Chapter 5
Enhancing Security in a Big Stream Cloud Architecture for the Internet of Things Through Blockchain

Luca Davoli
University of Parma, Italy

Laura Belli
University of Parma, Italy

Gianluigi Ferrari
University of Parma, Italy

ABSTRACT

The Internet of Things (IoT) paradigm is foreseeing the development of our environment towards new enriched spaces in most areas of modern living, such as digital health, smart cities, and smart agriculture. Several IoT applications also have real-time and low-latency requirements and must rely on specific architectures. The authors refer to the paradigm that best fits the selected IoT scenario as "Big Stream" because it considers real-time constraints. Moreover, the blockchain concept has drawn attention as the next-generation technology through the authentication of peers that share encryption and the generation of hash values. In addition, the blockchain can be applied in conjunction with Cloud Computing and the IoT paradigms, since it avoids the involvement of third parties in a broker-free way. In this chapter, an analysis on mechanisms that can be adopted to secure Big Stream data in a graph-based platform, thus delivering them to consumers in an efficient and secure way, and with low latency, is shown, describing all refinements required employing federation-based and blockchain paradigms.

DOI: 10.4018/978-1-5225-8295-3.ch005

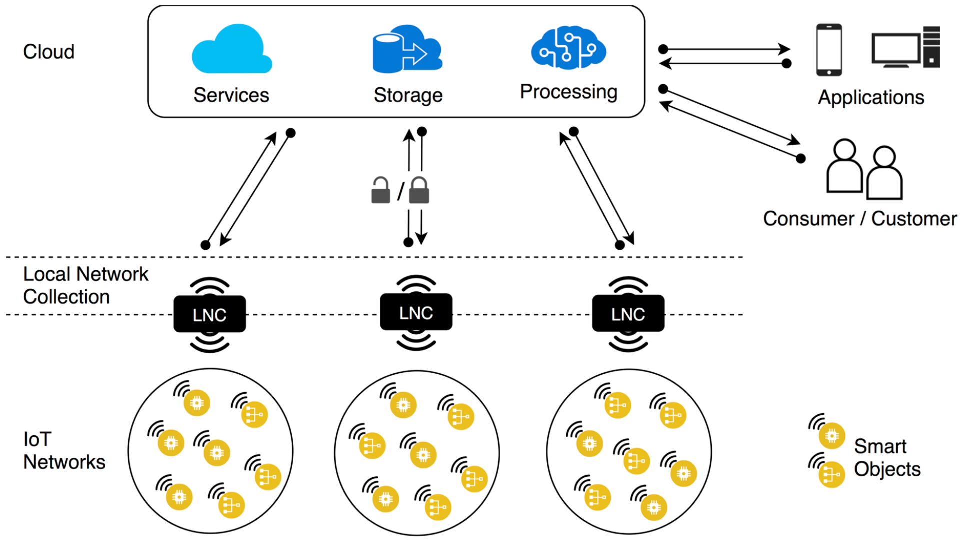
INTRODUCTION

Considering the last 15 years, the forecast of a worldwide network of pervasively deployed and connected heterogeneous networks is now a reality. The Internet of Things (IoT) is now involving billions of different devices, connected in an Internet-like structure, and has definitely changed the way in which people and things interact, in several aspects of our modern living. The actors involved in IoT scenarios have extremely heterogeneous characteristics, in terms of energy supply and consumption, processing and communication features, and availability and mobility, spanning from Smart Objects (SOs) - i.e., constrained devices equipped with actuators or sensors, smartphones, wearable devices and other personal ones - to Internet hosts and the Cloud.

In order to allow heterogeneous nodes to efficiently communicate with each other and with existing Internet actors, shared and interoperable communication mechanisms and protocols are currently being defined and standardized. The most prominent driver for interoperability in the IoT is the Internet Protocol (IP), namely its 128-bit version called IPv6. An IP-based IoT can extend and operate with all existing Internet nodes, without any additional efforts. Standardization institutions, such as the Internet Engineering Task Force (IETF) and several research projects, are contributing to the definition of mechanisms able to bring IP to SOs (e.g., the 6LoWPAN (Kim, Kaspar, & Vasseur, 2012) adaptation layer). This is motivated by the need to adapt higher-layer protocols (e.g., application-layer protocols) to constrained environments. As a result, IoT networks are expected to generate huge amounts of traffic, whose transmitted data can be subsequently processed and used to build several useful services for end users. In this way, the Cloud has become the natural collection environment for sensed data retrieved by IoT nodes, due to its cost-effectiveness, scalability, and robustness. In Figure 1, the hierarchy of different levels involved in data collection, processing and distribution in a typical IoT scenario is shown.

Sensed data are collected by SOs composing the IoT networks and sent uplink to the Cloud, which operates as a collection entity and service provider. In some cases, intermediate processing entities, identified as Local Network Collectors (LNCs), can perform some preliminary tasks on the traffic before sending data uplink, such as protocol translation, data aggregation, and temporary data storage. This layered model is extremely general and can be applied to several IoT scenarios, in which, as an example, the LNCs functionalities can be impersonated by proxies or border routers.

Figure 1. Actors involved in an IoT and Cloud platform: data generated by IoT networks are sent to the Cloud, where services are provided to consumers. An intermediate level, performs local operations, such as data collection, processing, and distribution.



Several relevant IoT application environments (e.g., industrial monitoring, automation, and transportation) aften require real-time performance guarantees or, at least, a predictable latency. Moreover, the performance requirements (e.g., in terms of data sources) may change even abruptly. The potentially large number of IoT nodes, acting as data sources and generating a high rate of incoming data, and the low-latency constraints, call for innovative Cloud architectures able to efficiently handle such massive information amount.

