [EPx](EPy) の $\mathcal{G}_T$ への逆射影である。

2. [KeyID] を識別子とする鍵レコードを生成し、[ID]、<GroupDef>及び $k$ を記録する。

### 8.6.5 UA_Initiate

#### 8.6.5.1 規定

```
<UA_Initiate>          ::= [SessionID]^2 [AuthMode]^1 [KeyID]^4 <ApplianceInfo>
                           [Position]^1 [HashedApplianceInfo]^H+
<ApplianceInfo>        ::= [Privileges]^1 (Ax)[Ay]^1
```

UA_Initiate の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [SessionID] | 固定長 | $o^2$ | *UA* が一意に生成する、*UA/SP-A* インタフェースにおけるセッションの識別番号、同一セッション内の全てのメッセージに付与、*SA/UA* インタフェースで利用する SessionID とは異なる |
| [AuthMode] | 固定長 | $o^1$ | SA_Call から *UA* が選択した認証モード |
| [KeyID] | 固定長 | $o^4$ | *aid* に対応して *SP-A* が交換した鍵 $k$ を記録する鍵レコードの識別子 |
| <ApplianceInfo> | 複合 | — | *SA* に許されている認証モードと公開鍵を指定 |
| [Privileges] | 固定長 | o | 認証モードの実行権限の有無 |
| (Ax) | 可変長 | $o^*$ | *SA* の公開鍵 $S$ の $x$ 座標 |
| [Ay] | 固定長 | $o$ | *SA* の公開鍵 $S$ の $y$ 座標 |
| [Position] | 固定長 | o | <ApplianceInfo が引き続く [HashedApplianceInfo]+中で占める位置 |
| [HashedApplianceInfo]+ | 繰返し | — | SP_Issue で受け取った [HashedApplianceInfo]+を同じ順序で指定、各<ApplianceInfo>は以下のフィールドを持つ |

#### 8.6.5.2 *UA* によるメッセージの生成手順

1. 2バイトの [SessionID] を生成する。*UA* と *SP-A* がセッションを識別できることが目的であり、その生成方法は *UA* の実装に依存する。*SA/ UA* インタフェースで利用する H バイトの [SessionID] とは異なる。

2. UA_AID の送信時に選択した権利レコードから [KeyID] 及び [HashedApplianceInfo]+を取得する。

3. SA_Initiate 中の<ApplianceInfo>から計算したハッシュ値と、リスト [HashedApplianceInfo]+のエントリーとを比較し、値が一致するエントリーの位置を [Position] に符号化する。位置は、先頭のエントリーを 1 とした出現順序とする。

#### 8.6.5.3 *SP-A* によるメッセージの処理手順

1. <ApplianceInfo>から計算したハッシュ値と、リスト [HashedApplianceInfo]+中で位置が [Position] で特定されるエントリーとが一致することを検査する。

2. [AuthMode] ビット毎否定 (bit-wise negation) と、UA_Initiate 中の<ApplianceInfo>に指定される [Privilege] とのビット毎論理和 (bit-wise OR) を計算して、結果が 255 になることを検査する。

3. [KeyID] を鍵として内部データベースから鍵レコードを検索し、$(\mathcal{G}_S, G_S, n_S)$ (UA_Issue 中の <GroupDef>)、#識別子 (UA_Issue 中の [ID])、及び、$k$ を取得する。

4. SPA_Witness を *UA* に送付する (8.6.6)。

### 8.6.6 SPA_Witness

#### 8.6.6.1 規定

```
<SPA_Witness>          ::= [SessionID]^2 [AuthMode]^1<Witness><QWitness>?
<Witness>              ::= (W'x)[W'y]^1
<QWitness>             ::= (Q'x)[Q'y]^1
```

**[AuthMode] のビットと省略可能フィールドの対応**

(MSB)<QWitness>

UA_Witness の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [SessionID] | 固定長 | $o^2$ | セッションの識別番号 |
| <Witness> | 複合 | — | *SP-A* が生成する仮のウィットネス $W'$ |
| [AuthMode] | 固定長 | $o^1$ | UA_Initiate で受け取った [AuthoMode] |
| (W'x) | 可変長 | $o^*$ | $W'$ の $x$ 座標 |
| [W'y] | 固定長 | $o$ | $W'$ の $y$ 座標の最下位ビット |
| <QWitness> | 複合 | — | *SP-A* が生成する仮のウィットネス $Q'$ |
| (Q'x) | 可変長 | $o^*$ | $Q'$ の $x$ 座標 |
| [Q'y] | 固定長 | $o$ | $Q'$ の $y$ 座標の最下位ビット |

#### 8.6.6.2 *SP-A* によるメッセージの生成手順

1. 乱数 $w' \in [0, n_S)$ を生成し、$W'$ を下式により計算して、<Witness>に符号化する。

$$W' \stackrel{G_S}{=} w'G_S$$

2. 顕名認証の場合は、乱数 $q' \in [0, n_S)$ を生成し、$Q'$ を下式により計算して、<QWitness>に符号化する。

$$Q' \stackrel{G_S}{=} q'G_S$$

#### 8.6.6.3 *UA* によるメッセージの処理手順

1. UA_Witness を *SA* に送付する (8.5.5)。

### 8.6.7 UA_Challenge

#### 8.6.7.1 規定

```
<UA_Challenge> ::= [SessionID]^2 [AuthMode]^1(Challenge) [HashedRules]^H
                   <Seed>?(RRL)?[HashedNewRules]^H?[KeyCert]^H
                   <Masks>?<Test>?(Wmask)
<Seed>         ::= (Cx)[Cy]^1
<Masks>        ::= (AIDmask)(Qmask)
<Test>?        ::= (Ux)[Uy]^1
```

**[AuthMode] のビットと省略可能フィールドとの対応**

(MSB)<Masks>, <Test> (-1) (-2) [HashedNewRules] (-3) <Seed>, <Test> (-4) (RRL)

getResponseInput の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [SessionID] | 固定長 | $o^2$ | セッションのの識別番号 |
| [AuthMode] | 固定長 | $o^1$ | UA_Initiate に指定した AuthMode |
| (Challenge) | 可変長 | $o^*$ | *SA* が *UA* に送るチャレンジ $c \in [0, n_S)$ |
| [HashedRules] | 固定長 | $o^H$ | この認証イベントで有効な*認証子であり、*SA* が計算した (Rules) のハッシュ値 |
| [HashedNewRules] | 固定長 | $o^H$ | 権利更新における新しい*認証子であり、*SA* が生成した新たな (NewRules) のハッシュ値 |
| <Seed> | — | — | 鍵転送における転送鍵計算のためのシード $C \in \mathcal{G}_S$ |
| (Cx) | 可変長 | $o^*$ | $C$ の $x$ 座標 |
| [Cy] | 固定長 | $o$ | $C$ の $y$ 座標の最下位ビット |
| (RRL) | 可変長 | $o^*$ | 権利無効化における無効化対象となる権利の#識別子のリスト |
| [KeyCert] | 固定長 | $o^H$ | *SA* と *SP-A* で交換する *key* のハッシュ値 |
| <Masks> | — | — | 顕名認証時に *SP-A* に送付する $\rho$ 及び $q''$ |
| (AIDmask) | 可変長 | $o^*$ | *UA* が匿名化の目的のために生成した乱数 $\rho \in [0, n_S)$ |
| (Qmask) | 可変長 | $o^*$ | *UA* が匿名化の目的のために生成した乱数 $q'' \in [0, n_S)$ |
| <Test> | — | — | 顕名認証或いは鍵転送時に *SP-A* によるメッセージを検証するための暗号化データ |
| (Ux) | 可変長 | $o^*$ | $U \overset{\mathcal{G}_S}{=} xG_S + yQ + zC$ の $x$ 座標 |
| [Uy] | 固定長 | $o$ | $U \overset{\mathcal{G}_S}{=} xG_S + yQ + zC$ の $y$ 座標の最下位ビット |
| (Wmask) | 可変長 | $o^*$ | *UA* が匿名化の目的のために生成した乱数 $w'' \in [0, n_S)$ |

#### 8.6.7.2 *UA* によるメッセージの生成手順

1. SA_Challenge に以下の変更を加える。
   - Hバイトの [SessionID] を *UA/SP-A* 間のインタフェースにおける2バイトの [SessionID] に置き換える。
   - (NewRules) のハッシュ値([HashedNewRules])を計算し、(NewRules) に置き換える。
   - <Masks>?<Test>?(Wmask) を末尾に付け加える。

2. 顕名認証の場合は、$q'' \in [0, n_S)$ をランダムに生成し、$\rho$ 及び $q''$ を (AIDmask) 及び (Qmask) に符号化し、<Masks>を生成する。また、$Q$ は以下のように計算する。

$$Q \stackrel{\mathcal{G}_S}{=} Q' + q''G_S \stackrel{\mathcal{G}_S}{=} (q' + q'')G_S$$

3. 顕名認証或いは鍵転送の場合は、$x, y, z$ を以下のように計算する。

   - $x \in [0, n_S)$ をランダムに生成する。
   - 匿名認証の場合は $y = 0$ とし、顕名認証の場合は $y \in [0, n_S)$ をランダムに生成する。
   - 鍵転送を行わない場合は $z = 0$ とし、鍵転送を行う場合は $z \in [0, n_S)$ をランダムに生成する。

   下式により $U$ を計算し、(Ux) [Uy] に符号化する。

$$U \stackrel{\mathcal{G}_S}{=} xG_S + yQ + zC$$

4. (Wmask) は $w''$ の符号化である。

#### 8.6.7.3 *SP-A* によるメッセージの処理手順

1. SPA_Response を *UA* に送付する (8.6.8)。

### 8.6.8 SPA_Response

#### 8.6.8.1 規定

```
<SPA_Response>        ::= [SessionID][2][AuthMode][1]
                          (Signature)?<EncryptedSeed>?(EncryptedMask)?(Response)<EncryptedTest>?
<EncryptedSeed>       ::= (R'x)[R'y][1]
<EncryptedTest>       ::= (Vx)[Vy][1]
```

**[AuthMode] のビットと省略可能フィールドとの対応**

(MSB)(Signature), <Evidence>, <EncryptedTest>　　(-1) <EncryptedSeed>, <EncryptedTest>

getResponseInput の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [SessionID] | 固定長 | $o^2$ | セッションの識別番号 |
| (Signature) | 可変長 | $o^*$ | $Q$ と併せて $r \mid e_P$ への署名を構成 |
| <EncryptedSeed> | — | — | 転送鍵計算のためのシード $C$ の暗号化 $R \in \mathcal{G}_S$ |
| (R'x) | 可変長 | $o^*$ | $R' \overset{\mathcal{G}_S}{=} \mu(k,*)C$ (匿名)、或いは、$R' \overset{\mathcal{G}_S}{=} (\mu(k,*)+\rho)C$ (顕名) の $x$ 座標 |
| [R'y] | 固定長 | $o$ | $R'$ の $y$ 座標の最下位ビット |
| (EncryptedMask) | 可変長 | $o^*$ | $\pi((\mu(k,*)+\rho)Q)\oplus\rho$ で計算され、$Q$ と併せて $\rho$ の ElGamal 暗号を構成 |
| (Response) | 可変長 | $o^*$ | *SP-A* が *UA* に送るレスポンス $r' \in [0, n_S)$ |
| <EncryptedTest> | — | — | *UA* が検証のために送付した $U$ の暗号化 $V \in \mathcal{G}_S$ |
| (Vx) | 可変長 | $o^*$ | $V \overset{\mathcal{G}_S}{=} \mu(k,*)U$ (匿名)、或いは、$V \overset{\mathcal{G}_S}{=} (\mu(k,*)+\rho)U$ (顕名) の $x$ 座標 |
| [Vy] | 固定長 | $o$ | $\mathcal{V}$ の $y$ 座標の最下位ビット |

