Automatic Smart Contract generation for Internet of Media Things

Mohamed Allouche, Mihai Mitrea, Alexandre Moreaux, Sang-Kyun Kim

Available online at www.sciencedirect.com

KICS The Korean Institute of Communications and Information Sciences

ScienceDirect

ICT Express * (2019) 1-4

www.elsevier.com/locate/icte

Automatic Smart Contract Generation for Internet of Media Things

Mohamed Allouche[1], Mihai Mitrea[1,*], Alexandre Moreaux[1], Sang-Kyun Kim[2]

[1] *ARTEMIS Department, SAMOVAR Laboratory, Telecom SudParis – Institut Polytechnique de Paris, Palaiseau, France*
[2] *College of ICT Convergence, Myongji University, Seoul, Korea*

Received dd mmm yyyy; accepted dd mmm yyyy (8p)
Available online dd mmm yyyy

Abstract

Formally known as ISO/IEC 23093, Internet of Media Things (IoMT) is an emerging paradigm ensuring interoperability among media-centric applications and services designed and deployed for the interpretation, representation or analysis of multimedia content collected by media devices such as cameras or microphones. Under this framework, the present paper establishes the proof of concept for the automation of IoMT specified Smart Contract generation. The advanced *modus operandi* enables media devices and their content to be seamlessly protected against payment counterfeiting, malicious control, access, interception and/or redirection. To this aim, a comprehensive architectural and experimental framework is conceived, designed, and demonstrated. While the methodological approach is blockchain agnostic, the experimental results are obtained on a 3-node, EEA (Enterprise Ethereum Alliance)-compliant private blockchain.

Keywords: Internet of Media Things, multimedia content protection, automatic Smart Contract generation, blockchain agnostic architecture

1. Introduction

Within the field of Internet of Things (IoT), media-centric applications and services can offer the provision, interpretation, representation or even analysis of multimedia content collected by media devices such as cameras, microphones, vibro-haptic devices, etc. The Moving Pictures Expert Group (MPEG, formally known as ISO/IEC JTC1 SC29) introduced the notion of *Media Thing* (MThing), which is defined as a *Thing* capable of sensing, acquiring, actuating, or processing media content or metadata related with such content. The notion gave birth to IoMT (Internet of Media Things), an extension of the IoT paradigm for visual and audio devices introduced by MPEG and currently under the scope of the ISO/IEC 23093 Internet of Media Things (IoMT) standard family [1-4].

Providing reference architectures, APIs (Application Programming Interface) and data formats for the discovery and usage of MThing, IoMT facilitates the design and implementation of complex systems capable of processing massive multimedia content, thus bridging the gap between user expectancies and their feasibility [1]-[4].

The present paper contributes to the field by establishing the proof of concept for the joint use of blockchain technologies and IoMT. To this aim, a comprehensive architectural and experimental framework is advanced. The principle consists in automating the production of Smart Contracts around IoMT specifications while allowing fine-grade control and the tracking of device performed actions, from their activation to the transmission of generated content. This *modus operandi* protects devices and their content against payment counterfeiting, malicious control, access, interception and/or redirection and can be seamlessly integrated with any other complementary protection solution.

This paper is structured as follows. *Section 2* presents IoMT standard concepts and their synergies with blockchain solutions before we introduce the main paper contribution (the novel

*Corresponding author: mihai.mitrea@telecom-sudparis.eu
E-mail addresses: mohamed.allouche@telecom-sudparis.eu, alexandre_moreaux@telecom-sudparis.eu, goldmunt@gmail.com
Peer review under responsibility of The Korean Institute of Communications and Information Sciences.

https://doi.org/10.1016/j.icte.2018.01.001

architectural framework) in *Section 3*. We move onto the experimental framework in *Section 4* and conclude the paper in *Section 5*, discussing short-term working perspectives.