A possible and suitable solution is given by Big Data approaches, developed in the last few years and become popular due to the evolution of online and social/crowd services, which can address the need to process extremely large amounts of heterogeneous data for various purposes and coming from very diverse sources. However, these techniques typically focus on the data and have an intrinsic inertia (as they are based on batch processing), rather than providing real-time processing and dispatching (Zaslavsky, Perera, & Georgakopoulos, 2013; Leavitt, 2013). For this reason, Big Data approaches might not represent the right solution to manage the dynamicity of IoT scenarios with real-time processing. In order to better fit these requirements, it is possible to shift the Big Data paradigm to the "Big Stream" paradigm.

While both paradigms deal with massive amounts of data, Big Data and Big Stream paradigms differ in the following aspects.

- Nature of data sources: the Big Data paradigm deals with a wide range of different areas composed by heterogeneous data sources (e.g., health, economy, social, industrial, natural phenomena), not necessarily related to IoT. Instead, data sources dealing with the Big Stream concept are strictly related to the IoT, where heterogeneous devices generate, as a whole, a continuous and massive information stream, even sending small amounts of data.
- The meaning of the term "Big:" in Big Data it refers to "volume of data," while in Big Stream it refers to "data generation rate."
- Low-latency and/or real-time or requirements of different consumers: typically both these concepts are not taken into account by Big Data.
- Objective: Big Data focuses on information analysis, management, and storage, following the Data-Information-Knowledge model (Aamodt & Nygard, 1995); instead, the Big Stream paradigm focuses on the data flows management, being able to perform *ad-hoc* and real-time data processing, in order to speed up the incoming data stream forwarding to consumers.

Another keypoint of Big Stream-oriented systems is that they should provide smart resource allocation, efficiently reacting to changes and, thus, implementing scalable and cost-effective Cloud services. This also affects the relevance given to the data, in different processing steps, for the final consumers. For instance, while for Big Data applications the ability to store all data is important, in order to be able to perform any successive required computation, Big Stream applications might decide to perform data filtering, aggregation or pruning, in order to minimize the information latency in conveying the final processing output to consumers, with no persistence need. Eventually, as a generalization, Big Stream data flows can be delivered to a Big Data application, which can act as a final consumer performing storage operations.

For these reasons, in previous works (Belli, Cirani, Ferrari, Melegari, & Picone, 2014; Belli, et al., 2015) we have designed and implemented a graph-oriented Cloud architecture suited for IoT scenarios, composed by applications with real-time and low-latency requirements (i.e., Big Stream applications). The proposed architecture relies on the concepts of data listener and data-oriented graph processing, in this way implementing a highly configurable, scalable, and dynamic computations chain on incoming data streams, dispatching data with a push-based approach, and providing the shortest delay between the time instant in which the information is generated and the time instant in which it is consumed.

To fulfill the IoT vision, some issues must be addressed, such as the inbound and outbound security of such IoT scenarios. Securing the IoT faces off with different aspects and, thus, entails both authorization and authentication mechanisms in order to address privacy, confidentiality, and trust. Security has been identified as one of the most critical aspects that should be dealt with in several IoT applications.

As in a previous work (Belli et al., 2015), we assume that streams generated by IoT nodes in the proposed graph-oriented architecture are "open" and potentially accessible by any interested entity, which can be seen as a subscriber for an interested topic. However, this assumption can not meet the security requirements of all developers or consumers accessing the Big Stream-oriented architecture, even more so that security management has become necessary in application scenarios that require to control or filter accesses to one or more streams by subscribers. For this reason, the architecture preliminary proposed in (Belli et al., 2015; Belli, Cirani, Ferrari, Melegari, & Picone, 2014) has to be extended also taking into account security aspects. In particular, the extension introduces additional modules useful to make the proposed graph-oriented architecture able to handle both secured and "open" data streams, focusing on solutions able to decouple security roles and management purposes (e.g., OAuth protocol, an *n*-legged authorization protocol), even adopting some central authority-free models (e.g., relying on the blockchain paradigm).

The rest of this work is organized as follows. In the following, an overview of related works is presented. Next, the new concepts and modules needed to "secure" data streams are detailed, analyzing which are the main issues in Cloud-oriented architectures and how the proposed entities can intervene against these issues, in this way increasing the platform security. Then, Big Stream architecture component are shown, highlighting the differences between the unsecured platform and describing how the blockchain technology can enhance the platform. Finally, we draw our conclusions.

RELATED WORKS

Internet of Things

In (Mineraud, Mazhelis, Su, & Tarkoma, 2016), the authors present a survey on IoT platforms in order to highlight differences in term of distribution of applications and services and to evaluate the gap between current IoT solutions and expectations of users. Finally, the authors provide a list of recommendations that should be followed by future platforms to fill the identified gaps.

The work described in (Afzal, Umair, Shah, & Ahmed, 2017) focuses on the recent paradigm of Social IoT (SIoT), an emerging subset of IoT aiming at establishing social relationships among SOs which start to interact each others and with the final users. The main concept characterizing this particular paradigm is that each device in the network cooperates to provide different kind of services, trying to achieve a common goal. After describing SIoT features, the authors analyze the Operating Systems (OSs) that better fit this scenario, also proposing an OS model architecture.

At the same time, the European Union (EU), under its 8th Framework Program Horizon 2020 (H2020, 2014-2020), is supporting a significant number of IoT related projects addressing different challenge and aspects. Among them, we recall the "Aggregate Farming in the Cloud" (AFarCloud) project (European Union, 2017), which aims at providing a new distributed platform for autonomous farming based on the real time integration and cooperation of agriculture Cyber Physical Systems (CPS). The AFarCloud platform also support monitoring and decision-making solutions and has the goal of increase efficiency, productivity, animal health, and food quality. The "Interoperability of Heterogeneous IoT Platforms" (Inter-IoT) is another European project (European Union, 2016) focusing instead on the integration problem, since, according to authors, most of existing IoT systems are isolated, and based on "closed-loop" concepts. InterIoT partners propose a multi-layered approach integrating different IoT devices, networks, platforms, services and applications allowing a global continuum of data and services creating an ecosystem of interoperable IoT platforms. With this approach, companies and developers can quickly create new IoT services and devices for the ecosystem, even in absence of global IoT standards.