#### 8.6.8.2 *SP-A* によるメッセージの生成手順

1. (Response) は、下式で定義される $r'$ の符号化である。

   **匿名認証の場合** $r' = a\mu(k,*) + w' + w'' \bmod n_S$

   **顕名認証の場合** $r' = a(\mu(k,*) + \rho) + w' + w'' \bmod n_S$

   但し、$*$ は、下記の符号とする。

   $*$ = [HashedRules][HashedApplianceInfo]+

   - $w''$ は UA_Challenge 中の (Wmask) の数値化である。
   - $W$ は下式で計算される。

     $$W \overset{\mathcal{G}_S}{=} (w' + w'')G_S$$

   - $a$ は、$W$ の符号化 (Wx)[Wy][1] と UA_Challenge の内容の一部を接合して得られる下記の符号のハッシュ値とする。

     (Wx)[Wy][AuthMode](Challenge)[HashedRules]<Seed>?(RRL)?[HashedNewRules]?

- $k$ は UA_Issue 中の [KeyID] により特定される秘密鍵である。
- $\rho$ は UA_Challenge 中の (AIDmask) の数値化である。

2. 顕名認証の場合は、$e_P$ 及び $s$ を下式に従って計算し、それぞれ、(EncryptedMask) 及び (Signature) に符号化する。

$$Q \overset{G_S}{=} (q' + q'')G_S$$

$$e_P = \pi((\mu(k,*) + \rho)Q) \oplus \rho$$

$$s = b(\mu(k,*) + \rho) + q' + q''$$

- $q''$ は UA_Challenge 中の (Qmask) の数値化である。
- $b$ は以下の符号のハッシュ値。
  (EncryptedMask)(Response)(Qx)[Qy]
  但し、(Qx)[Qy] は $Q$ の符号化。

3. 顕名認証或いは鍵転送の場合は、匿名認証・顕名認証の別により、下式に従って $V$ を計算し、(Vx)[Vy] に符号化する。

**匿名認証の場合** $V \overset{G_S}{=} \mu(k,*)U$
**顕名認証の場合** $V \overset{G_S}{=} (\mu(k,*) + \rho)U$

4. 鍵転送を実行する場合は、匿名認証・顕名認証の別により、下式に従って $R'$ を計算し、(Rx)[Ry] に符号化する。

**匿名認証の場合** $R' \overset{G_S}{=} \mu(k,*)C$
**顕名認証の場合** $R' \overset{G_S}{=} (\mu(k,*) + \rho)C$

#### 8.6.8.3 *UA* によるメッセージの処理手順

1. $r'$ が下式を満たすことを検証する。

**匿名認証の場合** $r'G_S \overset{G_S}{=} a(S - aid \cdot G_S) + W$
**顕名認証の場合** $r'G_S \overset{G_S}{=} a(S - anm \cdot G_S) + W$

2. 顕名認証の場合は、$r'$ の検証に加えて、下式を検査する。

$$sG_S \overset{G_S}{=} b(S - anm \cdot G_S) + Q$$

3. 顕名認証或いは鍵転送の場合は、下式を検査する。

**匿名認証の場合** $V \overset{G_S}{=} x(S - aid \cdot G_S) + zR'$
**顕名認証の場合** $V \overset{G_S}{=} x(S - anm \cdot G_S) + y\pi^{-1}(e_P \oplus \rho) + zR'$
但し、鍵転送を行わない場合は $z = 0$。

4. UA_Response を *SA* に送付する (8.5.7)。

### 8.6.9 UA_Confirm

```
<SA_Confirm>          ::= [SessionID]^2 [Confirm]^H
```

UA_Confirm の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [SessionID] | 固定長 | $o^2$ | セッションの識別番号 |
| [Confirm] | 固定長 | $o^H$ | 鍵 *key* による ASCII 文字列 SUCCESS への HMAC |

#### 8.6.9.1 *UA* によるメッセージの生成手順

1. SA_Confirm の [SessionID] を、*UA/SP-A* 間インタフェースの 2 バイトの [SessionID] に置き換える。

#### 8.6.9.2 *SP-A* によるメッセージの処理手順

1. ASCII 文字列 SUCCESS への HMAC を *key* を鍵として計算し、UA_Confirm に指定された [Confirm] と一致することを検査する。
2. SPA_Confirm を *UA* に送付する (8.6.10)。

### 8.6.10 SPA_Confirm

<SA_Confirm> ::= [SessionID][2] [AuthMode][1](Difference)?

**フィールドと Finger_Print のビットとの対応**

(MSB)(Difference)

UA_Confirm の定義

| フィールド名 | 型 | 符号化 | 定義 |
|---|---|---|---|
| [SessionID] | 固定長 | o[2] | セッションの識別番号 |
| (Difference) | 可変長 | o* | 権利消費及び権利更新時において、*aid* を更新するための差分 $\Delta \in [0, n_S)$ |

#### 8.6.10.1 *SP-A* によるメッセージの生成手順

1. 権限消費を行う場合、このセッションで有効な [KeyID] 及び $k$ を含む鍵レコードをデータベースから消去する。
2. 権限無効化を行う場合、このセッションで有効な#識別子が、入力された (RRL) 中に指定されているかを検査し、指定されている場合は現在のセッションで有効な鍵レコードをデータベースから消去する。
3. 鍵更新を行う場合は、新たに乱数 $\bar{k} \in \{0,1\}^{\mathrm{H}}$ を生成し、下式により $\Delta$ を計算して、(Difference) に符号化する。

   $$\Delta = \mu(k, *) - \mu(\bar{k}, \bar{*}) \bmod n_S$$

   更に、このセッションで有効な $k$ を含む鍵レコードにおいて、$k$ の値を $\bar{k}$ に置き換える。

   - $*$ は、下記の符号とする。

     $*$ = [HashedRules][HashedApplianceInfo]+

   - $\bar{*}$ は、下記の符号とする。

     $\bar{*}$ = [HashedNewRules][HashedApplianceInfo]+

#### 8.6.10.2 *UA* によるメッセージの処理手順

**手順**

1. 下式により、*aid* を更新する。

   $$aid_{\mathrm{new}} = aid + \Delta \bmod n_S$$

2. 現在のセッションで有効な *aid* を含むレコードにおいて、*aid* を $aid_{\mathrm{new}}$、(Rules) を (NewRules) で置き換える。

# 第9章　デジタルコンテンツ流通への応用

## 9.1 デジタルコンテンツ流通における相互運用性の課題

### 9.1.1 ユビキタスコンピューティングとデジタルコンテンツ流通

ユビキタスコンピューティングでは、多様なサービスがドメインを横断してシームレスに提供されるが、以下の点において、デジタルコンテンツ流通はユビキタスコンピューティングで提供されるサービスとしては格段に複雑な様相を内包する例である。

- デジタルコンテンツはドメインを横断してワールドワイドに提供される。
- デジタルコンテンツのオーナは、所有するコンテンツの著作権保護の観点から、コンテンツの利用環境に強い関心を抱いている。
- デジタルコンテンツの利用環境は、デジタルコンテンツのオーナとは全く独立に提供されるだけでなく、極めて多様である。更に、デジタルコンテンツのオーナと利用環境との間には、部分的には利害が相反するため、権限発行者と検証者との間の分離が強く成り立つ。
- ユーザは、デジタルコンテンツの利用においてプライバシーが守られることを既得の権利であると意識しており、高いレベルでの匿名性と追跡不能性を要求する。
- デジタルコンテンツ流通では、多様なビジネスモデル (*e.g.* Try-Before-Purchase, Pay-Per-View) が既に存在し、複雑なアクセス制御が求められる。

このため、本論文では、アクセス制御プロトコルを設計するにあたって、デジタルコンテンツ流通で求められる要件を満足することをひとつの指標としてきた。

更に、デジタルコンテンツ流通は、本研究の適用例として、短期的に実用的なインパクトが期待できる点でも重要である。

しかしながら、現状のデジタルコンテンツ流通には、コンテンツ保護機能と決済・著作権処理機能との間の相互運用性に関して課題があり、本研究の適用を考えるためにはこの課題の解決を避けて通れない。

この章では、上記の相互運用性に関する課題について考察し、課題を解決する方法として**ブリッジレイヤ**を提案する。

### 9.1.2 コンテンツ保護機能と決済・著作権処理機能との間の相互運用性の阻害

近年目覚しいブロードバンドの普及により，高品質コンテンツ流通のための環境が整いつつある．その一方で，コンテンツの不正利用の危険が強く指摘されており，コンテンツ保護のための多くの方式が提案されているが，その殆どはコンテンツの種別や処理アプリケーション群毎に固有のコンテンツアーキテクチャ（以下，CA）に組み込まれて提供される点で相互運用性はない [42]．コンテンツ流通の基盤構築という観点からは多くのコンテンツ保護機能が徒に分立する現状は望ましくないが，標準化による解決を目指す SDMI [58] などの動きも，著作権者・著作権団体・流通業者・機器製造業者などの異なる利

益代表の足並みが揃わない．更に，多くの企業が独自に製品を導入している現状や，統制に対するユーザの根強い警戒感を考え併せると，共通基盤の実現にはまだ時間が必要である．

一方，デジタルコンテンツの流通ビジネスの成立に不可欠の要素である決済及び著作権処理サービス（以下，CHS）は，標準化や基盤化によってのみ解決し得る問題を孕んでいるため，CA を横断する汎用の機能としての提供が望まれている [49, 30, 48, 31, 42]．

このように，CHS が汎用化される一方で CA が分立せざるを得ない状況では，CHS と CA との連携が重要な課題となる．これは，技術的課題であるだけでなく，ビジネスとしてのコンテンツ流通に影響を与える重要性を有する．すなわち，コンテンツ流通ビジネスの事業者によるアドホックな連携は，セキュリティ上の脆弱性が紛れ込む蓋然性を高めると同時に，相互運用を阻害し，ために開発・維持のためのコストが当事者の重荷となり，連携への動機付けを希薄化する．

本稿では，CHS と CA の連携の統一的なフレームワークとして，ブリッジレイヤを提案する．ブリッジレイヤは CHS と CA のレイヤの中間に位置する．任意の CHS と CA との間の連携は，それぞれがブリッジレイヤに準拠することのみで実現されるので，開発・維持コストが低減され，相互運用性と連携への動機付けを高める．また，連携における安全性は，CHS 及び CA とブリッジレイヤとの間のプロトコル及びブリッジレイヤの実装の安全性に集約されるため，脆弱性の混入の危険を低減することが可能となる．