2. Context

2.1 Internet of Media Things (IoMT)

The ISO/IEC 23093 series (a.k.a. MPEG-IoMT) provides an architecture, specifies APIs, and details compressed representations of data flowing between MThings [2]-[4]. Various MThings (MCameras, MMicrophones, MDispalys, MAnalysers, or MStorages) can thus be designed, orchestrated, and operated in a large variety of tasks such as acquisition, rendering, processing, or multimedia content storage.

The IoMT standards also make provision for the use of blockchain solutions in conjunction with MThings. In fact, the standard already specifies the APIs and the usage of digital coins or legal tender to ensure the payment of Mthings for their use, as detailed hereafter for the case of an MCamera (similar mechanisms exist for every MThing).

2.2 Blockchain and IoMT

Figure1 shows the sequence diagram of a transaction process between a user and an MThing (e.g., MCamera), as specified by [3]. To watch a video captured from a camera, the user inquires the cost per minute for that MCamera, using the standard API (Figure1-1) with the desired media token defined by *tokenType* (e.g., cryptocurrency or legal tender) and *tokenName* (e.g., Bitcoin or US dollar). The MCamera returns the cost per minute to let the user access the live video by using the *getVideoURL()* function (Figure1-2). If the user agrees to the price, he/she asks for a wallet address (Figure1-3) of the desired type of currency (i.e., *MToken*). Again, the MCamera responds with the proper wallet address (Figure1-4) to which the user sends the agreed amount of *MTokens* through the blockchain (Figure1-5), which returns a transaction ID (Figure1-6) used to confirm the good standing of the payment (Figure1-7). With the transaction confirmed (Figure1-8), the user can ask for the video stream service to the MCamera using *getVideoURL(tid)* (Figure1-9). Before granting access, the MCamera uses *checkTransactionCompletion(tid)* (Figure1-10) to check the completion of the transaction and the amount of received MTokens (Figure1-11). Then, the MCamera returns a video URL and streams the video according to user rights (Figure1-12). Details about *getVideoURL, CostPerMinute()* and *getVideoURL ()* can be found in ISO/IEC 23093-3 while the details of *getWalletAddress(), sendTokens()*, and *checkTransactionCompletion()* are described in ISO/IEC 23093-2.

2.3 Current limitations

The above *Section 2.2* shows that although standard APIs and data formats are already specified by IoMT for blockchain integration, their use still requires complex operations (e.g., Smart Contract development) that must be solved on a case-by-case basis, as the process is *a priori* blockchain specific.

As each Blockchain operates with specific programming languages and constraints, the software development process becomes complex and doubts about the possibility of specifying a unitary solution arise. Moreover, conventional Blockchain architectures are prohibitive in terms of memory, computation, and energy resources. Hence, doubts about the possibility of executing IoMT related Smart Contracts in predefined tasks and limited time also exist.

As a preliminary step towards the reuse of Smart Contract code, [5] brought to light the idea of Smart Contract auto-generation from domain-specific ontologies. The following *Section 3* goes one step further and presents a solution for the automation of IoMT Smart Contract generation and a way to logically make the process blockchain agnostic.

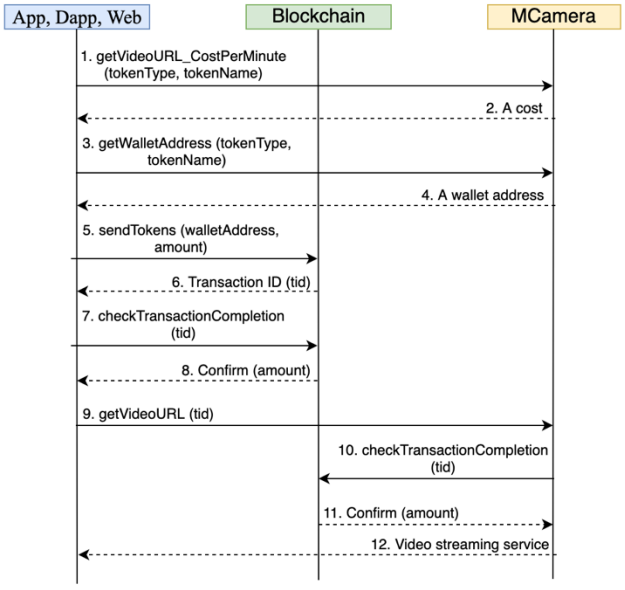

Figure 1. Sequence diagram between a user and an MCamera using the blockchain [3].