Security With Blockchain Technology

The concept of blockchain technology can be presented from different perspectives and can be applied in different contexts, ranging from monetary scenarios to security ones. In (Park, & Park, 2017), this newly emerging technology is discussed in both analytical and practical ways, and thus focusing on its most relevant research trends, as well as investigating the way in which the blockchain security can be adapted to Cloud Computing. In (Gaetani, et al., 2017), the authors highlight the relevant importance of the data, together with the malicious effects that data attempts of threats may have on business decisions, especially in Cloud Computing-related scenarios. To solve these issues, it is possible to adopt a federated infrastructure that may improve data integrity and safety, rather than a trusted and decentralized blockchain-based data provenance architecture, as described in (Liang et al., 2017), in which the relevance of Cloud, secure and blockchain-based data origin, and their involvements in each specific context is discussed, thus pursuing blockchain transactions characteristics for embedding the data provenance. In (Puthal, Mohanty, Nanda, & Choppali, 2017), an overview of traditional security solutions, cyber threats, and possible security models to overcome current security drawbacks are discussed. Considering the wide variety of devices used in computer networks and the role played by cybersecurity in securing and improving the performance of these systems, traditional security solutions often target to network-based cyberspace solutions, which is well-known to be sometimes an open door to attackers. Hence, these security holes sometimes happen when the communication starts before the authentication, thereby leaving a space for entering the system before authentication by an attacker. In order to improve the strength of Cloud-based architectures, it is possible to refer also to the interconnection of Edge Datacenters (EDCs) and Cloud DataCenters (CDCs) in scenarios in which IoT-oriented sensing devices are communicating together, considering, as a key performance indicator, the impact of data analysis (in terms of data storage, fusion, and processing) and the effects of applying security measures in these scenarios (Puthal, Nepal Ranjan, & Chen, 2016). In (Biswas, & Muthukkumarasamy, 2016), the context of smart cities is analyzed, proposing a blockchain-based security framework that provide a secure communication platform with IoT-oriented smart devices, in order to protect and assure the data privacy when managing physical, social, and business infrastructures, for enhancing the direct interaction and collaboration between citizens and the local government (Kshetri, 2017). Moreover, knowing that one possible problem related to the adoption of the blockchain technology in IoT-oriented scenarios is the need of an algorithm based on Proof of Work (PoW) for defining the parties involved in the mining process, the definition of a blockchain-based IoT-oriented solution may be a turning point to eliminate the PoW in the system core; this may happen modelling the system in several tiers, in which the owner of the resource impersonates the role of a miner, in charge also of handling all communication within and external to the smart home (Dorri, Kanhere, Jurdak, & Gauravaram, 2017). Finally, another interesting scenario which can be covered by both a Cloud Computing-based data processing and blockchain-based data storage and retrieval is represented by Vehicular Networks (VNs). In these scenarios, it is possible to define secure and reliable architectures allowing a better utilization of both the infrastructure and resources of Intelligent Transport Systems (ITSs) and smart cities (Sharma, Moon, & Park, 2017).

Protocols and Communication Models for IoT

It is a common assumption that, in IoT, the most prominent driver to provide interoperability is IPv6. Referring to the IP stack, at the application layer developers find a variety of possible protocols applicable to different IoT scenarios, according to specific application requirements. Among the many options, the following are relevant.

- HyperText Transfer Protocol (HTTP) is mainly used for the communication with the consumer's devices.
- Constrained Application Protocol (CoAP), defined in (Shelby, Hartke, Bormann, & Frank, 2014), is built on the top of User Datagram Protocol (UDP), follows a request/response paradigm, and is explicitly designed to work with a large number of constrained devices operating in Low power and Lossy Networks (LLNs).
- Extensible Messaging and Presence Protocol (XMPP) is based on XML, supports decentralization, security (e.g., TLS), and flexibility.
- MQ Telemetry Transport (MQTT) is a lightweight publish/subscribe protocol running on top of TCP/IP. It is an attractive choice when a small code footprint is required and when remote sensors and control devices have to communicate through low bandwidth and unreliable/intermittent channels. It is characterized by an optimized information distribution to one or more receivers, following a multicast approach. Being based on a publish/subscribe communication paradigm, it acts through a "message broker" element, responsible for dispatching messages to all topic-linked subscribers.
- Advanced Message Queuing Protocol (AMQP) is an asynchronous protocol based on the publish/subscribe model that arose from the financial industry. It can run on top of different transport protocols but it assumes an underlying reliable transport protocol such as TCP. The main advantage of this protocol is its store-and-forward feature that ensures reliability even after network disruptions.
- Constrained Session Initiation Protocol (CoSIP), described in (Cirani, Picone, & Veltri, 2013; Cirani, Davoli Picone & Veltri, 2014), is a lightweight version of the Session Initiation Protocol (SIP) aiming at allowing constrained devices to instantiate communication sessions in a standard fashion, optionally including a negotiation phase of some parameters, which will be used for all subsequent communications.

In (Karagiannis., Chatzimisios, Vazquez-Gallego, & Alonso-Zarate, 2015), the authors provide a comprehensive description and comparison of IoT protocols employed in IoT at the application layer. the paper analyzes both communications between things and between end-users application and the internet, evaluating their reliability, security, and energy consumption characteristics.

