Poster: Securing IoT services using DLTs and Verifiable Credentials

Nikos Fotiou, Iakovos Pittaras, Vasilios A. Siris, Spyros Voulgaris, George C. Polyzos
Mobile Multimedia Laboratory,
Department of Informatics School of Information Sciences and Technology
Athens University of Economics and Business, Greece
{fotiou,pittaras,vsiris,voulgaris,polyzos}@aueb.gr

***Abstract*—Verifiable Credentials (VCs) and blockchain-based tokens have some intriguing properties when it comes to user authentication and authorization. For instance, VCs can encode an access control policy and provide privacy, whereas blockchain-based tokens enable auditability and accountability, facilitate revocation, and create potential for novel token exchange methods. Here, we present our recent progress in integrating these technologies into the OAuth 2.0 authorization protocol. Our efforts focus on push and pull Internet of Things (IoT) services, accessed through IoT gateways. We present design trade-offs, we provide preliminary results related to the communication and computation overhead, and we discuss future directions.**

I. Introduction

Securing IoT services requires proper user authentication and authorization. However, implementing these operations directly in the IoT devices creates security risks and management issues. For this reason, most IoT systems delegate user authentication and authorization to a third party using solutions such as the OAuth 2.0 authorization framework [1]. Using OAuth 2.0, a *resource owner* can generate an *authorization grant*, which a *client* can use with an *authorization server* in order to receive an *access token*. Then, the access token can be used for accessing a protected resource, stored in a *resource server*. However, OAuth 2.0 specification does not define any particular grant or token format, instead this is left as a design choice for the system developers. In this work, we explore the potentials of using *Verifiable Credentials* (VCs) as authorization grants, and blockchain-backed access tokens.

VCs [2] allow an *issuer* to assert one or more *claims* about a *subject*. A VC includes a set of claims, metadata that describe properties of the credential, as well as a digital proof. A VC is issued to a *holder*. In most cases, a subject and a holder are the same entity. A holder, can prove the possession of one or more VCs to a *verifier* by generating a *verifiable presentation*. A verifiable presentation includes data from one or more verifiable credentials, and is packaged in such a way that the authorship of the data is verifiable [2]. When it comes to authorization, VCs have two significant advantages: (i) they support improved privacy, since they disclose minimal information about the holder, and (ii) they can be used to indicate what a holder is allowed to do, i.e., an issuer can encode a simple access control policy in a VC.

Blockchains, and more generally Distributed Ledger Technologies (DLTs) can be a regarded as an immutable, distributed ledger. Some blockchain systems, such as Ethereum [3], are capable of running (distributed) applications, often referred to as *smart contracts*. Blockchains have great availability and robustness, hence they are a promising solution for recording auxiliary information about access tokens, which can be used for accountability, auditability, as well as for token verification. Furthermore, smart contracts can be used for verifying access tokens, as well as for implementing novel token management mechanisms, e.g., exchanging tokens for money.

We summarize here our efforts (undertaken in the context of the H2020-SOFIE project[1]) for realizing a system that can be used for accessing protected IoT resources. Our system is standards-compliant and it leverages OAuth 2.0, VCs, and blockchain-backed access tokens.

II. System overview

A. Entities and interactions

Our system is a typical OAuth 2.0 architecture realization, hence it is composed of the following entities. A *resource server* that hosts a protected resource owned by a *resource owner*, a *client* wishing to access that resource, and an *authorization server* responsible for generating *access tokens*. These entities interact with each other as follows. A client first requests an authorization grant from the resource owner. This grant verifies that the "resource owner authorizes the client to access the resource." Then, the client uses the grant to obtain an access token from the authorization server. Finally, the client accesses the protected resource by proving the access token ownership.