加えて，ブリッジレイヤは，現行の CHS と CA に許された拡張性のみを利用し，各々のアーキテクチャの再構築を要求することなく連携を実現できることを要件の一つとする．アーキテクチャの再構築は安全性の観点からは重要であるが，それを前提とした解決策は現実性に乏しい．

本稿では，ブリッジレイヤに求められる要件を明らかにし，併せて実現のための方式を提案する．

個別の CA を横断する汎用の CHS では，著作権及び決済に関する規則をメタデータ（例えば [48] など）や権利記述言語（例えば [31] など）を用いて記述する．本稿ではこれらの規則を**ルール**と呼ぶ．ルールは，コンテンツに依存しない生成及び解釈の規則に従う．

一方，CHS の構成としては，ユーザサイドのクライアントプログラムとサーバとが協力してルールを処理する構成が最も典型的である．例えば，MMG [49] では，クライアントプログラムは，許諾管理サーバと呼ばれるインターネット上のサーバにルールを送付し，許諾管理サーバによる処理結果の通知を受けて，コンテンツの利用を制御する．

CHS の基本的な機能は以下の 3 項目である．

**著作権処理** メタデータ及び権利記述言語は，著作権関連の法律への準拠に加えて，著作権団体との連携のための汎用性を備えるため，標準化が求められる．

**決済機能** 銀行やクレジットカード会社が提供する既存の決済サービスへの接続性が求められる．

**実世界の契約モデルのサポート** ルールは現実の契約モデルを忠実に記述する能力を備える必要がある．

更に，CHS の処理には安全性が要求される．例えば，ルールの改ざん検知やクライアントプログラムとサーバ間の通信における認証・機密性・完全性などであり，そのため，CHS は独自にセキュリティ機能を具備するのが通例である．しかしながら，CHS による決済・著作権処理と CA におけるコンテンツ処

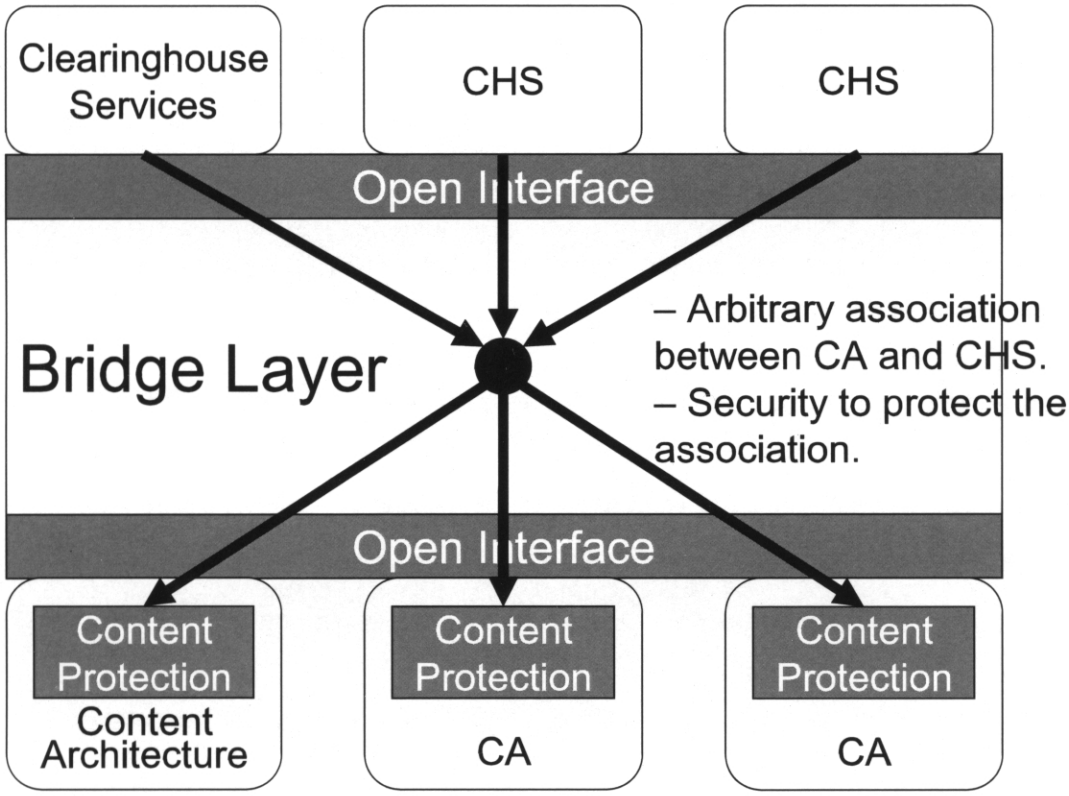



図 9.1: ブリッジレイヤ

理との間に攻撃者が割り込む脅威に対しては，CHS ではこれを排除する仕組みを提供し得ない．コンテンツ保護の仕組みが個別の CA に依存しているからである．

CA 固有のコンテンツ保護では，暗号化と鍵管理が技術的な核となる．

例えば，DVD 及び SD カードで広く利用されている CPPM [1] 及び CPRM [3] では，認定されたメディア（DVD，SD カード）とデバイス（DVD プレイヤ，ドライブ）との組み合わせでのみ，コンテンツ鍵の利用を可能とする仕組みを提供する．

また，Acrobat では，PDF の暗号化に用いるコンテンツ鍵を，ファイル作成者が指定するパスワードを鍵として暗号化し，ファイル中に埋め込む．暗号化された PDF ファイルを表示する際には，Acrobat の標準のセキュリティハンドラが，ユーザが入力したパスワードを用いて，コンテンツ鍵を復号する．

CHS との連携が課題を孕んでいることから，多くの CA は独自の決済や著作権処理の機能をコンテンツ保護の仕組みと一体化して提供する．SD カード [2] では，著作権管理の基本要素であるコンテンツのチェックイン/アウト機能 [58] やデバイスのリボケーション機能を提供し，Adobe 社は Acrobat による文書の販売手段として WebBuy プラグインを提供する [4]．また，主として移動体通信網上で流通するコンテンツの著作権保護を狙い，OMA は，権利記述言語からシステムまでの一貫した仕様を発表している [54, 53].

しかしながら，広くビジネスに適用するためには，標準化されたルールのサポートが重要であり，CA固有の決済・著作権処理機能では限界が見えている．そのため，CA が CHS との連携を目的の一つとした拡張性をサポートする例もある．例えば，CPRM では，認定機関が認定した企業に対してデバイス鍵を発行することで，企業が独自に CHS との連携を実現する手段を与えている．また，Adobe 社からライセンスを取得することにより，Acrobat にコンテンツ鍵を転送する I/F の開示を受け，独自のセキュリティハンドラを開発することが可能である．しかし，安全性の理由から，拡張性の利用はライセンスなどにより制限されることが通例である．

現状では，CHS と CA の連携のためには，それぞれの拡張性を利用して，個別に連携を実現する方法が唯一の解である．この方法は，以下の問題を孕んでいる．

1. $m$ 例の CHS と $n$ 例の CA との連携を可能とするためには，$m \times n$ の拡張が必要であり，開発と維持（バージョンアップへの対応など）で高コストとなる．
2. 1 回の拡張で接続される相手は 1 つに限定され，事業者にとって拡張を行う動機付けが希薄である．
3. 連携には，セキュリティの観点から周到な設計と実装が必要であり，難度の高い開発を伴う．
4. CA の拡張性の利用には，ライセンス取得が必要であり，開発及び展開に制約がある．

これらの問題は，CHS と CA との連携を停滞させ，音楽以外の分野でのデジタルコンテンツ流通の普及を阻害する重大な要因となっている．勿論，法制度も含めて，デジタルコンテンツの著作権保護に関する社会的な合意の形成は必須の条件であるが，音楽配信では技術が社会的合意の形成を促した経緯を考えると，上記の問題を解決・改善する技術的フレームワークを提案することには十分な意義があると考える．

### 9.1.3 Jamkhedkary 等による階層モデルの課題

著作権保護技術の相互運用性に関しては，Jamkhedkary [42] などの研究がある．[42] は，OSI のネットワーク階層モデルに倣って，著作権管理技術を上位の Application/Negotiation Layer，中位の Rights Expression/Interpretation Layer，そして，下位の Digital Rights Enforcement 及び Physical Layer に層別し，上位及び下位レイヤがそれぞれ中位レイヤに対する相互運用性を確保することにより，全体の相互運用性を確保する考え方を示している．

以下で，[42] と本稿の提案内容とを比較する．

[42] ではコンテンツ処理と暗号処理は上下に分離して定義されているの対し，本稿では，コンテンツ処理と暗号処理とを CA レイヤに定義し，[42] の中位レイヤに相当する CHS レイヤとの間に両者の連携の標準化のみを目的としてブリッジレイヤを定義する．

この相違は，以下の理由による．

[42] に従えば，アプリケーションは権利記述言語の処理モジュールを介して隠蔽されたコンテンツ保護機能（暗号機能など）を利用することとなる．これは，暗号機能などのコンテンツ保護機能をアプリケー

ションに不可分に組み込むことで安全性を保証しようとする現状のCAのアーキテクチャに対して，大きな変更を要求することとなる．また，相互運用性確保のためにはCHSの基幹機能である権利記述言語処理の標準化が前提であるので，複数の権利記述言語の開発が推進されている現状との間には少なからぬギャップがある．

対照的に，本稿は，相互運用性を早期に確立するためには，現行のアーキテクチャを許容する必要があるとの立場に立つ．ブリッジレイヤは，CHSやCAのアーキテクチャの再構築を要求することなく，既存の拡張性を利用するだけで，CHSとCAの連携を実現する仕組みを標準化することを狙いとする．

また，本稿では，コンテンツ処理と暗号化処理とを一体化或いは隣接して実装することは，技術的観点からも安全性の観点からも合理的であり，両者を同一のレイヤに定義するべきであると考える．例えば，ハードウェアとしてコンテンツ保護を実現しようとする場合，コンテンツ処理のLSIと暗号化処理のLSIとを一体化或いは隣接して実装することで，バス上でコンテンツなどが窃取されることを防ぐことができる．

## 9.2 ブリッジレイヤによる解決の提案

### 9.2.1 ブリッジレイヤの基本設計

ブリッジレイヤは，9.1.2 に述べた問題を解決或いは改善するため，以下に述べる外的要件を満たす．

1. CHS 及び CA がブリッジレイヤに準拠すれば，両者の間では自動的に相互接続性が保証される．
2. 連携において CHS の処理と CA の処理とが正しく対応することを保証し，CHS と CA の間への割り込みによる改ざんや成りすましの攻撃を排除する．特に，CHS 及び CA がともにブリッジレイヤに準拠すれば，(CHS・CA の脆弱性に起因する場合を除いて) 連携の安全が保証される．
3. CHS・CA のブリッジレイヤへの準拠は，既存の拡張性のみを仮定する．
4. ブリッジレイヤへの準拠に必要な技術情報をすべて開示する．

