Javaを用いた動画配信カプセルの実装

阿部 剛仁，谷口 展郎，塩野入 理
NTTサイバースペース研究所

ネットワークを用いたディジタル情報の流通が盛んになり，情報の知的所有権を保護する仕組みが強く求められるようになった．我々はこれまでに，内包する情報に対して自らアクセス制御を行い，不正利用を防止する機能をもつ情報カプセルMatryoshkaを提案してきた．本稿では，クロスプラットフォーム環境で動作するJavaベースの動画カプセルを設計するとともに，その実装について報告する．

Implementation of Java-based Secure Movie Capsule

Takehito ABE, Noburou TANIGUCHI and Osamu SHIONOIRI
NTT Cyber Space Laboratories

The more digital information is distributed in computer network, the more strongly required a scheme to protect intellectual property rights of the content. We have proposed an information capsule mechanism called *Matryoshka*. The capsule has functions preventing illegal accesses to its own content. In this paper, we proposed the design for new Java based movie capsule that works on cross platform environment, and then introduce its implementation.

1. はじめに

パーソナルコンピュータや携帯端末の普及により，ネットワークを通じたディジタル情報の流通を，誰もが手軽に行える環境が整いつつある．また，MPEG 等の符号化技術と情報通信技術の進歩によって，扱える情報形態も文字情報から高品質の静止画像，音声，動画像へと広がり，ネットワークを用いた映画や楽曲の販売も試みられるようになった[1][2]．しかしながら，ディジタル情報は複製が容易で劣化しないという特徴をもつため，一定の価値を持つ情報(以下コンテンツと呼ぶ)を流通する際には，不正な利用を防止し，知的所有権を保護する仕組みが不可欠である．

我々はこれまで，コンテンツの生成，流通，利用，加工といったライフサイクルを一貫して管理する情報流通システムと，その情報格納単位である情報カプセルMatryoshkaを提案してきた[3]．Matryoshkaでは，カプセルが内包するコンテンツは，提供者が指定する条件に従ってのみ利用される等，自律的な管理が行われる．既報[4]では，動画コンテンツの配信を目的としたJavaベースの情報カプセルを提案し，そのプロトタイプについての紹介を行った．プロトタイプでは，Matryoshka の基本的な自律管理機能を確認することができたが，カプセルの構造と動画処理の方式から，拡張性や実行環境等に制約が残っていた．

本稿では，前述の動画カプセルを基本とし，クロスプラットフォーム環境で動作する拡張性の高い新たなJavaベースの動画カプセルの設計と，その実装について報告する．

2. コンテンツカプセル

2.1 自律情報カプセル Matryoshka

Matryoshkaとは，流通対象となる価値を持つ

情報［コンテンツ］と，利用制約条件やコンテンツの説明等の情報［コンテンツ関連情報］と，カプセル全体の動作を統括し，利用制約条件に基づきコンテンツを実際に表現する機能［コントロール］を有する単一の格納単位であり，通常はそれらを内包する実行ファイル形式で存在する．Matryoshkaカプセルを実行すると，カプセル内のコントロールは，同じくカプセルに内包されているコンテンツ関連情報を読み込むとともに，実行環境をセンシングし，利用制約条件のチェックを行う．条件が適合する場合，コントロールは履歴情報の更新処理を行った後に，制約条件に基づいてコンテンツを表示する．コンテンツ及びコンテンツ関連情報は通常暗号化処理がなされており，カプセルに内包されるコントロールのみが必要な暗号復号化処理等を行い，コンテンツの正常な表示ができるよう設計されている．このようにMatryoshkaカプセルは，コンテンツ自身に表現手段と利用制御機構を併せ持つことで，どの流通過程においても，自律的にコンテンツの保護を行うことが可能になっている．