Cloud Stream and Real-Time Management

Many researchers focus their efforts on the definition of Cloud Computing-based IoT architectures, where: the Cloud is the infrastructure for data and service management; and the final consumer/user drives the use of data and infrastructure to develop new IoT applications.

As IoT development is generally complex and extremely difficult to do it from scratch, many companies offer IoT data platforms that can be employed as a starting point, as they usually combine many of the tools needed to manage the deployment of an IoT system, spanning from the device management, till data prediction and insights into one service. In the following, some examples of these commercial platforms are provided.

The Google Cloud Platform, and in particular the Google Cloud IoT Core (Google, 2018) is a fully managed service Cloud platform to connect and manage IoT devices, from a few to millions. The service can store data from connected devices and build new applications that can also be integrated with the other Big Data services provided in the Google Cloud Platform ecosystem.

Azure IoT (Microsoft, 2018) is the Microsoft collection of services and solutions designed to help developers to create end-to-end IoT applications. Solutions can be hosted both as Software as a Service (SaaS) or specialized Platform as a Service (PaaS) and can be integrated with other technologies (e.g., SAP, Oracle databases, Microsoft Dynamics). Finally, this platform supports different protocols commonly employed in IoT, like HTTP, AMQP and MQTT.

Another relevant solution is the IBM Watson IoT platform (IBM, 2018), which propose a set of tools promising to securely connect, collect and start processing IoT data quickly and easily. Watson permits to build a growing ecosystem connecting different IoT devices using standard protocols like MQTT and HTTP. It also allows developers to aggregate and transform collected data into custom structures leveraging on machine learning and cognitive APIs.

Cloud Computing, as well as others paradigms, has to deal with different kinds of sources and data amounts. One of the most important challenges in the IoT era is to be able to collect and process massive and heterogeneous data flows generated by billions of interconnected SOs.

In the realm of Cloud-related IoT user-driven architectures, many propose the concept of Fog Computing (Bonomi, Milito, Natarajan, & Zhu, 2014) as a solution for a variety of IoT services and applications that have low-latency and location awareness requirements. Fog Computing can be described as a highly virtualized platform that provides networking, processing, and storage services, acting on the edge of the constrained network of SOs, working between endpoint IoT devices and traditional Cloud Computing platforms. Thus, it can be considered as an integration or an extension of the Cloud, providing support to the endpoints to fulfill services that could comply with low-latency and real-time consumer requirements. Another solution, besides the Big Data Paradigm, is provided by the recent Big Stream approach. The term Big Stream was first introduced in (Belli, Cirani, Ferrari, Melegari, & Picone, 2014) and has been then developed in other works. In particular, in (Belli, et al., 2016; Davoli, Belli, Veltri & Ferrari, 2018) the authors faced the problem of the introduction of security functionalities on the proposed Big Stream platform. In the last few years, the research community worked on this topic, providing other solutions. In (Dastjerdi, et al., 2016), it is remarked how most of IoT solutions based on a centralized Cloud does not scale to requirements of such environment generally requiring low latency or realtime response (e.g., health monitoring and emergency applications). The delay of these solutions is mainly caused by transferring data to the Cloud to be stored, and then back to the application needing that information. For the authors, data in IoT environment can be classified into two categories:

- **Little Data or Big Stream:** Transient data that is captured constantly from IoT smart devices.
- **Big Data**: Persistent data and knowledge stored and archived in centralized Cloud storage.

IoT environments, including smart cities and infrastructures, according to authors, need both Big Stream and Big Data to achieve effective real-time analytics and decision making. For this reason, authors in (Dastjerdi et al., 2016) propose the Fog Computing paradigm as a suitable solution, discussing a reference architecture.

In (Malek et al., 2017), authors propose an holistic platform combing tools for IoT, complex event processing, and Big Data technologies. The aim of the platform is to allow processing of large-scale sensor data and to develop context-aware applications and services. The implementation of the proposed system is based on the combination of Kaa (Kaa, 2018) and Storm (Mera et al., 2014) technologies. Kaa is an open-source middleware tool for building, managing and integrating with various

IoT devices, Apache Storm instead, is a free, open-source, distributed event stream processing system that allows the processing of streams into several processing units to reach near real-time requirements. It can be integrated with different queueing and database technologies, providing mechanisms to define custom network topologies in which nodes consume and process data streams in arbitrarily and complex ways.

Security Issues in Cloud and IoT Publish/Subscribe Scenarios

In the literature, several methods and strategies to enable confidentiality in publish/subscribe IoT infrastructures are proposed. IoT systems have to avoid security threats, providing strong security foundations built on a holistic view of security for all IoT elements at all stages: from object identification to service provision; from data acquisition to stream processing. All security mechanisms must ensure: resilience to attacks; data authentication; access control; and client privacy.

In (Collina, Corazza, & Vanelli-Coralli, 2012), IoT systems are expected to bridge the physical and the "virtual" worlds, using a novel broker that supports protocols such as HTTP and MQTT, adhering to the REST paradigm and allowing developers to easily and responsively expose fundamental entities as REST resources. This broker does not address any security issues, claiming that possibles solutions could include: plain authentication; Virtual Private Networks (VPNs); Access Control Lists (ACLs); as well as OAuth, a new type of authorization which is used to grant access to personal data by third-party applications (Hardt, 2012).

In (Lagutin, Visala, Zahemszky, Burbridge, & Marias, 2010), the authors examine the roles of different actors comprising an inter-domain publish/subscribe network, along with security requirements and minimal required trust associations between entities, introducing and analyzing an architecture that secures both data and control planes. The main security goals for a publish/subscribe architecture are: (i) integrity; (ii) scalability; (iii) availability, and (iv) prevention of underived traffic. Finally, in (Lagutin, Visala, Zahemszky, Burbridge, & Marias, 2010) different actors and security mechanisms are identified. The main used mechanism is Packet Level Authentication (PLA) which, combined with cryptographic signatures and data identifiers tied to secured identifiers, creates a strong binding between data and traffic, thus preventing Denial of Service (DoS) attacks.

