AN INTERPRETED APPROACH TO MULTIMEDIA STREAMS PROTECTION

C. Alberti, A. Romeo, M. Mattavelli, D. Mlynek
Swiss Federal Institute of Technology.
Integrated Systems Laboratory, ISL/LSI
CH-10151 Lausanne, Switzerland
E-mail: claudio.alberti@epfl.ch

ABSTRACT

Exploiting experiences accumulated during MPEG-4 Structured Audio [1] [5] process of standardization and OCCAMM [2] European project related to multimedia content protection, this paper proposes a new approach to Intellectual Property Management and Protection (IPMP) of multimedia content streams.

The definition of a sort of C-like language for IPMP tools description is proposed. The core of such language shall be a set of IPMP-primitives able to be combined and provide all the range of functionality required during content consumption. The suggested approach will also allow an implementation independent method of complexity evaluation. Moreover, a sort of IPMP Scene Description Language is here suggested as a suitable solution for the formalization of interactions between different IPMP tools cooperating during the process of content consumption.

The potential effectiveness of the above-mentioned structured approach has been recognized by the MPEG working group through the addition in the current IPMP Extensions Committee Draft of the opportunity for a structured description of IPMP tools.

1. INTRODUCTION

With the deployment of digital technology to deliver digital multimedia material to users a host of issues have surfaced, arising from the fact that audio-visual information have always carried intrinsic value but because of the poor performance of analogue media and their rapid deterioration in subsequent processing/copying, the protection of this value has in general not been an issue. With digital broadcasting (via satellite, Internet or terrestrial carriers) the traditional paradigm no longer holds because audio-visual content in its digital form has the ability to allow infinite replication without loss of quality. Content protection has therefore become a major issue. On the one hand content has to be protected so that access to it is enabled only to those who have acquired the right to do so, on the other hand content is to be protected so as to prevent its dissemination.

Further, the flexibility of digital technologies allows a major overhaul of the way content is accessed today. While today the economic models that can be used still suffer from the rigidity of analogue technologies and the reference to physical support media, the Internet provides almost limitless business models. This is reflected in the objective of many IPMP initiatives to provide value-chain participants with the ability to acquire, supply, process and consume multi-media services on a worldwide basis in accordance with the rights associated with these services. In other words content protection is to be extended to encompass "content management".

In this scenario interoperability among different manufacturers' products is of crucial importance to provide consumers the easiest accessibility to content.

Two kinds of interoperability can be defined and in this paper will be shown that the proposed solution can provide both:

1. Allow the same protected content to be consumed on different vendors' devices.
2. Allow the same content to be protected by different vendors' IPMP tools.

2. TOOLS DESCRIPTION

Following the success of MPEG-4 Structured Audio Orchestra Language (SAOL) [1] [4] [7] [8], conceived and currently used to describe instruments as a network of primitives producing the desired sound synthesis or processing, we propose to define a set of IPMP-primitives able to be combined and provide all the range of functions required during content consumption. For instance such primitives shall be able to provide functionality needed when performing cryptographic algorithms. In case of symmetric encryption/decryption, a DES algorithm would be described in terms of a sequence of primitives constituting the analog of a SA instrument (that henceforth will be called either IPMP instrument) receiving the ciphertext as input and providing the plaintext as output.

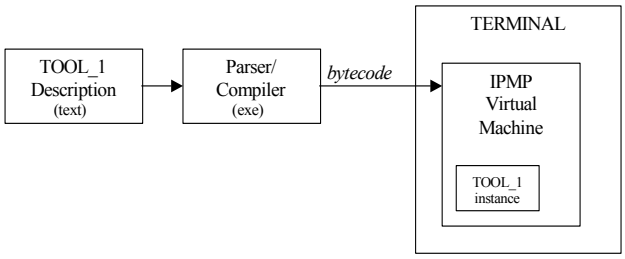

Figure 1. *Tool description parsing, compiling and download into the terminal.*

Every IPMP tool would be described through this language either as a single IPMP instrument or as a set of instruments composing an IPMP orchestra. The description would be parsed and compiled by suitable parsers and compilers whose output shall be a bytecode to be downloaded into the recipient terminal where an IPMP virtual machine will be able to instantiate and use the tool (see figure 1). We use here the term virtual machine since it is a platform-dependent executable able to execute a platform-independent (byte)code.