2.2 Java クラスファイルによる動画カプセルの実装

動画Matryoshkaカプセルは，ネットワークから切り離された状態においても，内包する動画コンテンツの利用制御を自立的に行うことを目的としており，現在のところ蓄積型の動画メディアのみを対象とし，再生中に常時接続を必要とするストリームメディアは対象にしていない．既報のJava を用いた動画カプセルは，動画コンテンツとコンテンツ関連情報をファイル内に併せ持ちながら，見かけ上Javaのクラスファイルとして実装され，Java のアプリケーションとして動作するものである（以下この実装をクラスファイル実装動画カプセルと呼ぶ）．このカプセルは通常のJavaアプリケーションと同様の手順により起動される．カプセルは内包する利用制約条件（パスワード認証，利用期限，利用期間，利用回数）をチェックし，利用可能な条件であると判断すると，同じく内包する動画コンテンツの再生を行う．

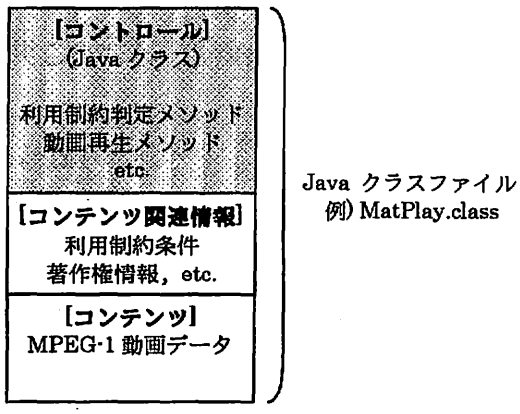

図1．Java によるクラスファイル実装動画カプセル

Java 言語においては，１クラス１ファイルという規約が存在するため，利用制約条件の解釈や動画を表示するためのコントロールは，すべてそのクラス内のメソッドとして実装されている（図１）．一方，動画コンテンツは暗号化状態でカプセルに内包され，カプセル利用の際には，カプセル内のコントロールによってリアルタイムに復号され画面に表示される．このクラスファイル実装動画カプセルでは，暗号の復号化処理をCODEC に送られる直前で行っているため，MPEG-1 System[5]のデータ構造に依存したブロック単位での暗号化/復号化がなされている．したがってカプセルに内包可能な動画フォーマットはMPEG-1に限られている．

動画の再生には，Sun Microsystemsの提供するJavaの拡張API，Java Media Framework (JMF)[6]を利用している．JMFには２つのタイプがあり，ひとつは全てのJava Compatible™クライアントで利用可能なクロスプラットフォーム実装(JMF-CP)で，もうひとつは特定のOSに依存のライブラリを用いるパフォーマンスパック(JMF-PP)である．MPEG-1フォーマットの動画を再生できるのはJMF-PPでのみであるため，このカプセルが動作可能なのは，現在JMF-PPが提供されているWindows，Solarisプラットフォームに限定される．

3. JAR 構造を持つ動画カプセルの実装

前章にて示したとおり，クラスファイル実装動画カプセルでは，カプセルの実行が特定の OS に限定されていた．また，カプセル内のすべてのコントロール機能を１クラスのメソッドとして実装する設計から，カプセルの保守・拡張性が損なわれるとともに，カプセルのファイル名が変更できない等の運用面での制約も生じていた．

このような問題点を解決するため，Java のもうひとつの実行形式である JAR(Java Archive) 形式の構造をもち，JMF-CP/JMF-PP といった実装環境や，動画コンテンツのフォーマットに依存しない独自の処理方式を採用した新たな動画 Matryoshka カプセルの開発と実装を行った(以下この実装を，JAR 実装動画カプセルと呼ぶ)．以下の項目にて，この JAR 実装動画カプセルの詳細を述べる．

3.1 基本構成

JAR 実装動画カプセルは，Java のアプリケーションとして JavaVM 上で動作し，実行時に他の外部アプリケーションを使用しない．内包する情報は，クラスファイル実装動画カプセルと同様に，コントロール，コンテンツ関連情報，動画コンテンツである．これらの情報を含むファイルを JAR 形式でアーカイブすることによって，カプセルを生成する(図２)．JAR 実装動画カプセルでは，複数のクラスを内包することが可能であり，コントロール部分は，最初に呼び出されるメインのクラスの他，利用条件判定，表示レイアウト設定，動画再生制御等を行う複数のクラスからなるクラス群で構成される．JAR のアーカイビング時に，Manifest 情報へ最初に呼び出すコントロールクラス名を記録することで，カプセル起動時に内包クラス名を考慮する必要はなくなる．したがってクラスファイル実装動画カプセルとは異なり，カプセル名(JAR ファイル名)の変更は任意に行うことが可能である．