これらの要件を満たすことにより，以下のように，9.1.2 に述べた問題を解決或いは改善することができる．

**開発・維持コストの低減** 開発・維持コスト上の問題は，例えば，コンテンツの配信事業者が複数の CHS と CA とを連携させるには，両者の組み合わせの数だけ連携システムを開発・維持する必要があるという点に由来する．一方，本稿の提案では，CHS と CA の連携を実現するためには，各々がブリッジレイヤに準拠するだけで十分であることから，連携のための開発・維持のコストを数分の一以下に抑制することができる．

**連携の動機付け** 更に、ブリッジレイヤは，CHS 事業者と CA ベンダに連携のためのコストを負担する動機付けを与えることを狙いの一つとする．その結果，コンテンツビジネスの事業者が負担から解放され，コンテンツビジネスへの参入障壁は大幅に緩和されるとともに，ライセンスに関する問題も解決される．

CHS 事業者の立場からは，連携するべき CA が数多く存在する場合，費用対効果が不明確であることから，自ら開発コストを負担して多数の連携システムを開発することは難しい．対照的に，ブリッジレイヤが導入されると，ブリッジレイヤへの準拠のためのコストを負担すれば，ブリッジレイヤに準拠しているすべての CA との連携が保証されることとなる．連携により CHS の利用機会が増大するので，開発・維持に関わる費用対効果が明確となる．

CHS が CA と連携するための拡張性を有しているのに対し，CA に許される拡張の幅は狭く，といって，アーキテクチャの変更は CA ベンダが連携システムを開発する上での最大の阻害要因となる．ブリッジレイヤは，準拠のために既存の拡張性のみを仮定するので，前記の阻害要因を排除する．

**セキュリティの向上** CHS と CA との連携では，割り込みによる攻撃に対抗する機能を具備することが必須となる．連携システムに脆弱性が存在すると，コンテンツが制限を越えて不正に利用されたり，最悪の場合，コンテンツの平文データが窃取されてしまう．

アドホックな連携の場合，開発者が多様化することからシステムの安全性がばらつき，加えて，多くの実装が存在するのでシステムに脆弱性が混入する蓋然性が飛躍的に高まる．対照的に，ブリッ

ジレイヤでは，レイヤ間のプロトコル設計及びブリッジレイヤの実装を集約するので，脆弱性混入の蓋然性は小さくなる．

### 9.2.2 ブリッジレイヤの技術的要件

この節では，主としてセキュリティの観点から，ブリッジレイヤに求められる技術要件を整理する．

ブリッジレイヤは，コンテンツ毎にCHSとCAとの連携を記述するファイルと該ファイルを処理するプログラムとから構成される．本稿では，前者をブリッジファイル，後者をブリッジングプログラムと呼ぶ．

ブリッジファイルは，決済・著作権処理のためのルールの記述，CAを特定する識別情報，及び，コンテンツを特定する情報（コンテンツデータの指定も含む）の三つ組を内容として持ち，コンテンツをユーザに供給するコンテンツプロバイダによって作成される．

一方，ブリッジングプログラムは，ユーザがコンテンツを利用する機器（PCなど）上で動作し，ブリッジファイルの処理に当たって以下の機能を提供する．

1. ブリッジファイルの検証
2. CHSとの連携による決済・著作権処理の実行
3. コンテンツへのアクセス制御
4. CAへのコンテンツのエクスポート

以下では，ブリッジングプログラムの機能毎に，ブリッジレイヤに求められる技術要件を整理する．

#### 9.2.2.1 ブリッジファイルの検証

ブリッジファイルに対する改ざん検知及び作成者認証を行うためには，電子署名の利用が適切な技術選択であるが，CHSが随時ルールを更新することを許さなければならないので，一律にコンテンツプロバイダが署名を施せばよいわけではない．永続的なルールと変動性のあるルールとを分離し，前者にはコンテンツプロバイダが，後者には更新の都度CHSが署名を施すようにする仕組みを用意する必要がある．

一方，署名は，永続的なルールと変動性のあるルールに各々について改ざんを防止するが，変動ルールを署名ごと置き換える攻撃の可能性を排除しない．したがって，以下を必須の技術要件としなければならない．

**偽造への対処** 正当でない署名者による署名を排除するには，署名検証鍵を検証する必要がある．ブリッジファイル中のルールの処理をどのCHSに委ねるべきかは，コンテンツプロバイダのみが決定し得る事項なので，ブリッジングプログラムはコンテンツプロバイダが署名検証鍵を認定していることを検証する．

**リプレイ攻撃への対処** CHS は多くの変動ルールに署名するため，正当に署名された別の変動ルールへのすり替え，所謂，リプレイ攻撃の危険が残る．リプレイ攻撃を防止するために，CHS が署名するデータにセッションの識別データ（使い捨て乱数など）とブリッジファイルの識別データ（ハッシュなど）とを含める．

### 9.2.3 決済・著作権処理

ブリッジングプログラムは，ブリッジファイルに指定されるルールを CHS に送付すると同時に，ルールの処理結果を CHS から受け取る必要がある．CA によるコンテンツの処理は，決済・著作権処理が正しく実行された場合に限り実行されなければならないからである．この際，ブリッジングプログラムは，以下を検証する必要がある．

a. 通信相手の CHS がコンテンツプロバイダによって承認されている．

b. CHS に送付したデータが改ざん・リプレイ攻撃に晒されていない．

c. CHS から受信したデータが改ざん・リプレイ攻撃に晒されていない．

通信のセキュリティでは通信相手の認証，通信内容の機密性と完全性の 3 項目が要求されるが，（a）は認証，（b）及び（c）は完全性に相当する．ルールに機密性が要求される場合は，CHS において暗復号の機能が用意されるので，ここでは機密性を考慮する必要はない．

### 9.2.4 コンテンツのアクセス制御

ブリッジレイヤにとって最も重大な脅威は，決済・著作権処理を迂回したコンテンツの不正利用にある．言い換えるならば，ブリッジングプログラムによる正当な処理を経ることなく，CA にコンテンツの処理を許すことがあってはならない．

この目的のために，ブリッジングプログラムのみが復号できるようにコンテンツを暗号化することとなるが，公開情報のみに基づいてブリッジレイヤが利用可能であるべきことを考えると公開鍵暗号の利用が必然となる．すなわち，暗号化にはブリッジングプログラムの公開鍵を利用し，復号にはブリッジングプログラムが排他的にアクセスできるプライベート鍵を利用する．

更に，公開鍵暗号の標準的な利用法では，大容量のコンテンツデータは効率のよい共通鍵暗号で暗号化し，公開鍵では共通鍵を暗号化するが，ブリッジレイヤでもこの方法を踏襲するべきである．この時，処理効率の点から，CA が暗復号機能をサポートしているか否かによって，ブリッジングプログラムの公開鍵で暗号化する対象を変えることが望ましい．

- CA において暗復号機能をサポートしない場合には，必然的にブリッジングプログラムがコンテンツの暗復号を実行することとなる．この場合，コンテンツの暗復号に用いる鍵（以下，**コンテンツ鍵**）を，ブリッジングプログラムの公開鍵で暗号化する．

- CA において暗復号が実行される場合には，可能である限り，ブリッジレイヤにおいてコンテンツを暗号化せず，CA に暗復号を任せる方が冗長性がなく処理効率がよい．この場合は，CA による暗復号で必須となる鍵データ（以下，**CA 固有鍵**）をブリッジングプログラムの公開鍵で暗号化する．

### 9.2.5 CA へのコンテンツのエクスポート

9.2.4 で述べたように，ブリッジングプログラムにおいてコンテンツ或いはコンテンツ鍵/CA 固有鍵の復号処理がなされない限り，CA はブリッジングファイルを処理することはできない．復号されたコンテンツ或いはコンテンツ鍵/CA 固有鍵を CA に供給するプロセスを，コンテンツのエクスポートと呼ぶ．

エクスポートの実現方法には CA の仕様に依存して制約があり，安全性にも差異がある．例えば，ファイル渡しなど，OS の標準的な機能を利用したエクスポートは，ほぼ全ての CA の実装に適用できる利点はあるが，安全性は低い．一方，CA が用意する API を利用するエクスポートは，安全性は高いが，CA が API を用意することが前提である．ブリッジングプログラムは，複数のエクスポートの方式をサポートし，CA の仕様に応じて適切なエクスポート方式を選択して実行できる必要がある．

### 9.2.6 ブリッジファイルの XML 定義

この節では，前節で整理したブリッジレイヤへの技術要件を満足する具体的な方式を提案する．

以下において，ブリッジファイルを XML Schema [66] によって定義し，ブリッジングプログラムと CHS 及び CA とのプロトコルを規定する．また，暗号化及び署名の書式に関しては，XML Encryption [65] 及び XML Signature [67] に従い，以下の名前空間を用いる．

```
<schema
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xenc="http://www.w3.org/2001/049/xmlenc#"
/>
```

ブリッジファイルを，ルールなどを指定する Header 要素とコンテンツとの関連付けを指定する Body 要素とから，更に，Header を ConstantPart 要素と VariablePart 要素とから構成する（図 9.2）．ConstantPart には永続的なルールを指定してコンテンツプロバイダによる署名を付し，VariablePart には変動性のあるルールを指定して CHS による署名を付す．ConstantPart は BodyDigest 要素（図 9.3）に Body のダイジェスト値を含むので，ConstantPart に対する署名は Body に対する署名を兼ねる．また，ルールは<xsd:any namespace="##other"/>のインスタンスとして指定される．

9.2.2.1 で述べた技術要件を以下のように満足する．

**偽造の防止** ブリッジングプログラムは VariablePart の署名の検証に ConstantPart の CHS-Keys 要素（図 9.3）に指定される署名検証鍵を用いる．ConstantPart にはコンテンツプロバイダの

```
<xsd:element name="Header">
  <xsd:complexType content="element">
    <xsd:sequence>
      <xsd:element ref="ConstantPart"/>
      <xsd:element ref="ds:Signature"/>
        <!-- Signature for ConstantPart -->
      <xsd:element ref="VariablePart"/>
      <xsd:element ref="ds:Signature"/>
        <!-- Signature for VariablePart -->
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

図 9.2: Header の構造

署名が付されるので，コンテンツプロバイダが承認した CHS による署名であることを検証できるとともに，署名検証鍵の不正な挿入などの改ざんを検知することができる．

**リプレイ攻撃の防止** Header のダイジェスト値を VariablePart 中の HeaderDigest 要素（図 9.3）に指定する．異なる ConstantPart を含む Header は異なるダイジェスト値を持つので，Header のダイジェスト値と HeaderDigest とを比較することで，VariablePart が別のブリッジファイルからリプレイされていないことを検証できる．Nonce 要素にセッションを特定する乱数を指定するが，詳しくは 9.2.7 で説明する．

### 9.2.7 決済・著作権処理

ブリッジングプログラムと CHS との間のプロトコルを以下のように規定する（図 9.4）．ただし，Nonce, PreviousHeaderDigest, HeaderDigest は VariablePart 中の要素である（図 9.3）．

**Step 1** ブリッジングプログラムは Header とセッション毎に生成する乱数（Random）とを CHS に送付する．

**Step 2** CHS は Header からルールを取得して，決済・著作権処理を実行する．

**Step 3** CHS は以下の手順で VariablePart を更新し，ブリッジングプログラムに返送する．

- **3.a** Header のダイジェスト値を計算し，PreviousHeaderDigest に指定する．
- **3.b** Random を Nonce に指定する．
- **3.c** VariablePart 中のルールを更新する．
- **3.d** HeaderDigest を空バイトとした後に，更新された Header のダイジェスト値を計算し，結果を HeaderDigest に指定する．
- **3.e** 更新された VariablePart に署名する．

**Step 4** ブリッジングプログラムは，以下の手順で，VariablePart を検証する．

- **4.a** CHS に送付した Header のダイジェスト値と PreviousHeaderDigest との一致を検証する．