An exact specification of primitives' implementation is not necessary. Only their input-output relation is required to be standardized in order to allow different developers to exchange IPMP tools without risk of incompatibility. This implies that a given description of an IPMP instrument shall correspond to a unique bytecode.

The ability to describe IPMP tools in terms of sets of basic primitives to be executed a certain number of times during tool use will provide an implementation independent method of complexity evaluation. In order to achieve meaningful measurements a complexity vector (i.e. a set of basic primitives or classes of primitives) shall be defined. Every element of this vector will be a counter of the number of times a primitive (or primitives belonging to a class of primitives) is used during the IPMP tool execution. The final complexity will be quantified through the values of the different vector dimensions.

3. TOOLS INTERACTION

In case the multimedia content consumption requires the use and interaction of different IPMP tools coming from different vendors, they are likely to be described by different IPMP orchestras. The authors propose that the way in which every tool has to cooperate with the others has to be specified by means of an IPMP Scene Description Language similar to MPEG-4 Binary Format for Scene (BIFS) [3]. The scene description shall be coded independently from streams related to primitive media objects so that it will not be necessary to decode the objects in order to access and if necessary modify parameters describing the scene.

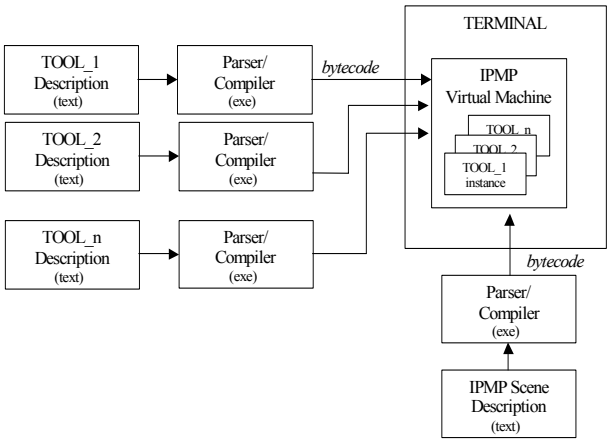

Figure 2 *A process of content consumption may require several interoperating IPMP tools.*

Here the different IPMP tools will play the role of BIFS nodes and their relationship will be described in terms of a hierarchical structure. Each node is linked to one or several other nodes in a non-static way. This means that node attributes can be changed while nodes can be added, replaced, or removed. This scenario introduces the need for a normative entity able to coordinate the different IPMP tool instances running inside the virtual machine. OPIMA architecture approach investigated in the framework of OCCAMM European project seems to be suitable to offer a platform to conceive this normative entity. Actually this architecture appears as the best candidate to interpret and execute languages oriented to secure multimedia content handling. Some modifications are likely to result unavoidable in order to best fit the set of primitives that will be defined and thus provide the maximum degree of effectiveness.

As for tools description, scene description has to be parsed and compiled in order to produce the bytecode to be downloaded into the terminal and allowing a correct synergy among the several IPMP tools instances. The exact specification of this parser and compiler shall be given only in terms of input-output relation (a given description corresponds to a given bytecode). Such a description shall be downloaded into the terminal with the content since the content vendor will have to specify (in terms of IPMP scene description language) the exact IPMP environment in which its material has to be consumed. A schematic of this scenario is proposed in figure 2.