コンテンツ関連情報及びコンテンツ自身には，

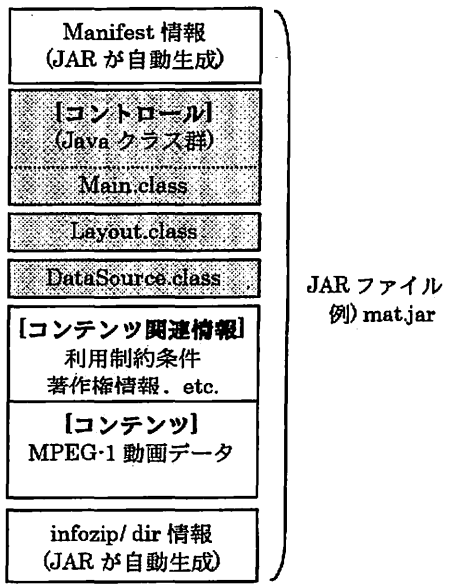

図２．JAR 実装動画カプセル

暗号化と結合化処理を行い，一連のデータ列としてカプセル内に格納する．このデータ列は，外部からは JAR 内の１ファイルとして認識されるが，その内部には一種のファイル内ファイルが存在し，個々の情報へはコントロールを介してアクセスする．コントロールは，コンテンツ管理情報及びコンテンツにアクセスする際に，データの暗号復号化処理を同時に行う．

3.2 利用動作制御

JAR 実装動画カプセルで利用できる利用制約は，クラスファイル実装動画カプセルと同じく以下の４つである．

(1) 利用者認証(パスワード)
(2) 利用期限指定
(3) 利用期間指定
(4) 利用回数指定

利用者認証は，カプセル起動時にパスワードを入力することで行われる．このパスワードは，コントロールがコンテンツ関連情報及びコンテンツにアクセスする際の復号情報として利用される．利用期間指定は，コンテンツの利用を許可する期間を設定するもので，本実装ではカプセルが最初

に利用された時刻からの期間を指定する．利用期限指定は，利用を許可する期日を絶対時間で指定するもので，その期日を過ぎた場合には，カプセルが起動しないよう設計されている．利用回数指定はカプセルの起動回数をカウントし，カプセルの利用回数を指定するものである．

これらの利用動作制御は，コンテンツ関連情報に記された利用制約条件と，カプセルの利用履歴情報を元に，コントロールの利用許可判定機能によって行われる．また，利用履歴情報は JAR アーカイブ内追加記録され，ユーザーが行うカプセルの起動・終了，及び動画再生・停止の操作を記録している．

3.3 動画再生処理

動画の再生には，JMF の Player インターフェイスを利用する．JMF では DataSource クラスがコンテンツメディアの場所とデータ制御のプロトコル情報を管理し，Player にデータを受け渡す．本実装では，独自の DataSource クラスを作成し，カプセル内の動画コンテンツを含む特殊な暗号化ファイルから，要求に応じて必要な部分のデータを読み取り，暗号復号化する処理を行わせている．このようなカスタム DataSource をカプセル内に配置することで，Matryoshka カプセル固有のデータ抽出処理や復号化処理は，全て JMF のメディアハンドリングレイヤーで行うことができ，JMF が用意する各種動画ファーマットのパーサー及び CODEC を，全てそのまま利用することが可能になっている．これにより，JMF-CP が対応するフォーマット[7]の動画コンテンツを内包するカプセルは，Java の実行環境が有るどの環境においても実行可能である．

3.4 カプセル生成ツール

JAR 実装動画カプセルは，コントロールを行う各種 Java クラスファイル，動画コンテンツファイル，利用制約条件等のコンテンツ関連情報を，JAR アーカイビングすることによって生成される．その際，カプセルの作成者はコンテンツ等の暗号化に用いるパスワードの設定を行う．我々は必要な情報を与えることで，カプセルを自動生成するツールを作成した．GUI もしくはコマンドライン入力により，カプセルに様々な設定を行うことができる．このカプセル生成ツールも Java によって作成されているため，様々なプラットフォームで動作可能である．