In (Yang, et al., 2017), the privacy problem is faced, and authors propose an Attribute-Keyword-based data Publish-Subscribe (AKPS) scheme for Cloud platforms protecting the privacy of the published data against the Cloud server and other entities which are not subscribers of the system. To protect the subscribers' interests, a new searchable encryption is proposed, to enable the subscribers to selectively receive interested data. The AKPS supports multiple publishers and multiple subscribers, while none of two publishers/subscribers share the same secret keys. To avoid bypassing an access/subscription policy checking procedure, the AKPS ties both access policy and subscription policy by two secrets: one to bundle the ciphertext and the tags together; and the other to bundle the subscription trapdoor and the pre-decryption key together.

In (Raiciu, & Rosenblum, 2006), the authors present a study of confidentiality in Content-Based Publish/Subscribe (CBPS) systems, defined as an interaction model storing the interests of subscribers in a content-based infrastructure, to guide routing of notifications to interested subjects. In (Fremantle, Aziz, Scott, & Kopecky, 2014) the use of Federated Identity and Access Management (FIAM) in IoT is analyzed, following a consumer-oriented approach, where consumers own data collected by their devices, having a control over entities who access these data. Traditional security models, based on the concept of roles with a hierarchical structure, are not applicable to IoT scenarios (because of the billions of devices involved, the impossibility to adopt a centralized model of authentication, and the necessity to support mechanisms for delegation of authority). The authors propose OAuth2 as a possible solution to achieve access management with IoT devices which support the MQTT protocol. The overall system consists of: (i) a MQTT broker, (ii) an Authorization Server supporting OAuth2, (iii) a Web Authorization tool, and (iv) a device.

The work of (Bacon, et al., 2010) tackles the problem of application security in the Cloud, aiming at incorporating end-to-end security, so that Cloud providers not only can isolate their clients from each other, but can also isolate the data generated by multiple users which access a particular service provided by the Cloud. An approach called "application-level virtualization," which consists of (i) removing from applications all the details regarding security and flow control, (ii) placing the security management logic in the Cloud infrastructure, and (iii) allowing providers to permit only the interactions that the clients specify, is proposed.

Finally, a comprehensive state-of-the-art analysis of security solutions for Cloud publish/subscribe systems is provided in (Uzunov, 2016), where a set of solutions providing concrete security architectures are reviewed "horizontally" considering several aspects, more in detail the author analysis took into account the constituent security patterns the pertinent threats addressed, thus providing a contextualized "vertical" dimension to each solution individually.

Blockchain in a Nutshell

Blockchain has drawn attention as the next-generation technology through the authentication of peers that share encryption, the generation of hash value, and virtual money (e.g., Bitcoin, Ethereum, Ripple, EOS, IOTA, Tron, Monero (Bushmaker, 2018)). In addition, the blockchain technology can be applied in conjunction with the Cloud Computing and the IoT paradigms, due to its efficiency in sharing the stored information and availability in being shared among different parties. One of its main strength points is that it involves the communication solely between peers and without the involvement of third parties, in a broker-free way. In general, the use of the blockchain technology can provide higher security levels compared to storing all data in a centralized database, since attacks on a database can be prevented more easily. Moreover, since the blockchain has an openness connotation, it can provide data transparency when applied to contexts in which data disclosure is required. Due to such strengths, the blockchain technology can be utilized in diverse areas including the financial sector and the IoT environment, and its applications are expected to rapidly expand.

The blockchain main objective is to finalize transaction records through the work of an authentication process, such as the Proof-of-Work (PoW). The blockchain corresponds to a structured list that saves data in a form similar to a distributed database allowing all members of a community to keep a sort of ledger containing all transaction data, and to update and maintain their ledgers integrity when there is a new transaction. Since the advancement of the Internet and encryption technology has made possible for all members to verify the reliability of a transaction, the single point of failure arising from the dependency on an authorized third party has been solved. In fact, as a blockchain can be considered as a public ledger for transactions, it can prevent hacking during transactions involving different entities. The distributed database associated to transactions is a data record list that continuously grows, and it is designed to disable arbitrary tampering.

In order to authenticate a new blockchain transaction, a hash value is generated by verifying the PoW process and connecting to the previous block, in turn periodically updated and reflected on the electronic transaction details to share the latest transaction detail block. This process is a reliable mechanism to provide security since the network participants save and verify the blockchain. Since the hash values stored in each peer in the block are affected by the values of the previous blocks, it is very difficult to falsify and alter the registered data. Although data alteration is possible if at least 51% of peers are hacked at the same time (meaning that the attacker must have at least 51% of the computational power of all users), this attack scenario is realistically very difficult (especially in terms of coordination activity directly orchestrated by the attacker). This is called "51% attack" and is due to the fact that although there should be only one blockchain, being a sequential connection of generated blocks, a blockchain may be intentionally divided into two different sub-branches because the two latest blocks can be generated temporarily if two different peers succeed in mining the answer for generating the block at the same time. In this case, the block that is not chosen as the latest block by the majority of peers will become meaningless. This reflects into the fact that the transactions will follow the majority of peers who have 50% or more mining capability and, if this happens, the attacker has the control on the overall blockchain and it can include falsified transactions, and this may represent an enormous problem. To solve the problem, and prevent such tampering, an intermediate verification process must be provided. Thus, in blockchain-oriented scenarios, in order to improve the security of the parties participating in the P2P network, it is possible to adopt public key-based verification mechanisms, such as the Elliptic Curve Digital Signature Algorithm (ECDSA) (Triwinarko, 2002). In this way, although the identification of an account information with an anonymous public key enables one to know the entity who sent the information to another peer, it still ensures a certain anonymity, since there is no way of finding information pertaining to the original owner. Moreover, if an enhanced security measure is needed, it is possible to adopt a hash function to verify that the data block containing the transaction details have not been altered, as well as to guarantee the data integrity during a transaction. Hence, since the information on a transaction ownership is shared by a number of entities, this makes hacking more difficult than on other data models, transactions are automatically approved and recorded in the ledger, and the overall architecture is always ready to process data. Moreover, in modern blockchain architectures it is possible to outsource different functionalities out of the core of the ledger; this opens several possibilities, allowing the system to be easily implemented, connected, and expanded using open source libraries. Unfortunately, this latter point clearly introduces a warning point: since these open source scripts can be written using heterogeneous programming language with some flexibility, there can be the risk that an improperly configured transaction may damage the overall blockchain (e.g., a money transaction using an improperly configured locking script may be discarded since nobody can use it as the unlocking script cannot be generated).