3. Advanced solution

The main innovation disclosed by our study is a generic, blockchain-agnostic architectural framework and its specified, designed, and implemented underlying modules. The architecture is centered around the automatic production and execution of Smart Contracts from IoMT information, as presented in Figure2. It combines expertise from both IoMT and blockchain Experts and it is composed of three main components: the *IoMT Parser*, the *Smart Contract Developer*, and the *Blockchain Manager*.

The process starts with an IoMT expert who deploys and/or exploits an IoMT system in which at least one blockchain is required for the operation of at least one MThing. Hence, the IoMT expert designs and develops the software application based on the blockchain-specific IoMT APIs and data formats, as illustrated in *Section 2.2*. Yet, the IoMT expert is unaware of the technical complexity of blockchain operations.

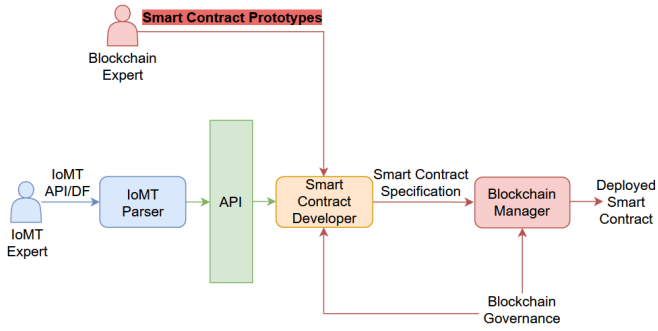

Figure 2. Architectural framework for automating the conversion of blockchain IoMT API and data formats into Smart Contracts.

The *IoMT Parser* extracts the relevant specifications for a blockchain operation (e.g., Figure 1) and returns a structured set of transactional rules and conditions. This module is independent of the blockchain, and its output can be accessed through a generic API.

In parallel, the Blockchain Expert (with deep skills in the blockchain technology but with limited knowledge regarding IoMT standards) elaborates a set of specific *Smart Contract Prototypes* for IoMT usage. Although these *Smart Contract Prototypes* are logically independent with respect to the blockchain solution, their code depends on the specific programming language supported by the blockchain in context.

By combining IoMT and blockchain information (*Blockchain Governance* and *Smart Contract prototypes*), the *Smart Contract Developer* produces the Smart Contract specification. Note that the *Smart Contract Developer* has both generic and blockchain specific input information thanks to the API on the one hand, and *Blockchain Governance* and *Smart Contract prototypes* on the other; yet its specification and implementation are blockchain agnostic.

The *Blockchain Manager* is the native module of the specific blockchain and is considered to be the exit point of our architecture as it deploys the contract.

4. Experimental framework

4.1 Testing environment

The experiments reported in this paper use a 3-node, EEA (Enterprise Ethereum Alliance)-compliant Proof of Authority private blockchain deployed on an AWS server. The 3 nodes are validators and run on different ports of the same server using Hyperledger Besu. All 3 are complemented with their own Orion nodes that act as Private Transaction managers. Users can communicate with the nodes through EthSigner proxies that have access to the private keys of 3 accounts made available for testing purposes. Alethio Lite Explorer is used to help us monitor blockchain activity.

The architecture and its modules are implemented in Python and JavaScript: the software is made available as open source under Apache license (upon paper publication). Illustrative parts of the corresponding data formats / files are included in Section 4.2.

4.2 Workflow illustration

This section illustrates the generic information flow related to the conversion of an IoMT specification (for an MThing or a collection of MThings) into a Smart Contract, for the case of an MCamera and the previously specified EEA blockchain.