3.5 カプセルの動作検証

カプセルの生成/動作検証には，以下の実行環境と動画コンテンツを用いた．

端末 PC:Pentium II 300MHz, 320MB RAM.
OS: Windows 98SE
Java: J2SE1.2.2，JMF2.1 Windows-PP

動画:QuickTime Movie
Video: H263, 176x132 pixels, 15 fps
Audio: ULAW 800kHz, 8bit, mono
Size: 268KB, Duration: 10.26 sec

本実装において，暗号化は J2SE にて標準で提供されている擬似乱数アルゴリズムを用いている．前項で示したツールに対して，利用条件等の必要なパラメータを入力することにより，JAR 形式でアーカイブされた単一のファイルが正常に作成されるのを確認した．この JAR ファイルは，コマンドラインから java コマンドを用いるか，アイコンのダブルクリックによって起動され，カプセル内に記述されている利用制約条件が満たされる場合のみ，内包する動画コンテンツの再生が行われる．実際の再生品質は，カプセル化前の動画ファイルの再生と同程度であった．このカプセルを Linux テスト環境(同一 PC，Vine Linux 2.0，J2SE1.2.2 without JIT，JMF2.1-CP)で実行し，利用制限等を含め正常に動作するのを確認した．しかし，すべて Java で実装されている JMF-CP は，各プラットフォームに最適化された JMF-PP に比べると，動作パフォーマンスの低下がみられた．

また，以下の MPEG-1 フォーマットの動画デ

ータについて，同様の条件でカプセルを作成し，Windows+JMF-PP 環境で動作の検証をした．

MPEG-1 (System) Movie
Video:MPEG-1Video, 176x112, 30 fps
Audio:MPEG-1Audio layer2

こちらも見た目の再生品質は，元の動画コンテンツと変わらず，その他の機能も正常に動作した．MPEG の再生は JMF-PP のみ再生が可能である．この MPEG-1 カプセルは，JMF-PP が提供されている Solaris の環境においても正常に動作することを確認した．

4. 考察

4.1 前実装方式との比較

従来のクラスファイル実装動画カプセルにおいては，以下の 4 つの問題点が明らかになっていた．

(1)Java バイトコードの脆弱性
　逆コンパイルによるリバースエンジニアリング．
(2)ハードウエア情報が取得できない
　端末限定の利用制約が不可能．
(3)クラスファイルの利用による制限
　実装の柔軟性低下．ファイル名の選択不可能．
(4)JMF による制限
　MPEG-1 のみ対応．特定プラットフォームに限定．

(3)と(4)については，カプセルの JAR ファイル化と，カスタム DataSource を用いた動画データ処理方式の採用によって，本実装方式においてほぼ解決することができた．カスタム DataSource を用いる方式では，JMF の Player からは通常の動画ファイルの再生と同様の手順で，透過的にカプセル内の動画データにアクセスすることが可能である．従ってカプセル内コントロールの設計が容易になるとともに，JMF の対象動画フォーマットの拡張等に柔軟に対処できるという利点もある．また，現状のカプセルにおいて利用制御可能なコンテンツは，動画コンテンツ 1 つのみであるが，複数のコンテンツへ対応する際は，JAR ファイルへのコンテンツリソース(動画コンテンツ+コンテンツ関連情報)の追加と，プレーヤーの一部機能の改造のみで行える利点もある．一方，カプセルを JAR ファイルとしたことで，カプセルに内包されるコントロールやコンテンツリソースの分離が，より容易に行えるという弊害もある．コンテンツリソースは暗号化されているため，分離されても直接利用・改竄することはできないが，コントロールや利用制約情報の不正な改竄をより強力に防止するためには，ハッシュ関数を用いた改竄チェックを行うなどの対策が必要であると考えている．

(1)と(2)については，Java の仕様に深くかかわる部分であり，いまだ問題を解決するに至っていない．Java クラスの逆コンパイルを容易に行えないようするために，カプセル内のクラスファイルに対しても暗号化を施し，独自のクラスローダを設計するなど対応が考えられる．また，端末限定に近い効果を得るために，IC カードを利用した認証を行う等，ハードウエアの併用を含め解決の道を探る予定である．