Another measure that can be adopted in order to assure the security of a virtual information wallet is the multisig technique, which manage multiple signature and allow a new transaction only when there is more than one signature for the transaction. This approach can be used as a redundant security feature of the wallet, and can be enforced also adopting biometric or physical, offline, cold storage-type equipments, in this way adhering to a two-factor authentication mechanism. An example of hardware wallet is Trezor (Trezor, 2018), that stores the key in a tamper-proof storage unit connected to the processing unit through a USB connection only when needed. In this way, the tamper-proof storage unit is connected only when a transaction should be established, remaining in cold status the rest of the time. As can be easily understood, problems such as lack of usability and loss of tamper-proof storage unit also afflict the hardware wallet.

Among the different attacks related to the pure networking aspect of a blockchain-oriented architecture, those which can introduces several issues, and to which some countermeasures are needed in these scenarios, are the following: (i) Distributed Denial of Service (DDoS) attacks, which flood a target with superfluous requests, in order to overload the system, prevent the normal service provisioning and, in case of blockchain's users, prevent from receiving the regular service; (ii) bandwidth-consuming attacks, which exceed the bandwidth available for the overall network; (iii) Packet-per-Second (PPS)-consuming attacks, which cause the denial of service to other servers in the same network or an internal system failure; and (iv) HTTP-flooding attacks, which cause a denial of service to a targeted server transferring a large amount of HTTP packets to it.

As can be easily understood, sensitive user data leaks may provide several damages in all the fields in which they are employed and this is even more pronounced in Cloud Computing environments. With these premises, the blockchain technology can be seen as a convenient service that provides stronger security, with the agreement of following some basic rules on the user data management. Otherwise, this could backfire on security: as an example, if user anonymity is ensured using some blockchain methods, when a user decides to delete its electronic profile from the system (and, thus, from the blockchain), he/she has to be sure that all data will be promptly removed. If it is not the case, and his/her electronic profile is not properly deleted, the user information can be left behind, and this make the entire system vulnerable, as there is the possibility to exploit the remaining user information to guess the real user profile data. Hence, the blockchain has to provide solid measures for secure restoration in case of infringement by an attacker, verification of execution code for self-protection, and data and settings protection. Finally, the blockchain has to provide the following features: (i) privacy protection, standing for the protection that has to be assured to the peers' information participating in the transaction; (ii) anonymity, assuring that the entity involved in a transaction is not identifiable in any way; (iii) integrity, standing for the need to check if the information used in a transactions have been falsified or altered during the transaction itself; and (iv) confidentiality, meaning the need to check if a data has been leaked by unauthorized peers from the blockchain itself. These needs highlight the necessity of protecting a Cloud-based system against threats to data integrity, since it is of paramount relevance and due to the fact that normally data owners can not control some fundamental data aspects, such as physical data storage and its accesses. Unfortunately, this become more and more true even knowing that nowadays public and private organisations resort to the data outsourcing due to the local need of data storage and the burden of maintenance cost.

As highlighted before, the use of the blockchain technology to fulfill these security needs may represent an interesting choice, having to deal with its throughput, latency, and stability. In the light of what has been previously said, it is possible to classify different data impairments for each category, as detailed in the following.

The compromission of data availability prevents data to be retrieved only for a certain time period, thus providing full functionality once data are accessible again. Sabotaging confidentiality cannot be retrieved and discloses private data, thus leaving original ones still usable and available. Data tampering can thus be undetected for a long time and drive malicious operations, by altering or deleting specific data entries. Finally, the compromission of the data integrity involves that there is no way to restore the original data, since they are lost forever.

Focusing on on the use of the blockchain concept in Cloud Computing-oriented scenarios, one of its strengths may be the possibility to audit all the operations happening in fixed moment, thus enforcing the concept that Cloud auditing can only be effective if all operations on the data can be tracked reliably. This may be guaranteed on the data assuring that the origin of each of them along all the path that they follow, determining their history from their original sources till their arrival point. As can be easily understood, data provenance can help in detecting access violations, that need to be identified at a fine granularity, within the Cloud Computing infrastructures. However, data provenance represents a critical issue, since it may contain sensitive information about the original data and the data owners. Hence, it is required that a blockchain-based Cloud Computing-oriented architecture needs to secure data and also to ensure the trustworthiness of data origin. In detail, as can be easily understood, sometimes it is preferable to not directly store the data, but instead the list of modifications that have been applied to them. In this way, the insertion and validation time inside the blockchain by each participating peer is reduced, while the overall blockchain operativity is maintained and preserved.

As for other types of architectures, even in Cloud Computing-based ones, there exists several security issues at each level of the architecture itself, ranging from physical device level to application level.