```
<xsd:element name="ConstantPart">
  <xsd:complexType content="element">
    <xsd:sequence>
      <xsd:element ref="PayloadDescriptors"/>
        <!-- Sequence of PayloadDesciptor elements -->
      <xsd:element ref="CHS-Keys"/>
        <!-- Sequence of ds:KeyInfo -->
      <xsd:element ref="BodyDigest"/>
        <!-- ds:DigestMethod & ds:DigestValue-->
      <xsd:any namespace="##other"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="VariablePart">
  <xsd:complexType content="element">
    <xsd:sequence>
      <xsd:element ref="Nonce"/>
      <xsd:element ref="HeaderDigest"/>
        <!-- ds:DigestMethod & ds:DigestValue -->
      <xsd:element ref="PreviousHeaderDigest"/>
        <!-- ds:DigestMethod & ds:DigestValue -->
      <xsd:any namespace="##other"/>
    </xsd:sequence>
    </xsd:attribute name="CHS" type="xsd:anyURI">
  </xsd:complexType>
</xsd:element>
```

図 9.3: ConstantPart と VariablePart の構造

**4.b** Random と Nonce の値との一致を検証する.

**4.c** CHS-Keys に指定される署名検証鍵を用いて **CHS** の署名を検証する.

上記のプロトコルが 9.2.3 で述べた（a），(b)，(c）の各条件を満足することを示す.

c. Random は現在のセッションのみにおいて使用されるので，Random と Nonce との一致，及び，**CHS** による署名とを検証することで，VariablePart が現在のセッションにおいて **CHS** によって生成され，かつ，改ざんされていないことを確認できる.

a. CHS-Keys は ConstantPart 中に指定され，コンテンツプロバイダにより署名されている.

b. PreviousHeaderDigest の値と **CHS** に送付した Header のダイジェスト値との一致の検証により，**CHS** に送付した Header に基づいて VariablePart が計算されていることを検証できる.

### 9.2.8 コンテンツのアクセス制御

ブリッジングプログラムの公開鍵で暗号化したコンテンツ鍵或いは **CA** 固有鍵を，PayloadDescriptor 要素（図 9.3）の xenc:EncryptedType（図 9.5）に指定する．ブリッジングプログラムは，9.2.7 のプロトコルに従って決済・著作権処理が完了したことを確認した後，プライベート鍵で xenc:EncryptedType を復号する.

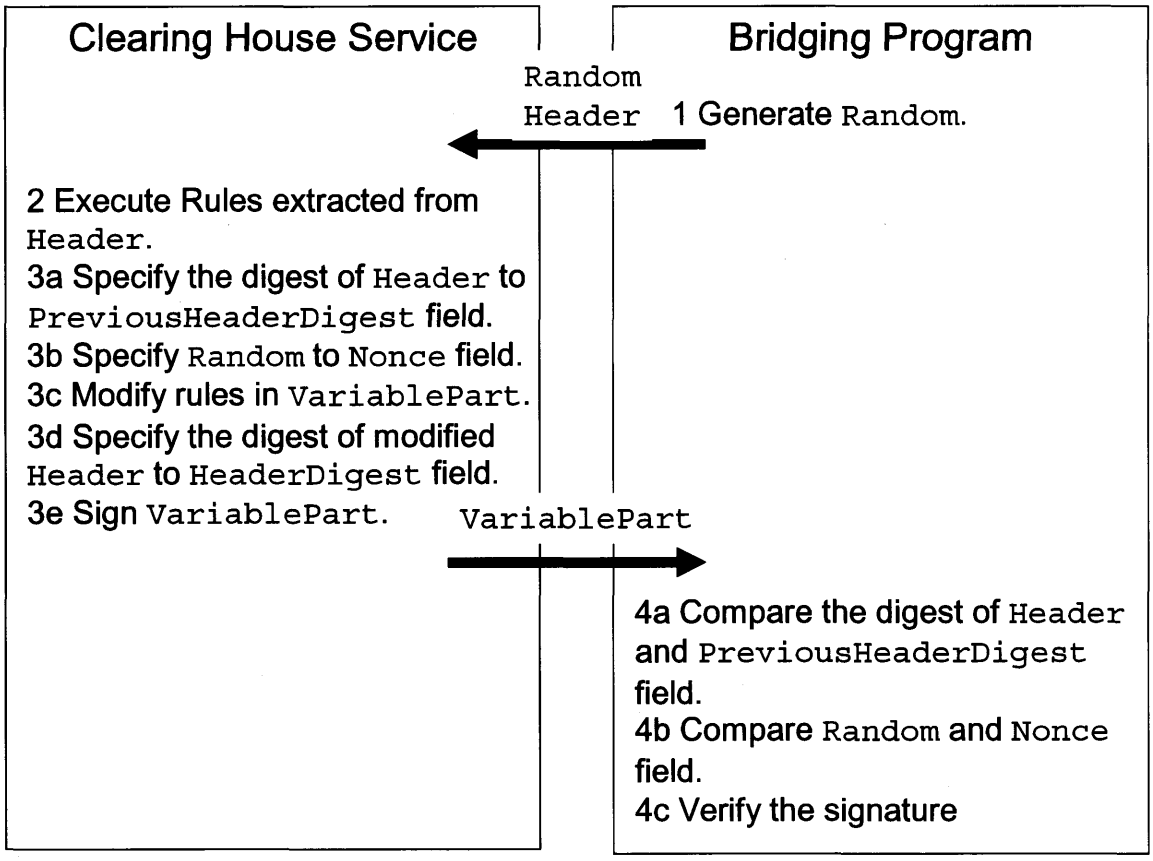



図 9.4: CHS とブリッジングプログラムのプロトコル

### 9.2.9 コンテンツのエクスポート

CA の仕様は，“外部モジュールから CA にコンテンツ或いは CA 固有鍵を引き渡すための I/F” の有（II）無（I），CA 固有鍵のオブジェクト化の可（A）否（B）の 2 軸で，合計 4 タイプに分類される．CA 固有鍵のオブジェクト化とは，CA 固有鍵を外部モジュールがオブジェクトとして認識できることを指す．

CA がタイプ I の場合，ブリッジングプログラムは，復号したコンテンツ或いは CA 固有鍵を，ファイル渡しやプロセス間通信など，OS に標準の機能を用いて CA に渡さざるを得ず，途中で漏洩する危険を免れない．

一方，CA がタイプ II の場合，ブリッジングプログラムは用意された I/F を介して，復号したコンテンツ或いは CA 固有鍵を送るので，安全性が増す．特に，CA の仕様により，I/F が保護されたチャネルや安全性が考慮された API を利用する場合には，コンテンツ及び CA 固有鍵がこの I/F 中で窃取される危険は更に減じる．例えば，Acrobat は，CA 固有鍵（RC4 の鍵）を Acrobat に入力するための API を用意しているが，この API の利用には Acrobat 社から提供されるライセンスキーが必要となる．

また，タイプ II で CA の I/F を利用する際，ブリッジングプログラムがすべての I/F をサポートする仕様では効率が悪く，また，CA によってはプラグインにのみ I/F を提供する仕様をとるため，ブリッジングプログラムと CA とを橋渡しをするモジュールを別に用意する必要がある．このモジュールを**ブリッジ**