B. VC-based authorization grants

VCs can play the role of the authorization grant. This design choice is compatible with OAuth 2.0 RFC which specifies that "client credentials (or other forms of client authentication) can be used as an authorization grant" (section 1.3.4 of [1]). In our system, a resource owner generates VCs for the clients authorized to access a resource. VC generation is performed independently of the authorization server. Then, a client can request an access token from an authorization server by presenting a VC: the authorization server issues a VC *proof request* and the client generates the appropriate *proof*. If the latter proof is valid, the authorization server proceeds with the token generation process. The whole process requires no interaction with the resource owner. Furthermore, the authorization server learns no information about the client, apart from the

[1] https://www.sofie-iot.eu/

fact that it is authorized to access a resource: even if the same client tries to generate a new token, for the same resource, the authorization server will not be able to tell if this is the same client or not. Finally, authorization servers can be pre-configured with the appropriate proof request parameters: in that case, the only operation an authorization server has to perform is the verification of the correctness of a proof.

C. Blockchain-backed token life cycle management

1) Token generation: Our system uses JSON Web Tokens (JWTs) [4] for encoding access tokens. The generated token includes a *proof of possession* key which is either a client owned public key, or a client owned Ethereum address. This key is used by the client to prove access token ownership, when accessing a resource. Furthermore, the authorization server creates a new ERC-721[2] *Ethereum token*, i.e., a unique, non-tangible token, that includes in its metadata field the access token. ERC-721 tokens are used in our system for auditing and accountability purposes, and optionally for providing auxiliary information to IoT resources during access token verification.

2) Token delivery: Our system considers two approaches for delivering access tokens from authorization servers to clients: direct delivery, and conditional delivery. In the former case, access tokens are sent directly to the client (over a secured channel). Moreover, if the proof of possession key included in the access token is client's Ethereum address, then the corresponding ERC-721 token is *transferred* to the client's address. With conditional delivery, a smart contract assures that an access token is usable only if certain conditions are met (e.g., the client has paid a pre-agreed amount of money). The smart contract guarantees the *atomicity* of the process; the means achieving that depend on the token verification method: it can be done by encrypting the access tokens and by revealing the decryption key once the conditions have been met, using "hash-locks" and "escrows" [5], or by transferring the ERC-721 token to the client's address only once the payment has been made.

3) Token verification: Depending on whether the client interacts with the resource server directly, or through the Ethereum blockchain (e.g., for triggering pull-based services using Ethereum "events" [6]) token verification can be performed either by a smart contract or the resource server itself. In the latter case, the verification method depends on whether the resource server can read from the blockchain or not (e.g., due to limited computational power, connectivity restrictions, or other constrains). If the resource server has read access to the blockchain, then access token verification involves the retrieval of the ERC-721 token and the comparison of its metadata with the access token presented by the client. Using this verification method, tokens can be easily revoked by the authorization server (by modifying the corresponding ERC-721 token), and they can be easily and securely delegated to other users [7]. If the resource server cannot access the blockchain, then the standard JWT token verification process is followed.

[2]https://eips.ethereum.org/EIPS/eip-721

III. Preliminary results

Parts of our system have been implemented and various performance measurements have been made. For this purpose we have extended OAuth 2.0 php server[3] and we have used Mozilla's WebThings gateway[4], running on a Raspberry Pi, as a resource server. For generating and managing VCs, Hyperledger Indy [8] has been used. A VC proof is generated in 101ms using a PC equipped with an Intel-i7 7700 CPU and 4GB RAM and it can be verified in 58ms in a PC with the same specifications. The creation of an ERC-721 token that contains in its metadate a JWT access token "consumes" 254141 Ethereum 'gas' units, which in the public Ethereum network is translated to approximately $0.04 and requires, in average, 13 seconds. Similarly, exchanging a token with 'ETH coins,' i.e., the Ethereum-specific cryptocurrency, consumes 102476 'gas' units, which in the public Ethereum network cost approximately $0.016.