We assume the architectural modules to be deployed on the blockchain infrastructure and the *Blockchain Expert* to have already created the *Smart Contract Prototype*. This prototype is composed of a set of functions matching individual IoMT specifications to their blockchain counterparts, *cf.* Figure 3.

MCamera.sol

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.8.0;
import "./MCamFT.sol";

contract MCamera {
    mapping(string => int256) public MPEGVCameraCryptocurrencyCost;
    mapping(string => int256) public MPEGVCameraLegalTenderCost;
    mapping(string => int256) public videoURLCryptocurrencyCost;
    mapping(string => int256) public videoURLLegalTenderCost;
    mapping(string => int256) public imageURLCryptocurrencyCost;
    mapping(string => int256) public imageURLLegalTenderCost;
    mapping(string => string) public cryptocurrencyWalletAddress;
    mapping(string => string) public legalTenderWalletAddress;
    string id;
    string serverIPAddress;
    int serverPort;
    string MPEGVCamera = "sensed data defined by CameraSensorType from ISO/IEC 23005-5 (MPEG-V Part 5)." ;
    string videoURL = "THIS IS VIEDO URL";
    string imageURL = "THIS IS IMAGE URL";
    MCamFT public mCamFT;
    event startMPEGV ( address indexed _from, string _value );
    event startVideo ( address indexed _from, string _value );
    event startImage ( address indexed _from, string _value );
// constructors
    constructor (string memory _id ,string memory _serverIPAddress ,int _serverPort) {…
    }
// internal Functions
    function multiply(uint256 x, uint256 y) internal pure returns (uint256 z) {…
    function buyTokens(uint256 _numberOfTokens) public payable {…
// external Functions
    function getMPEGVCamera() public view returns(string memory){…
    function getMPEGVCamera(string memory tid) public returns(string memory){…
    function getMPEGVCamera_Cost(int8 tokenType, string memory tokenName) public view returns (int256){…
    function getVideoURL(string memory tid) public returns (string memory){…
    function getVideoURL() public view returns (string memory){…
    function getVideoURL_CostPerMinute(int8 tokenType, string memory tokenName) public view returns (int256){…
    function getWalletAddress(int tokenType, string memory tokenName) public view returns (string memory){…
    function getImageURL() public view returns (string memory){…
    function getImageURL(string memory tid) public returns (string memory){…
    function getImageURL_Cost(int8 tokenType, string memory tokenName) public view returns (int256) {…
}
```

Figure 3. Illustration of the *Smart Contract Prototype*.

Assuming the *IoMT Expert* creates a new application around the video stream produced by an MCamera, the IoMT Parser automatically extracts the set of information required by the blockchain and presents it to the *Smart Contract Developer* through a generic *API*, as illustrated in Figure 4.

Based on the *IoMT Parser* output and the *Smart Contract Prototype*, the *Smart Contract Developer* automatically (with no Blockchain Expert intervention) creates the Smart Contact (in this example, written in Solidity) that is then deployed by the *Blockchain Manager*, as illustrated in Figure 5.

4.3 Discussion of the results

The results presented in the previous section establish the proof of concept for the automation of IoMT Smart Contract production (see Figure 2). Once the *Smart Contract Prototype* is established, a Smart Contract can automatically be deployed for each and any new IoMT application without the need for human code development. The *Smart Contract Prototype* code only requires updates when new versions of the standard are released (about every 2-3 years).