```
<xsd:element name="PayloadDescriptor">
  <xsd:sequence>
    <xsd:element name="ContentType" type="xsd:anyURI">
    <xsd:element name="KeyProtection" content="empty"
      minOccurs="0" maxOccurs="1" />
    <xsd:element ref="xenc:EncryptionMethod"
      minOccurs="0" maxOccurs="1" />
    <xsd:element ref="xenc:EncryptedType"/>
  </xsd:sequence>
</xsd:element>
```

図 9.5: PayloadDescriptor の構造

表 9.1: 推奨される CA の仕様とエクスポートモード

| Type of CA | I–A | I–B | II–A | II–B |
|---|---|---|---|---|
| Appropriate Export Mode | (i) | (i)/(ii) | (iii) | (iv) |

**レイヤ対応拡張**と呼ぶ. ブリッジレイヤ対応拡張の実装は，CA へのプラグイン，ブリッジングプログラムへのプラグイン，或いは，独立プログラムなど，複数の形態が想定できる. ブリッジレイヤ対応拡張は，ブリッジングプログラムとの間では，ブリッジレイヤで定めるプロトコルに従い認証・機密性・完全性をサポートする保護されたチャネルを介してコンテンツ或いは CA 固有鍵を受け取り，CA との間では CA が用意する I/F を介してコンテンツ或いは CA 固有鍵を送付する.

CA が "CA 固有鍵のオブジェクト化" をサポートしている場合，ブリッジングプログラムはコンテンツの復号を行わず，CA 固有鍵の復号のみを実行し，冗長な暗号化を避け処理の効率化を図ることができる.

以上の考察に基づいて，本稿におけるブリッジングプログラムは，以下の 4 つのエキスポートモード（図 9.6）をサポートし，CA のタイプに応じて表 9.1 に従ってモードを選択するものとする.

i. コンテンツを復号し，CA にコンテンツのみをエクスポートする.

ii. CA 固有鍵を復号し，CA に CA 固有鍵をエクスポートする.

iii. コンテンツ鍵を復号し，CA のブリッジレイヤ対応拡張にコンテンツ鍵をエクスポートする. コンテンツの復号はブリッジレイヤ対応拡張が実行する.

iv. CA 固有鍵を復号し，CA のブリッジングレイヤ対応拡張に CA 固有鍵をエクスポートする. コンテンツの復号は CA が実行する.

ブリッジングプログラムは，PayloadDescriptor に指定される以下のパラメータ（図 9.5）を参照して，エクスポートを実行する.

ContentType エクスポート先の CA を特定する情報が指定される.

KeyProtection と xenc:EncryptionMethod ブリッジングプログラムは，表 9.2 に従ってエクスポートモードを選択する. ただし, (iii) と (iv) の区別はブリッジレイヤ対応拡張が行う. xenc:EncryptionMethod には，コンテンツの復号に必要な共通鍵暗号アルゴリズムなどのパラメータが指定される.

表 9.2: エクスポートモードの選択

| EncryptionMethod | KeyProtection | |
|---|---|---|
| | Left out | Specified |
| Left out | (ii) | (iii)/(iv) |
| Specified | (i) | N/A |

表 9.3: 実証対象アプリケーション

| Application | Vender | Export Mode |
|---|---|---|
| Acrobat | Adobe | (iv) |
| DocuWorks | Fuji Xerox | (iii) |
| Win32 Executable | N/A | (iii) |
| Real Player | Real Networks | (iii) |
| Windows Media Player | Microsoft | (iii) |
| MS Office | Microsoft | (ii) |

## 9.3 実現性の考察

以下では，ブリッジレイヤの要件の実現性，実行速度のオーバーヘッド，既存コンテンツの取扱いについて考察する．

9.2.1 に示した要件のうち，既存の拡張性のみを前提としてブリッジレイヤ準拠を保証する要件が，実現が最も困難で，実証が必要な項目である．

拡張性に関して CHS と CA とを比較すると，CHS は汎用性を維持するために，CA との連携を前提とする拡張性を有する．対照的に，CA は一般に CHS との連携を考慮した設計になっておらず，拡張性が提供される場合でも，仕様や実装に由来する制限が存在する．つまり，CHS のブリッジレイヤ準拠には実現上の困難はないと考えてよく，CA 側の準拠，すなわち，ブリッジングプログラムから CA へのコンテンツのエクスポートについては，実証が必要である．筆者らは，表 9.3 に示した CA アプリケーションに対して，9.2.9 のエクスポートモードが正しく機能するかを検査した．

Acrobat，DocuWorks，RealPlayer，Windows Media Player に対しては，ブリッジレイヤ対応拡張としてプラグインを開発した．実行形式ファイルである Win32 Executable では，任意のオブジェクトコードを暗号化した上でブリッジレイヤ対応拡張コードを添付するツールを開発した．ブリッジレイヤ対応拡張コードは，ブリッジングプログラムと通信してコンテンツ鍵を受け取った後，暗号化されたオブジェクトコードを復号して制御を移す．MS Office では，API が非公開なため，(ii) のモードを検証した．(i) の実現性は (ii) に準じるので，本稿では検証を割愛した．

検証の結果，すべてのアプリケーションにおいて，問題なくエクスポートが実行されることを確認した．なお，Acrobat，DocuWorks，RealPlayer に対しては，富士ゼロックス株式会社が本稿提案のエクスポート方式を商品の一部に採用した実績がある．

ブリッジレイヤ準拠による実行速度のオーバーヘッドの最大のファクタは，コンテンツ或いはコンテンツ鍵/CA 固有鍵の復号処理にある．しかし，標準的な暗号技術を利用する限りでは，オーバーヘッドは

表 9.4: 復号処理性能

| Algorithm | Pentium M 900MHz<br>Memory 512MByte<br>Windows XP Pro. | Pentium 4 3.4GHz<br>Memory 1.0GByte<br>Windows XP Pro. |
|---|---|---|
| DES 56bit | 15.0 MByte/s | 56.0 MByte/s |
| RSA 1024bit | 8.7 ms/operation | 4.1 ms/operation |

許容範囲にとどまる.

表 9.4 に、本稿で測定した復号処理性能を示す. これによると Pentium M 900MHz の場合でも，コンテンツ鍵と 10 MByte のコンテンツの復号に要する時間は 0.7 秒弱である. 実際に，表 9.3 のアプリケーションで検証した結果でも，ブリッジングプログラムとの連携の有り無しによる実行速度の差は，体感できなかった.

最後に既存のコンテンツの取扱について述べる. ブリッジファイルは，CA 固有の符号化方式やセマンティクスとは独立である. すなわち，9.2.6 で述べたファイル定義の知識があれば，既存のコンテンツを任意にブリッジファイルに変換することができるので，既に流通しているコンテンツをブリッジレイヤに準拠させることに障害はない. ただし，暗号アルゴリズムに関しては，ブリッジングプログラムの実装に依存する制約がある.

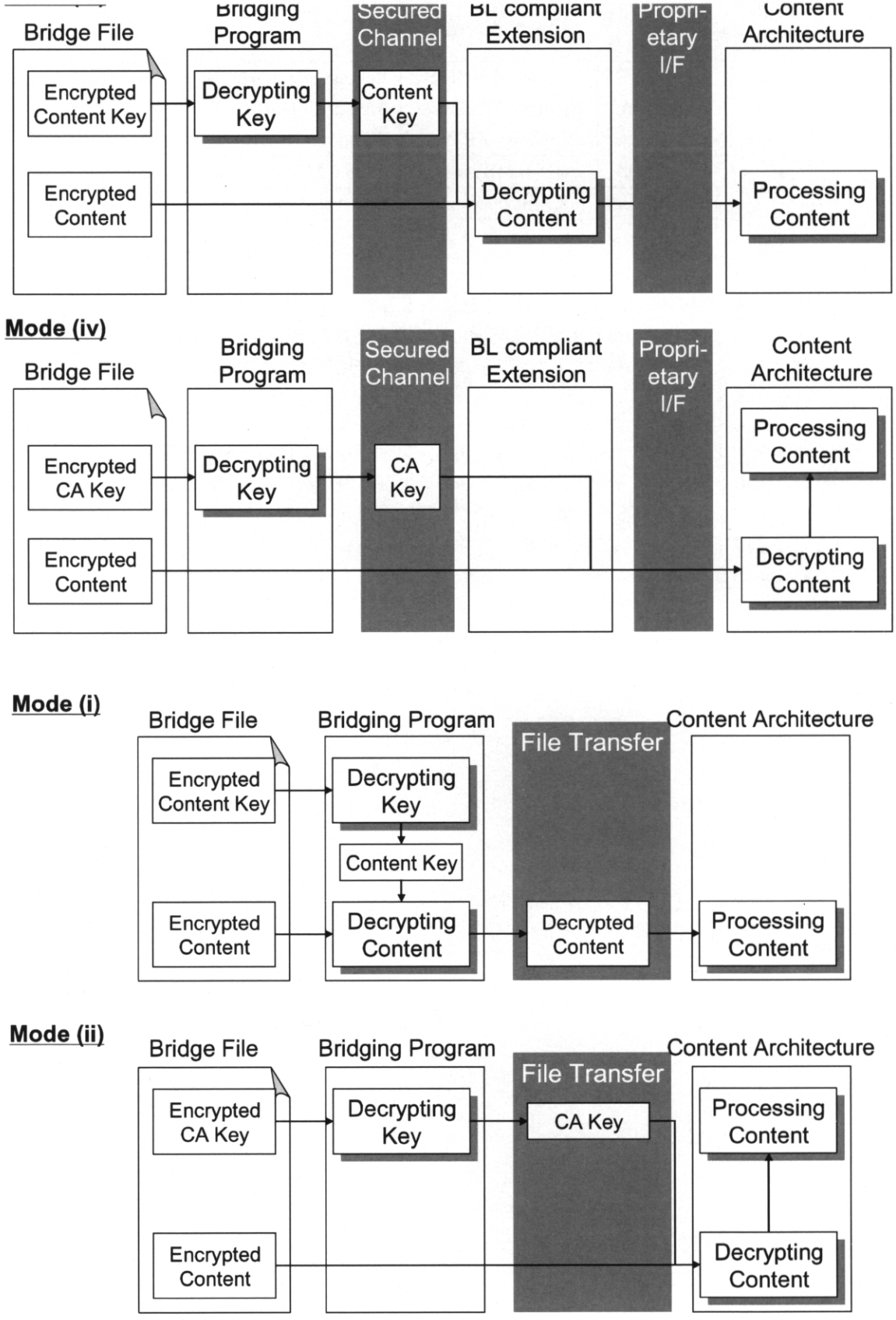



図 9.6: エクスポートモード

# 第10章　結論と今後の課題

本論文では、ユビキタス情報社会におけるアクセス制御について研究し、以下に述べる成果を得た。

- ユビキタスアクセス制御におけるプライバシーと公知との境界に関して、Consensual Disclosure の考え方を提案した。

  サービスの提供がシームレスかつ透過的に行われるユビキタスコンピューティングの環境では、ベースラインにおいてはユーザのアクセスには追跡不能性が保証されることが必要である。一方、サービス側の安全上の理由、或いは、社会的な理由により、アクセスの追跡が要請される場合に意は、ユーザが明示的な同意を与えた場合に限り、ユーザが同意する範囲に限って、追跡情報が開示されるべきである。

  本論文では、この考え方を Consensual Disclosure と名づけ、ユビキタスコンピューティングにおけるプライバシーと公知の境界の問題に対する解として、提案する。

- 追跡不能と Consensual Disclosure の実現として、実用的な計算量を有する具体的な方式を提案した。

  追跡不能性を有する認証方式としては、暗号技術であるグループ署名を使った方法が知られている。しかしながら、グループ署名において行われる暗号計算は、多大な計算量を必要とするため、高い頻度でアクセスイベントが発生するユビキタスコンピューティングに適用するには、欠点がある。

  本論文では、通常の高速な署名方式における鍵管理方式を工夫することにより、追跡不能性と Consensual Disclosure を満足し、かつ、高速な認証方式の考案に成功した。認証処理の計算量を、楕円曲線上のスカラー倍演算の実行回数に間然して比較すると、本方式では、追跡不能に対して 7 回、Consensual Disclosure に対して 21 回であるに対して、現時点で知られている最も効率的なグループ署名方式では 49 回が必要であることから、本方式が認証速度の観点で従来のグループ署名を大きく改善していることがわかる。

- ユビキタスコンピューティングでは、認証とアクセス制御を統合したプロセスとして取り扱うことが求められている。また、ユビキタスコンピューティング特有の事情として、アクセス権限の発行者と検証者との間で利害が背反する可能性が知られている。

  これらの要請や事情は、認証に対する基本的な要件に、付加的な要件を追加することとなる。しかしながら、ユビキタスアクセス制御に対する要件を、網羅的に整理しようとする研究事例は今までに報告されていなかった。

  本研究では、ユビキタスアクセス制御の要件について、可能な限り網羅的に考察し、以下の要件に整理した。

  1. Completeness
  2. Soundness
  3. Non-transferability
  4. Unlinkability
  5. Consensual disclosure
  6. Verifier authentication
  7. Revocation
  8. Access rule enforcement

9. Access rule update
10. Delegation
11. Message specification
12. Lower-layer connectivity

- ユビキタスアクセス制御に求められる要件の整理に基づいて、これらの要件をサポートしつつ、実用的な認証速度を有するプロトコルを考案し、提案した。

  本論文で提案するプロトコルは、先にグループ署名との比較において、追跡不能性と Consensual Disclosure とを満足する効率的な署名方式をベースとして、上記の要件を満足するように拡張したものである。但し、単に機能を追加するのではなく、計算量の多い暗号処理を共有できるよう、署名方式の特性を利用したことにより、効率性との両立を実現している。

  本論文の提案と前記要件との対応を表 10.1 に示す。

  また、グループ署名の各方式と本論文で提案するプロトコルとの計算量の比較を、表 10.2 に示す。

表 10.1: 要件のサポート

| Requirements | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rights-Issue | √ | √ | | √ | | | | | | | | |
| Unlink-Verify | √ | √ | | √ | | | | √ | | | | |
| Link-Verify | √ | √ | | | √ | | | √ | | | | |
| Unlink-Authenticate-SA | √ | √ | | √ | | √ | | √ | | | | |
| Link-Authenticate-SA | √ | √ | | | √ | √ | | √ | | | | |
| Unlink-Key-Transfer | √ | √ | | √ | | √ | | √ | | | | |
| Link-Key-Transfer | √ | √ | | | √ | √ | | √ | | | | |
| Unlink-Rights-Update | √ | √ | | √ | | √ | √ | √ | √ | | | |
| Link-Rights-Update | √ | √ | | | √ | √ | √ | √ | √ | | | |
| Unlink-Revoke | √ | √ | | √ | | √ | √ | √ | | | | |
| Link-Revoke | √ | √ | | | √ | √ | √ | √ | | | | |
| Unlink-Delegate | √ | √ | | √ | | √ | | √ | | √ | | |
| Link-Delegate | √ | √ | | | √ | √ | | √ | | √ | | |
| UACML | | | | | | | | | | | √ | √ |
| Tamper-Resistant H/W | | | √ | | | | | | | | | |

- 本論文では、更に、考案したプロトコルに対して、相互運用のためのメッセージ規定を提案する。UACML (Ubiquitous Access Control Message Layer) と呼ばれるこのメッセージ規定は、追跡不能性を有する既存の下位通信レイヤとの接続性と、Distributed Trust Management ととの整合を意識しながら、アプリケーションレイヤにおけるドメインを横断したアクセス制御を実現するものである。

- 最後に、ユビキタスコンピューティングにおける、重要なサービスの例としてデジタルコンテンツ流通を取り上げた。本論文で提案するアクセス制御方式をデジタルコンテンツ流通に適用することにより、明らかに、メリットが得られるが、そのためには、現在のデジタルコンテンツ流通に

表 10.2: 計算量の比較

| アルゴリズム/プロトコル | 署名 | 検証 | 合計 |
|---|---|---|---|
| Ateniese-Camenisch-Joyce-Tsudik [5] | 2700 | 2475 | 5175 |
| Boneh-Boyen-Shacham [12] | 20 | 33 | 53 |
| Camenisch-Lysyanskaya [15] | 42 | 69 | 111 |
| Camenisch-Groth [14] | 28 | 38 | 66 |
| Teranishi [60] | 65 | 86 | 151 |
| Nguyen-Safavi-Naini [51] | 38 | 37 | 75 |
| Furukawa-Imai [36] | 19 | 39 | 49 |
| Unlink-Verify | 5 | 2 | 7 |
| Link-Verify | 17 | 4 | 21 |
| Unlink-Authenticate-SA | 6 | 3 | 9 |
| Link-Authenticate-SA | 18 | 5 | 23 |