IV. Concluding remarks and future plans

We presented a preliminary system that incorporates Verifiable Credentials and blockchain-backed tokens into the OAuth 2.0 framework. Our design is compatible with the OAuth 2.0 protocol, therefore existing systems can be easily extended to include our approach. Furthermore, our design is composed of several stand-alone "pluggable modules," hence an integrator may choose those that fit his/her requirements. We have verified the feasibility of each module, and have performed some preliminary performance evaluation investigations. It is in our immediate plans to continue the development of these modules and to provide them as, open source, re-usable "framework components."

Acknowledgment

This research was supported by the EU funded Horizon 2020 project SOFIE (Secure Open Federation for Internet Everywhere), under grant agreement No. 779984.

References

[1] D. Hardt (ed.), "The OAuth 2.0 authorization framework," IETF, RFC 6749, 2012.

[2] Manu Sporny et al. (2019) Verifiable credentials data model 1.0. [Online]. Available: https://www.w3.org/TR/verifiable-claims-data-model/

[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014.

[4] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," IETF, RFC 7519, 2015.

[5] V. A. Siris, D. Dimopoulos, N. Fotiou, S. Voulgaris, and G. C. Polyzos, "OAuth 2.0 meets blockchain for authorization in constrained IoT environments," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, April 2019, pp. 364–367.

[6] N. Fotiou, I. Pittaras, V. A. Siris, S. Voulgaris, and G. C. Polyzos, "Secure iot access at scale using blockchains and smart contracts," in *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, June 2019, pp. 1–6.

[7] N. Fotiou, I. Pittaras, V. A. Siris, S. Voulgaris, and G. C. Polyzos, "OAuth 2.0 authorization using blockchain-based tokens," in *Workshop on Decentralized IoT Systems and Security (DISS 2020), in conjunction with the NDSS Symposium 2020*, San Diego, CA, USA, 2020.

[8] The Linux foundation. (2019) Hyperledger indy home page. [Online]. Available: https://www.hyperledger.org/projects/hyperledger-indy

[3]https://github.com/bshaffer/oauth2-server-php

[4]https://iot.mozilla.org/

Securing IoT services using DLTs and Verifiable Credentials

Nikos Fotiou, Iakovos Pittaras, Vasilios A. Siris, Spyros Voulgaris, George C. Polyzos
Mobile Multimedia Laboratory
Department of Informatics, School of Information Sciences and Technology
Athens University of Economics and Business, Greece
{fotiou,pittaras,vsiris,voulgaris,polyzos}@aueb.gr

Abstract: Verifiable Credentials (VCs) and blockchain-based tokens have some intriguing properties when it comes to user authentication and authorization. For instance, VCs can encode an access control policy and provide privacy, whereas blockchain-based tokens enable auditability and accountability, facilitate revocation, and create potential for novel token exchange methods. Here, we present our recent progress in integrating these technologies into the OAuth 2.0 authorization protocol. Our efforts focus on push and pull Internet of Things (IoT) services, accessed through IoT gateways. We present design trade-offs, we provide preliminary results related to the communication and computation overhead, and we discuss future directions.

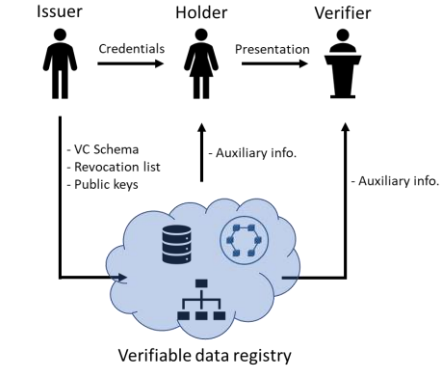
Contributions in a nutshell

- We use *Verifiable Credentials* as OAuth 2.0 authorization grants
- We propose blockchain-based access token lifecycle management
  - Auxiliary information can be recorded in the blockchain
  - Tokens can be transferred through the blockchain
  - Tokens can be exchanged with tangible assets
  - Tokens can be delegated
- We consider both powerful and constrained IoT gateways
- OAuth 2.0 compatible implementation
- Incorporated with a *Web of Things* gateway