At the *physical layer* the security issues may be related to: (i) sybil attack, in which an attacker forges multiple devices identities and inject corrupted data with multiple malicious data sources with the same identity (e.g., with the same IP address), thus compromising the data integrity; (ii) encryption leakage, in which an attacker exploit the knowledge of the identity of a device even if its traffic is encrypted; (iii) malicious data and fake device, in which an attacker can inject fake data or code through the usage of a malicious sensor node; (iv) node tampering, in which an attacker can physically tamper with different modes the node, stealing data, code and keys; and (v) malicious behaviour inference, in which the attacker tries to deduce the behaviour of a device even in presence of encrypted or anonymized traffic.

At the *perception layer* the security issues may include: (i) timing attack, in which an attacker analyzes the encryption algorithm trying to obtain the adopted secret key and, once it has collected all possible secret keys, it uses each key to decrypt the encrypted packets; (ii) jamming, in which an attacker forces a malicious device to broadcast radio signals on the same frequency of a fair device, overpowering its original signal; (iii) selective forwarding, in which an intruder commands its malicious nodes to forward, refuse or simply drop certain messages; (iv) replay attack, in which the attacker tries to destroy the certificate's validity replying with a false response on behalf of the destination device, in this way reaching in accessing to the trusted properties of the fair device itself; (v) sinkhole attack, in which an attacker exploits unfaithful routing information to attracts the fair devices, thus affecting the overall communication process; (vi) routing attack, in which an intruder creates routing loops, to block or resend the routing information and increase the overall transmission delay; (vii) spoofing, in which an intruder successfully reaches in creating routing loops, attracting or denying traffic, extending or shortening traffic routes, generating false error messages, partitioning the network, increasing end-to-end latency; (viii) wormhole attack, in which an adversary hijacks the incoming traffic to malicious destination nodes through a tunneling operation; (ix) HELLO flooding attack, in which an attacker broadcasts malicious data with enough transmission power to convince everyone in the network that its malicious node is the right neighbor for their routing tables entries; (x) exploit attack, in which an adversary profits by some existing vulnerabilities to mutate the normal behavior and to confuse the entities participating in the communication; and (xi) Man-In-The-Middle (MITM) attack, in which an attacker secretly introduces itself in the communication and becomes able to intercept and manage the traffic incoming from each side of the network communication.

At the *application layer* the security issues may be related to: (i) data availability, in which unauthorized accesses should be prevented and only authenticated users should be allowed to access the data (e.g., key agreement, maintaining confidentiality with private information protection, access control, data security protection); (ii) authentication, in which it is fundamental to apply effective mechanisms to authorize users to utilize the systems based on the specific privileges associated with their roles, preventing, unauthenticated access; (iii) software vulnerabilities, in which an attacker can exploit some known security holes at the software level, such as cross-site request forgery, SQL injection, and buffer overflow; and (iv) data protection and recovery, in which it is fundamental to prevent malicious attacks and to be able to safely and securely restore data after catastrophic events.

BIG STREAM CLOUD ARCHITECTURE

The IoT paradigm is foreseeing the development of our current environment towards new enriched spaces in the most areas of modern living, such as digital health, smart agriculture, pollution control, smart cities, smart homes, and smart grids. The research community and companies have been working on the IoT topic, from different perspectives, from more than fifteen years, providing an abundance of solutions to interconnect SOs for systems with different scales, technologies and objectives. In general, an IoT platform can be defined as an infrastructure allowing end users to interact with the SOs composing an IoT network and take advantage from personalized and custom services. IoT is already influencing common people activities and, depending on the specific applicative scenario, a lightweight platform can be deployed in a single home to manage connected objects such as the electrodomestics, lights, and the heating system. Considering a broader scale, a smart city can enhance its management with an IoT platform handling thousands of sensors. The outcome of this trend is the generation of a huge amount of data that should be handled, aggregated, analyzed, transformed and delivered to the final user of the platform in an effective and efficient way, through commodity services.

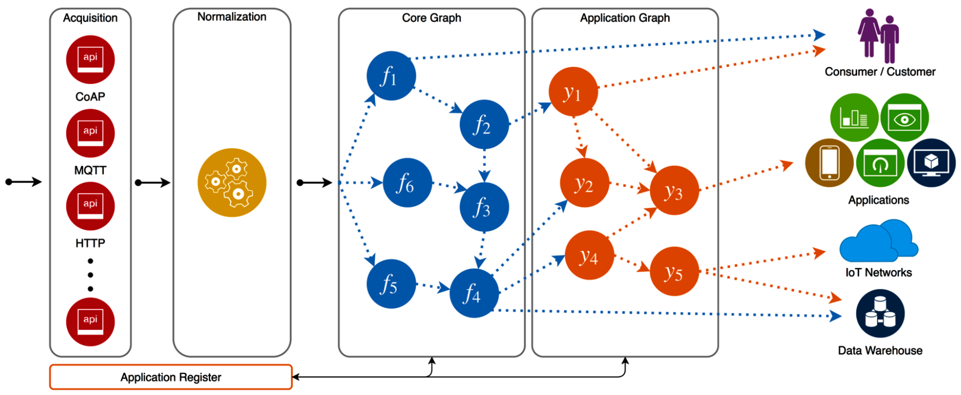
As already stated, most relevant IoT applications like alerting and monitoring systems have real-time performance or require, at least, a predictable and consistent latency. Moreover, in the IoT scenario, a large number of data sources globally generate data at a very high rate even though a single IoT service generates a limited amount of data. The low latency constraints call for an architecture able to efficiently handle such massive amount of information and to shift from the already known Big Data paradigm to the new one, denoted in literature as “Big Stream.” The term “Big Stream” was first presented in (Belli, Cirani, Ferrari, Melegari, & Picone, 2014), where a Cloud-oriented graph-based architecture, explicitly designed to fit Big Stream IoT scenarios, is proposed and implemented. The architecture aims at minimizing the delay between the instant of raw data generation by deployed sensors and the instant at which properly processed information is provided to a final consumer (e.g the user of a smartphone application, a data warehouse or a server). In order to achieve a better performance and to avoid whortless wait, the proposed Big Stream architecture adheres to the publish/subscribe model and is based on the concept of "listener." More in detail, the fundamental components of the graph-based Cloud architecture are the following.