4. TOOLS CONFIGURATION

Once an IPMP instrument has been described in its functionality and placed at its own place inside the scene, it has to be configured and used in the right way. This goal

can be achieved by means of score language providing information concerning basically:

- exact time in which a particular IPMP instrument has to be instantiated,
- duration of its performance (or exact time in which it has to be terminated),
- parameters for initialization.

For instance, if a process of content consumption requires the interaction with a remote IPMP tool, we can imagine that through a score event an IPMP instrument is properly instantiated and configured. Then it produces the appropriate message to be sent and stops. The remote tool, once the message is received, produces the answer and a score event aimed at activating the IPMP instrument instance recipient of the message. This provides an asynchronous mechanism of IPMP tool interaction.

By means of this score language it will also be possible to access some control variables inside the different IPMP instruments if they have been declared as exposed (i.e. accessible for modification) in the description. This feature will provide a powerful run-time tuning mechanism.

A score is a list of commands. A command performs a single action at a moment in time, such as changing the value of an exposed field or creating a new instance of an IPMP instrument (providing the set-up settings). The instantiation of new IPMP tools only requires that the bytecode related to the new tool description has already been downloaded into the terminal.

5. AN EXAMPLE OF TOOL DESCRIPTION

A very simple example is here provided in order to better illustrate the idea of IPMP tools description.

The described tool performs a classic digital signature generation and verification algorithm. In bold are functions that could belong to the primitive instructions set to be defined.

It is the Digital Signature Algorithm taken from [6] where the necessary keys are generated and then used to sign the document and verify its signature.

The example shows that typical primitives that are likely to be inserted into the core set of functions are operations modulo a generic integer *n* (*exp_mod, invert_mod, mult_mod*), operations on great numbers (*long_modulus*), basic cryptographic functions (*SecureHashAlgorithm*) and complex mathematical operations (*select_from_multiplicative_group*). The study and analysis of the widest range of currently used algorithms will assure the core primitives set to cover the largest variety of usage scenarios.

```
// SUMMARY: entity A signs a binary message m of arbitrary length. Any entity B can verify this
// signature by using A's public key.

DSAKeyGeneration(){

bit(160)        q, a;
bit(1024)       p, g, alpha, y;
uint            t, k, h;

// Key generation for the DSA

        alpha = 1;

        // Select a prime number  q such that 2^159 < q < 2^160 .

        while(!is_prime(q))
                q = select_random_pow2(159, 160);

        // Choose t so that 0 =t =8, and select a prime number p where 2^(511+64*t) < p < 2^(512+64*t) ,
        // with the property that q divides (p-1).
        while(long_modulus((p-1), q)){
                t = select_random(0,8);
                k = 64 * t;
                p = select_random_pow2(511 + k, 512 + k);
        }

        // Select a generator α of the unique cyclic group of order q in Z_q^*
        // Select an element g ∈ Z_q^* and compute α = g^((p-1)/q) mod p.
        while(alpha == 1){
                g = select_from_multiplicative_group(p);
                h = long_divide((p-1), q);
                alpha = exp_mod(g, h, p);
                }

        // Select a random integer a such that  1 =a =(q – 1)

        a = select_random(1, (q - 1));

        // Compute y = α^a mod p

        y = exp_mod(alpha, a, p);

// A's public key is (p, q, alpha, y); A's private key is a.
        output(p, q, alpha, y, a);

DSASignatureGeneration(p, q, alpha, y, m, a){

// Entity A should do the following:

bit(160)        k, invk, s, r;
bit(1024)       h;

// Select a random secret integer k, 0 < k < q

while((k == 0) OR (k == q))
        k = select_random(0, q);

// Compute r = (α^k mod p) mod q

h = exp_mod(alpha, k, p);
r = long_modulus(h, q);

// Compute k^-1 mod q.

invk = invert_mod(k, q);

// Compute s = (k^-1 mod q) * {h(m) + a * r} mod q.

hash_code = SecureHashAlgorithm(m);
s = long_modulus(invk * (hash_code + a *r), q);

// A's signature for m is the pair (r, s).

output(r, s);
}
```

```
DSASignatureVerification(p, q, alpha, y, r, s, m){

bit(160)    w, u1, u2, v, hash_code;
bit(1024)   k, j;

// To verify A's signature (r, s) on m, B should do the following:

// Obtain A's authentic public key (p, q, alpha, y).
// Verify that 0 < r < q and 0 < s < q; if not then reject the signature.

if (!((0 < r < q) AND (0 < s < q)) output (FALSE);

else{
        // Compute w = (s^-1 mod q) and h(m)
        w = invert_mod(s, q);
        hash_code = SecureHashAlgorithm(m);

        // Compute u1 = w * h(m) mod q and u2 = (r * w) mod q
        u1 = long_modulus(w * hash_code, q);
        u2 = long_modulus(r * w, q);

        // Compute v = (α^u1 * y^u2 mod p) mod q.
        k = exp_mod(alpha, u1, p);
        j = exp_mod(y, u2, p);
        v = mult_mod(k, j, q);

        // Accept signature if and only if v=r.
        if(v == r) ouput(TRUE);
        else output(FALSE);
}
```