| Unlink-Key-Transfer | 14 | 4 | 18 |
| Link-Key-Transfer | 21 | 6 | 27 |
| Unlink-Rights-Update | 10 | 5 | 15 |
| Link-Rights-Update | 22 | 7 | 29 |
| Unlink-Revoke | 5 | 2 | 7 |
| Link-Revoke | 17 | 4 | 21 |
| Unlink-Delegate | 21 | 5 | 26 |
| Link-Delegate | 27 | 5 | 32 |

(楕円曲線上のスカラー倍演算の実行回数)

おいて問題となっている「コンテンツ保護」と「決済・著作権処理」の両機能間のでの相互接続の問題が解決されることが前提である。

本論文では、現状のアーキテクチャをベースとして、コンテンツ保護機能と決済著作権処理機能とを橋渡しするレイヤとして、ブリッジレイヤを提案し、その具体的な実現方式を示した。

本論文の以上の成果は、これから焦点が当たるであろうユビキタスアクセス制御の基盤技術として、機能性と実現性の観点から重要であると確信している。また、UACML を提案したことにより、第三者による実装も容易であり、設計としては一端の完成を見たものと認識している。

しかしながら、実証の観点から述べると、まだ確認し、必要によっては、改善が必要となる部分もあることは確かであり、特に、下記の 2 項目については、今後の取り組みが必要である。

**認証速度の実証** 本論文で提案する方式は、従来の方式で有望とされるグループ署名に対して、2〜7 倍の高速性を有する。また、試験的に作成したプログラムにより、80〜210 ミリ秒 (Pentium 4 3.2 GHz) で一回の認証処理を実行できることが確かめられている。

しかしながら、現実のユビキタス環境において、機器、通信方式、環境を変えて、この速度が十分であるかの検証は必要である。

**ユビキタス環境における他の技術との整合性** ユビキタスコンピューティングに関連して、多方面で、技術的な研究が進められている。例えば、ドメインを横断したローミング技術や、新しいドメインでサービスを探索する技術などの研究が盛んであるが、これらの技術と、本論文で提案するアクセス制御技術との整合性は、実証を通じて確認していく必要がある。

現在、本論文で提案した技術及びUACMLをベースにプロトタイプの実装を進めている。今後は、このプロトタイプをベースとして、具体的な実証を進めていく予定である。

# 謝辞

本論文の筆をおくにあたり、ご指導・ご協力を頂いた多くの方々に謝意を表したいと思います。

本研究は、東京大学先端科学技術研究センター 安田 浩教授の多面にわたるご指導無しにはなしえませんでした。本研究の専門的な内容の多くに関して非常に有意義な議論と重大な貢献を負っているばかりではありません。そもそも、東京大学先端科学技術研究センターにて私が本研究に携わることができ、また、特任助手として研究に専念できる場を得ることできましたのは、偏に安田教授が機会を与えて下さったことによります。感謝の念は筆舌に尽くせませんが、この場をかりて深謝いたします。

東京大学先端科学技術研究センター 青木 輝勝講師には、本研究の内容、及び、論文の執筆に関しまして、多大なご助言とご教示を賜りました。特に、本論文の要旨を整理するに当たって頂いた洞察に富んだ示唆は、本論文執筆に当たって大きな助けとなりました。

本研究は、富士ゼロックス株式会社総合研究所勤務時代に私が従事していた研究プロジェクトに源流を求めることができます。青沼 英一氏、京嶋 仁樹氏、齋藤 和雄氏を始めとする当時のプロジェクトメンバーの方々とは、有意義な共同研究を行い、また、その成果の一部は本研究の基礎となっています。

本研究の応用であるデジタルコンテンツ流通のテーマに関しましては、富士ゼロックス株式会社 竹田 幸史氏、並びに、田口 正弘氏とは、互いに刺激を受けながら、共同研究者として共に研究をしてまいりました。また、共同研究の結果は、本論文の一部となりました。

株式会社ソルコム 和田 康氏、平岡 真樹氏、鎌谷 和也氏、株式会社ディー・ディー・エス 山村 雅典氏、保黒 政大氏、前田 勝之氏とは、UACML のメッセージ規定を作成する際に、事業と実装の観点から多くの有益な助言を頂きました。

本研究は、他にも多くの方々の助けを得て、現在の形となっています。特に、安田・青木研究室の秘書の方々には、研究推進に伴って発生する多くの事務処理を快くお引き受けいただき、本研究をバックアップして頂きました。

皆様に、深く感謝いたします。

最後に、本研究は、経済産業省商務政策局「平成 17 年度新世代情報セキュリティ研究開発事業（平成 17・11・25 財情第 2 号）」及び「平成 18 年度新世代情報セキュリティ研究開発事業」の研究として行われたものであることを附記いたします。

# 研究業績

## 筆頭著者査読付論文誌

1. 申吉浩, 青沼英一　ディジタルコンテンツの権利保護と流通, 情報処理, Vol.40, No.5（ベストオーサ賞受賞）, 1999
2. Kilho Shin, *Access Control for E-Business on MOPASS*, IPSI Transactions on Internet Research, Winter Issue, 2004
3. 申吉浩, 竹田幸史, 田口正弘, 青木輝勝, 安田浩　デジタルコンテンツ流通のためのブリッジレイヤ、電子情報通信学会, Vol.J88-D1, No.11, 2005

## 掲載予定筆頭著者査読付論文誌

1. Kilho Shin, Koji Takeda, Masahiro Taguchi, Terumasa Aoki and Hiroshi Yasuda, *Bridge Layer for Content Distribution*, Systems and Computer in Japan, Wiley, 掲載予定,「デジタルコンテンツ流通のためのブリッジレイヤ」の英訳
2. Kilho Shin and Hiroshi Yasuda, *Provably Secure Access Control Scheme for Ubiquitous Computing*, Journal of Computer, Academy Publisher, 条件付採録

## 筆頭著者全文査読付国際会議

1. Kilho Shin and Masaki Kyojima, *Secure and Efficient Schemes to Entrust the Use of Private keys*, Eighth IEEE International Workshops on Enabling Technologies (WET ICE 2000)
2. Kilho Shin, *Digital Qualification - an approach to infrastructures of access control for Internet commerce*, SSGRR 2001
3. Kilho Shin and Masahiro Taguchi, *Access Control for E-Business on MOPASS*, Applied Computing 2004 (AP 2004)
4. Kilho Shin, *Security of an access control protocol for MOPASS smart card*, International Conference on Internet Technology and Applications 2005 (ITA 2005)
5. Kilho Shin and Hiroshi Yasuda, *Provably secure anonymous access control for heterogeneous trusts*, The First International Conference on Availability, Reliability and Security (ARES 2006)

# 共著査読付論文誌他

## 共著査読付論文誌

1. 上林憲行，小澤英昭，平山智史，申吉浩　ディジタルコンテンツ流通の技術と制度：アトムからビットへ, 情報処理学会誌 40 周年記念特集号：【情報処理技術 －過去十年そして今後の十年－ 】, Vol.41, No.5, 2000 年 5 月

## 研究会

1. 申吉浩, 小島俊一　デジタル著作物流通の為のアクセス制御スキーム, 情報通信学会技報 ISEC97-20

## 競争資金獲得

1. 経済産業省商務政策局　平成１７年度新世代情報セキュリティ研究開発事業平成１７・１１・２５財情第２号
2. 経済産業省商務政策局　平成１８年度新世代情報セキュリティ研究開発事業

# その他の業績

## 論文誌

1. Tetsuji Kuboyama, Kilho Shin and Tetsuhiro MiyaharaA, *Theoretical Analysis of Tree Edit Distance Measures*, Transactions on Mathematical Modeling and its Applications (TOM), The Information Processing Society of Japan, Vol. 13, 2005
2. 石村享久, 申吉浩, 上林憲行　地域ウェブ空間の特徴抽出, 情報処理学会論文誌, 第 47 巻, 第 3 号, 2006

## 全文査読付国際会議

1. Yukihisa Ishimura, Kilho Shin and Noriyuki Kamibayashi, *Graph-Theoretic Nature of Local Web-Spaces*, DEXA Workshops 2005
2. Tetsuji Kuboyama, Kilho Shin, Tetsuhiro Miyahara and Hiroshi Yasuda, *A Theoretical Analysis of Alignment and Edit Problems for Trees*, Proc. of Theoretical Computer Science, The 9th Italian Conference, Lecture Notes in Computer Science 3701, 2005
3. Tetsuji Kuboyama, Kilho Shin and Hisashi Kashima, *Flexible Tree Kernels based on Counting the Number of Tree Mappings*, Mining and Learning with Graphs (MLG 2006)

## 標準化活動

1. Kilho Shin (Project Editor), ISO/IEC 8613-11 Tabular structures and tabular layout

## 研究会

1. A Hierarchy of Approximate Tree Matching, 情報処理学会九州支部火の国情報シンポジウム, 情報処理学会九州支部, 2005
2. A Hierarchy of Tree Edit Distance Measures, 数理解析研究所講究録計算機科学基礎理論とその応用, 京都大学数理解析研究所, Vol 1426, 2005
3. 木の編集距離尺度の理論的解析, 情報処理学会研究報告書 第 53 回 数理モデル化と問題解決 (MPS) 研究会, 2005
4. Merging Shock Trees Using Alignable Mappings, 電子情報通信学会 技術研究報告, Vol. 104, 2005
5. Measuring Distance and Finding Approximate Common Patterns in Trees - Focus on Edit Distance, 人工知能学会 人工知能基礎問題研究会 (第 57 回), 2004
6. Class Identification for Tree Mappings, 情報処理学会 アルゴリズム研究会研究報告, 2005
7. 半構造データ統合のための木構造の近似照合と結合手法, 情報処理学会情報基礎学研究報告, 2005
8. 木構造間ノード写像の近似照合クラス同定アルゴリズム, 人工知能学会人工知能基礎問題研究会 (第 60 回), 2005

# 関連図書


[1] 4C Entity, LLC. *Content Protection for Prerecorded Media Specification, DVD Book*, 2001.

[2] 4C Entity, LLC. *Content Protection for Recordable Media Speification, SD Memory Card Book*, 2002.

[3] 4C Entity, LLC. *Content Protection for Recordable Media Specification, DVD Book*, 2003.

[4] Adobe System Incorporated. *http://www.adobe.com/ products/acrobat/webbuy/main.html.*

[5] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. *Lecture Notes in Computer Science*, 1880:255 – 270, 2000.

[6] L. Barkhuus and A. Dey. Location-based services for mobile telephony: a study of users' privacy concerns. In *INTERACT 2003, 9th IFIP TC13 International Conference on Human-Computer Interface*, 2003.

[7] M. Bellare, R. Canetti, and H. Krawczyk. Keyed hash functions for message authentication,. In *Advances in Cryptology – CRYPTO 96*, pages 1 — 15, 1996.

[8] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Crypto '93, (LNCS) 773*, pages 232–249. Springer-Verleg, 1994.

[9] B. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The role of trust management ini distributed sistems. *Secure Internet Programming, LNCS*, 1603:185–210, 1999.

[10] B. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. *IEEE Proceedings of the 17th Symposium*, 1996.

[11] Gerrit Bleumer. Biometric yet privacy protecting person authentication. *Lecture Notes in Computer Science*, 1525:99–110, 1998.

[12] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Proceedings of Crypto '04*, 2004.

[13] S. Brands. Untraceable off-line cash in wallet with observers. In *Advances in Cryptology — CRYPTO '93, LSCN 773*, pages 302 – 318. Springer-Verleg, 1994.