These conceptual and applicative contributions are still limited to the specific case of the studied Blockchain, namely the EEA (Enterprise Ethereum Alliance). Arriving at a complete solution would require two types of activities to be carried out: adapting the *Smart Contract Prototype*'s syntax (while keeping its semantics) for each blockchain and specifying an open API for exchanging Blockchain Governance information.

Figure 4. Illustration of the *IoMT Parser* output.

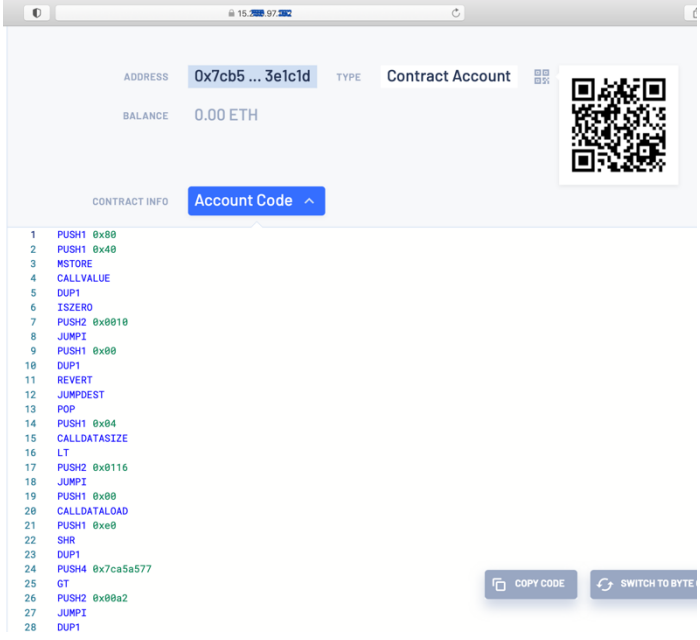

Figure 5. Automatically generated IoMT Smart Contract.

5. Conclusion

This paper establishes the PoC for the automatic production of Smart Contracts related to IoMT standard specifications. The main contribution consists in conceiving, designing, and demonstrating an architectural framework that is agnostic with respect to the blockchain in context. It is also demonstrated that IoMT-related Smart Contracts can be executed for predefined tasks in a limited time despite the memory, computational, and energy constraints in conventional Blockchain solutions.

The main benefit of this solution is at the operational level: once the architecture is set and deployed, it can serve applicative needs without the implication of Smart Contract developers. In fact, the Smart Contract is automatically generated and deployed for each new IoMT compatible workflow. This leverages high added value business models for the realms of IoMT and blockchain.

As a final remark, the advanced solution is positioned at the IoMT device level and can be integrated seamlessly with any other high level content protection and/or security application, such as traditional security solutions (firewalls, authentication tokens, certificates, watermarking, fingerprinting, etc.) or novel blockchain based solutions. Short-term future work will be devoted to the definition and deployment of a NFT (non-fungible token) controlling the device's temporal operations (time stamping controlled access) as well as the integrity of its stream content and origin.

Acknowledgments

Mohamed Allouche's contribution to this paper was mainly achieved during his MS internship at Telecom SudParis. He is currently a PhD student with Vidmizer (https://vidmizer.com/en/) and Telecom SudParis. Alexandre Moreaux' PhD thesis is developed under the framework of a tight R&D partnership between Blockchain Secure (https://blockchainsecure.io) and Telecom SudParis.

Conflict of interest

No conflict of interest relates to the present paper.

References

[1] M. Mitrea, IoMT White Paper, ISO/IEC JTC1/SC29/WG11 N18879, Oct. 2019, Geneva CH; https://mpeg.chiariglione.org/standards/mpeg-iomt/iomt-white-paper (last visited: March 19, 2021)

[2] ISO/IEC 23093-1:2020 Information technology – Internet of media things – Part 1: Architecture, https://webstore.iec.ch/publication/66671 (last visited: March 19, 2021)

[3] ISO/IEC 23093-2 Information technology – Internet of media things – Part 2: Discovery and communication API, https://www.iso.org/standard/74332.html (last visited: March 19, 2021)

[4] ISO/IEC 23093-3:2019 Information technology – Internet of media things – Part 3: Media data formats and APIs, https://www.iso.org/standard/74333.html (last visited: March 19, 2021)

[5] O. Choudhury, N. Rudolph, I. Sylla, N. Fairoza, A. Das, 'Auto-Generation of Smart Contracts from Domain-Specific Ontologies and Semantic Rules', IEEE Blockchain, Halifax, Canada, 30 July-3 Aug. 2018.