Figure 3 *Example of Digital Signature Algorithm described through the proposed approach.*

6. CONCLUSIONS

According to past experience in SA [4] [7] [8], the proposed approach provides a consistent number of advantages that we try here to summarize.

The tool description in terms of network of primitives

1. allows a range of possible tools creation as wide as the definition of the base primitives set is appropriate;
2. provides a simple way to redesign IPMP tools when they should become no more trustworthy for implementation bugs or algorithm intrinsic weaknesses;
3. does not need the sensible content to exit the terminal since the only data transfer involved is that concerning the bytecode download (on the other hand this shall be encrypted, but the amount of data is likely to be quite small);
4. allows a precise complexity evaluation in terms of number of basic instruction to be performed per time-frame and thus implementation independent. These complexity measurements could lead to a precise definition of levels and profiles.

The scene description in terms of dedicated language

1. provides a simple way to reconfigure the interaction between different IPMP tools even during the content consumption;
2. if coded in textual format and downloaded as bytecode, requires a low bit-rate data exchange with the terminal, keeping any sensible information inside it;
3. implies the definition of a normative scheduler supervising the overall execution, allowing the exploitation of efforts already produced in the framework of successful past projects [2].

The tool configuration by means of a dedicated score language

1. allows the run-time instantiation of new IPMP tools through events generation;
2. is suitable for a fine and direct control on IPMP tools performances thanks to the exposed fields mechanism.

7. REFERENCES

[1] ISO/IEC JTC 1/SC 29/WG11 - ISO/IEC FDIS 14496-3 sec5

[2] Project OCCAMM – IST-1999-11443 Deliverable D1 - Project description

[3] ISO/IEC JTC 1/SC 29/WG11 - ISO/IEC FDIS 14496-1 2000(E)

[4] G. Zoia, "A method for Complexity Measurements in Structured Audio". ISO/IEC JTC1/SC29/WG11 (MPEG98) document M3602, Dublin - July 1998.

[5] E. Scheirer, "SAOL: the MPEG-4 Structured Audio Orchestra Language". *In Proceedings of the International Computer Music Conference*. Ann Arbor, MI, October 1998.

[6] Menezes A.J., van Oorschot P.C., van Stone S. *A. , Handbook of Applied Cryptography* – CRC Press.

[7] G. Zoia, C. Alberti, "A virtual ALU for MPEG-4 Structured Audio on parallel architectures". ISO/IEC JTC1/SC29/WG11 (MPEG99) document m4706, Vancouver – July 1999.

[8] G. Zoia, C. Alberti, "A virtual DSP architecture for MPEG-4 Structured Audio". *In Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00),* Verona, Italy, December 7-9, 2000