[14] J. Camenisch and J. Groth. Group signatures: better efficiency and new theoretical aspects, 2004.

[15] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Proceedings of Crypto '04*, 2004.

[16] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 21–30, New York, NY, USA, 2002. ACM Press.

[17] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 93+, 2001.

[18] Jan L. Camenisch. Efficient and generalized group signatures. *Lecture Notes in Computer Science*, 1233:465–479, 1997.

[19] Jan L. Camenisch and Markus A. Stadler. Efficient group signature schemes for large groups. *Lecture Notes in Computer Science*, 1294:410+, 1997.

[20] R. Canetti and H. Krawszyk. Security analysis of ike's signature-based key-exchange protocol. In *Crypto '02, LNCS 2442*, pages 143–161. Springer-Verleg, 2002.

[21] D. Chaum. Showing credentials without identification: Transrerring signatures between unconditionally unlinkable pseudonyms. In *Proceedings of AUSCRYPT '90, LSCN 453*, pages 246 – 264. Springer-Verleg, 1990.

[22] D. Chaum. Achieving electronic privacy. *Scientific American*, 267(2):96 – 101, 1992.

[23] D. Chaum, B. den Boer, E. van Heyst, S. Mjlsnes, and A. Steenbeek. Efficient offline electronic checs. In *Advances in Cryptology — EUROCRYPT '89, LSCN 434*, pages 294 – 301. Springer-Verleg, 1990.

[24] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology — CRYPTO '88, LSCN 403*, pages 319 – 327. Springer-Verleg, 1990.

[25] D. Chaum and T. Pedersen. Wallet databases with observers. In *Advances in Cryptology – CRYPTO '92*, volume 740, pages 89–105, 1993.

[26] D. Chaum and E. van Heist. Group signatures. In *In Advances in Cryptology — EUROCRYPT '91, LNCS 547*, pages 257–265. Springer-Verlag, 1991.

[27] Lidong Chen and Torben P. Pedersen. New group signature schemes. In *In Advances in Cryptology — EUROCRYPT '94, LNCS 550*, pages 171–181. Springer-Verlag, 1995.

[28] S. Chokhanni and W. Ford. Internet x.509 public key infrastructure certificate policy and certification practices framework. RFC 2527, IETF, March 1999.

[29] S. Chokhanni, W. Ford, R. Sabett, C. Merrill, and S. Wu. Internet x.509 public key infrastructure certificate policy and certification practices framework. RFC 3647, IETF, November 2003.

[30] Content ID Forum. *cIDF Specification 2.0*, 2003.

[31] ContentGuard. *eXtensive rights Markup Language (XrML) 2.0 Specification*, 2001.

[32] R. Cramer and T. Pederson. Improved privacy in wallets with observers. In *Advances in Cryptology — EUROCRYPT '93, LSCN 765*, pages 329 – 343. Springer-Verleg, 1994.

[33] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B.M. Thomas, and T. Ylonen. SPKI certificate theory. RFC 2693, IETF, September 1999.

[34] A. Fiat and A. Shamir. How to prove yourself: Practical solution to identification and signature problems. In *Crypto '86, LNCS 263*, pages 186–194. Springer-Verlag, 1987.

[35] M. Franklin and M. Yung. Secure and efficient off-line digital money. In *20th International Colloquium on Automata, Languages and Computer Science, LSCN 700*, pages 265 – 276. Springer-Verleg, 1993.

[36] J. Furukawa and H. Imai. An efficient group signature scheme from bilinear maps. In *ACISP 2005*, pages 455–467, 2005.

[37] O. Goldreich. *Foundations of Cryptography, Volume II Basic Applications*. Cambridge University Press, 2004.

[38] G. Iachello and G.D. Abowd. A token-based access control mechanism for automated capture and access systems in ubiquitous computing. GIT Technical Report GIT-GVU-05-06, Georgia Institute of Technology, GVU Center, June 2005.

[39] Infrared Data Association. *Infrared Data Association: Serial Infrared Link Access Protocol (IrLAP) Version 1.1*, 1996.

[40] International Organization for Standardization. *ISO/IEC 14443-3: Identification cards – Contactless Integrated Circuit(s) cards – Proximity cards – Part 3:Initialization and anticollision*, 2000.

[41] International Organization for Standardization. *ISO/IEC 18092-3: Information technology – Telecomunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1)*, 2004.

[42] Pramod A Jamkhedkary and Gregory L Heilemanz. DRM as a layered system. In *Proc. of DRM 2004*, pages 11 – 21, 10 2004.

[43] Eija Kaasinen. User needs for location-aware mobile services. *Personal Ubiquitous Comput.*, 7(1):70–79, 2003.

[44] Lalana Kagal, Scott Cost, Tim Finin, and Yun Peng. A framework for distributed trust management. In *IJCAI-01 Workshop on Autonomy, Delegation and Control*, 2001.

[45] Lalana Kagal, Vladimir Korolev, Sasikanth Avancha, Anupam Joshi, Tim Finin, and Yelena Yesha. Centaurus: an infrastructure for service management in ubiquitous computing environments. *Wirel. Netw.*, 8(6):619–635, 2002.

[46] Iwata Laboratory. *Crypt Library AiCrypto*. Nagoya Institute of Technology.

[47] W. Mao. *Modern Cryptography — Theory and Practice*. Prentice Hall Professilnal Technical Reference, 2003.

[48] Jose M Martinez, Rob Koenen, and Fernando Pereira. MPEG-7: the generic multimedia content description standard. In *IEEE Multimedia*, volume 9 (2), pages 78 – 87, 2002.

[49] Melodies & Memories Global Limited. *http://www.m-m-g.net/*.

[50] A. Menezes, M. Qu, and S. Vanstone. Some new key agreement protocols providing implicit authentication. In *Preproceedings of Workshops on Selected Areas in Cryptography*, 1995.

[51] L. Nguyen and R. Safavi-Naini. Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings. In *Proceedings of AsiaCrypt '04*, pages 372 – 386, 2004.

[52] NIST. Digital Signature Standard (DSS). FIPS PUB 186, May 1994.

[53] Open Mobile Alliance. *DRM Rights Expression Language V2.0*, 2004.

[54] Open Mobile Alliance. *DRM Specification V2.0*, 2004.

[55] L. Palen and P. Dourish. Unpacking "privacy" for a networked world. In *CHI2003*, 2003.

[56] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120 – 126, 1978.

[57] C.P. Schnorr. Efficient identification and signatures for smart cards. In *Crypto '89, LNCS 435*, pages 239–252. Springer-Verlag, 1990.

[58] Secure Digital Music Initiative. *SDMI Portable Device Specification Part 1 Version 1.0*, 7 1999.

[59] Dawn Xiaodong Song. Practical forward secure group signature schemes. In *ACM Conference on Computer and Communications Security*, pages 225 – 234, 2001.

[60] I. Teranishi, J. Furukawa, and K. Sako. $k$-times anonymous authentication. In *Proceedings of AsiaCrypt '04*, pages 308 – 322, 2004.

[61] They Weave Themselves. The most profound technologies are those that disappear.

[62] H. C. A. van Tilborg. *Encyclopedia of Cryptography and Security*. Springer, 2005.

[63] M. Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–104, 1991.

[64] M. Weiser. The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communication Revuew*, 3:3–11, 1999.

[65] World Wide Web Consortium. *W3C Recommendation: XML Encryption*.

[66] World Wide Web Consortium. *W3C Recommendation: XML Schema*.

[67] World Wide Web Consortium. *W3C Recommendation: XML Signature*.

[68] 加藤 岳久, 岡田 光司, and 吉田 琢也. 匿名認証技術とその応用. 東芝レビュー 6, 東芝, 2005.

[69] 佐古 和恵 米沢 祥子 古川 潤. セキュリティとプライバシを両立させる匿名認証技術について. **情報処理**, 47(4):410 – 416, April 2006.