4.2 クロスプラットフォーム環境への対応

Matryoshka カプセルにおいては，内包するコントロールが動作しない環境では，コンテンツを利用することはできない．コンテンツプロバイダーにとっては，同一のカプセルで各種 OS に対応するカプセルの方が望ましいであろう．また，カプセルを利用するために，コンテンツ利用者が専用のヘルパーアプリケーションをインストールしなくてはならないのも望ましくない．

本実装カプセルは，コントロールをすべて Java によって実装しているが，動画の処理に用いられる JMF ライブラリが，現状では標準パッケージに入っていないため，Java 標準の実行環境 (Java2 Runtime Environment, Standard Edition 等)の他に，別途 JMF ライブラリを準備

する必要がある．特定 OS に最適化された JMF-PP を用いるときには必ず事前のインストールが必要である．クロスプラットフォーム対応の JMF-CP を用いる場合には，同様に事前にインストールすることも可能であるが，コントロールともにカプセル内必要な JMF ライブラリを内包することも可能である．このようなライブラリ内包型のカプセルは，Java 標準の実行環境があれば，どこでもカプセルを利用することができ，可搬性が大きく高まる．この時カプセル内に内包される JMF ライブラリのサイズは，QuickTime ムービー(H263,ULAW コーディング)対応の場合で，約 800KB であった．現在のダイアルアップ環境でのダウンロードを想定すると，データ量の増加は大きな負担となるが，数百メガバイトの動画コンテンツへの追加であれば，増加量は 1 % に満たない量であり，無視できる範囲であると考えられる．

JMF は現在，動作スピード，対応フォーマットの点で，JMF-CP より JMF-PP の方が勝っている．動作環境に JMF-PP がインストールされている場合にはそちらを利用し，それ以外のときに内包ライブラリを使うような，柔軟なライブラリ選択機能を持たせることを考えている．

5. まとめ

Java を用いた動画コンテンツ配信用カプセルの設計と実装を行い，カプセルの生成と，利用制約条件に沿った実行が，正常に動作することを確認した．カプセルは，利用者認証/利用期限指定/利用期間指定/利用回数指定の情報に基づき，自律的に利用の可否を判断し，可能と判断した場合のみ，内包する復号機能付き表現手段を用いて，コンテンツを復号・再生する仕組みを備えている．本実装による動画カプセルは，コンテンツの他に，利用制約条件等のコンテンツ関連情報と，コンテンツを表示するコントロール機能を併せ持ち，これらの情報を JAR 形式でアーカイブした構造を持つ．カプセル内のコンテンツは暗号化状態で内包されており，外部からの不正なアクセスを防止している．動画再生にかかわる JMF 関連の処理を，プラットフォーム・動画フォーマット非依存の手法によって行い，コントロールも全てネイティブコードを使わない Java クラスとして実装したことから，1 つのカプセルを Windows, Linux, Solaris 等クロスプラットフォーム環境にて動作させることが可能となった．

現状のカプセルでは，内包される動画コンテンツはあらかじめ用意された 1 ファイルであり，利用制約条件も単一に設定することしかできない．今後は，複数の動画コンテンツのから必要なフレームを切り出し，それぞれの利用制約条件を継承しながら，新たな動画コンテンツを作成するといった，編集機能等を持つ動画カプセルの開発にも取り組む予定である．

参考文献

[1] http://www.music.co.jp

[2] http://www.labelgate.com

[3] 谷口，森賀，久松，櫻井，“マルチメディア情報ベースとその格納単位 Matryoshka”，情処 DICOMO'99 シンポジウム論文集, pp.207, 1999.

[4] 谷口，阿部，塩野入，“Java を用いた動画配信用著作権保護カプセル”，情処研報 DPS 98-6, pp.31, 2000.

[5] International Standard, ISO/IEC 11172-1

[6] http://java.sun.com/products/java-media/jmf/

[7] http://java.sun.com/products/java-media/jmf/2.1/formats.html