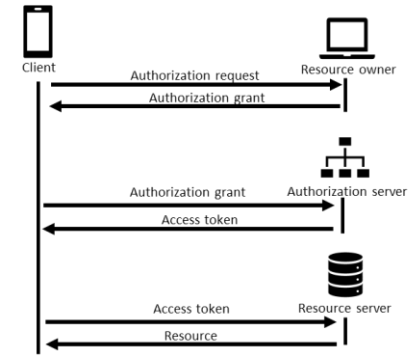
Advantages

- Authorization server implementation is simpler, since access control policies can be incorporated into VCs
- VCs enhance end-users' privacy
- Tokens can be revoked prior their expiration time
  - Without requiring communication between authorization servers and resource servers
- Auditing and accountability mechanisms are facilitated
- Clients do not need to store tokens or secrets corresponding to them

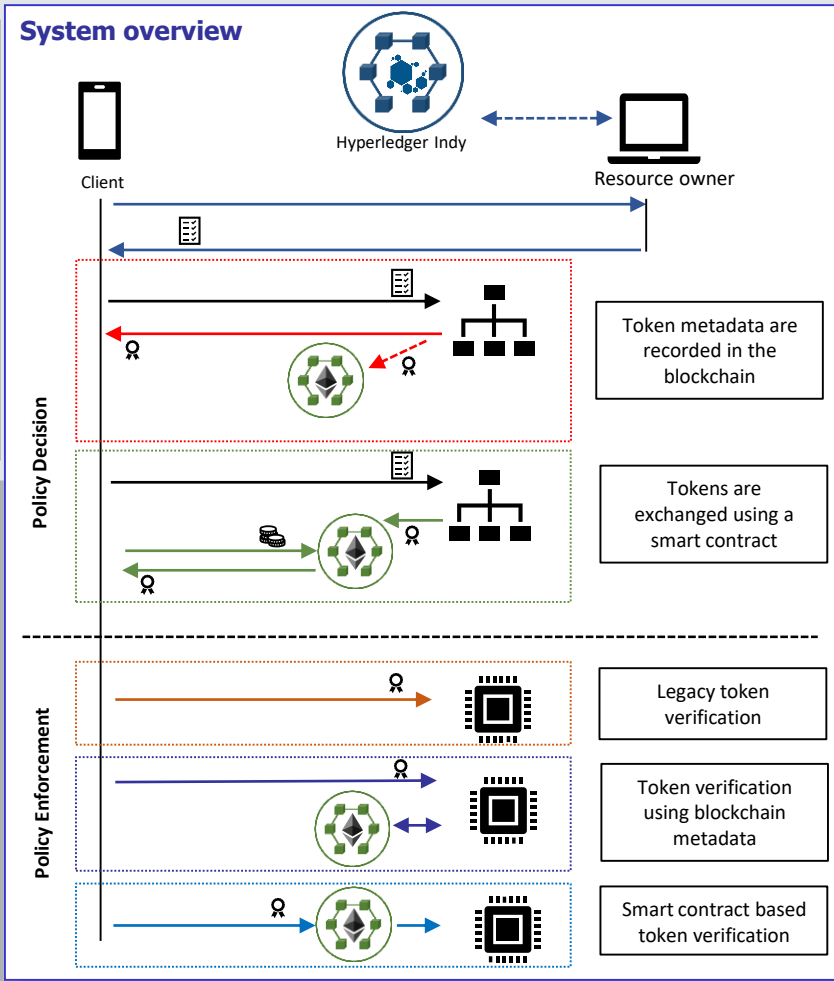
Verifiable credentials



OAuth 2.0



System overview



NDSS Symposium 2020
Find out more about our research at https://mm.aueb.gr
Find our more about H2020 SOFIE at https://www.sofie-iot.eu