- **Nodes**: Are processing units which treat incoming data streams to generate a new one. The task performed by each unit is self contained and is not directly linked to a specific task. A graph node can be a listener of one or more input streams and a publisher of a new stream for other nodes.
- **Edges**: Are streams linking together various nodes, allowing to build a complex behavior combining different processing operations.

Nodes in the Graph are organized in concentric layers, with an increasing degree of complexity. This means that data streams, started from IoT networks, flow in the graph and are processed by nodes. Streams, generated from nodes in a generic layer of the graph, can be used by nodes in higher-degree layers, to perform more complex processing operations, generating new streams which can be used in higher layers, and so on. The graph architecture should be exploited on the Cloud to improve the system's scalability and to manage the workload with a huge quantity of data sources and processing nodes. Therefore, this architecture can be considered as a common open platform in which data streams are shared for developers interested in building applications based on data originated by IoT networks, with real-time constraints and low overhead. Developers authenticated on the platform can thus customize paths in the graph through the definition and deployment and combination of new processing nodes. It is important to observe that, by accessing the Big Stream architecture, each developer can operate on data streams coming from IoT networks which he/she does not own.

In the proposed implementation (Belli, et al., 2015), shown in Figure 2, the system is based on three main modules: (i) the Acquisition and Normalization modules; (ii) the Graph Framework module; (iii) the Application Register module. As the architecture is based on a queue-communication paradigm, the implementation adopts an instance of RabbitMQ queue server as publish/subscribe engine. The first Acquisition step handles raw data coming from external SOs through different application-layer protocols. The implementation proposed in (Belli, et al., 2015) supports different protocols, namely: (i) HTTP, through Nginx and PHP; (ii) CoAP, by means of the mjCoAP (Cirani, Picone, & Veltri, 2015) library; and (iii) MQTT, with the ActiveMQ library. Then, the Normalization block consists of a set of Java processes, which work on incoming data from the Acquisition dedicated queues and perform some preliminary optimization operations, in order to structure data into

Figure 2. Generic structure of the Big Stream architecture with detail of composing modules



a common and easily manageable JSON format. The Graph Framework module represents the processing block, in which nodes of each level are implemented with Java processes connected through queues. To ensure proper data propagation and avoid loops, layers in the Graph Framework are implemented through dedicated RabbitMQ Exchanges, each of one connected, through one-way links, with the Exchange of the consecutive layer. The data flow starts from inner levels and goes out until it reaches the last layer's Exchange, responsible to manage notifications to external entities that are interested in final processed data. These external entities are heterogenous, and can be, in example, browsers, Data Warehouse, smart entities, or external Cloud Graph processes.

Finally, the Application Register (AR) module is the management Java process that coordinates both the interactions between Graph nodes and with external services. Moreover, the Application Register module has the fundamental responsibility to maintain all the information about the current status of the platform and the graph structure. More precisely, the Application Register module performs the following operations:

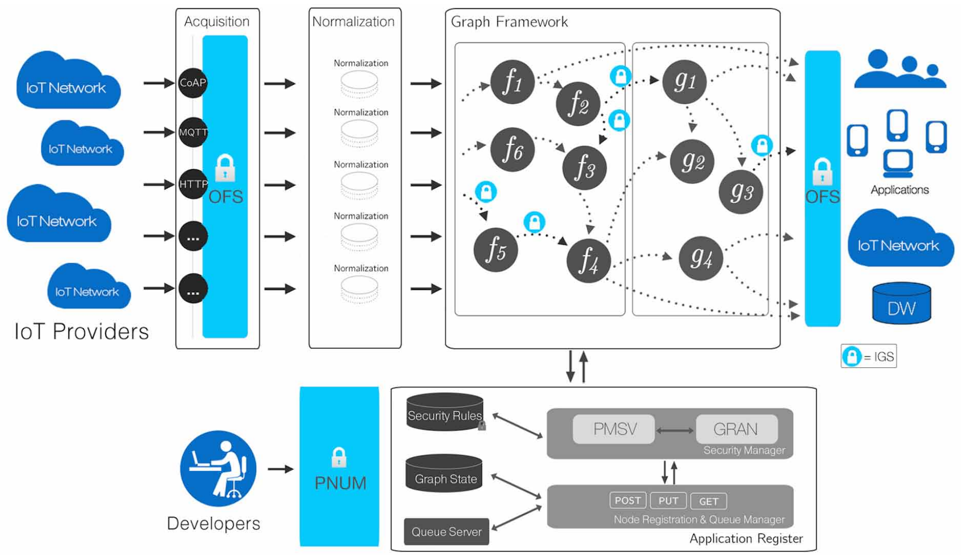
- manage the attachment of new nodes or consumers, interested in some of the streams provided by the platform;
- manage the detachment of nodes from the Graph, when they are no longer interested in receiving flows, and, possibly, re-attach them;
- handle nodes that are publishers of new streams;
- maintain information regarding topics of data, in order to correctly generate the routing keys, and to compose data flows between nodes in different Graph layers.

Security in Big Stream

Addressing the security problem in the graph-based Cloud system requires a general approach, owing to different needs of each specific component involved. In fact, giving external providers the possibility to upload into the graph custom nodes containing executable software code, clearly opens several security vulnerabilities in the system.

In (Belli, et al., 2016), security management in the Big Stream architecture has been introduced and implemented considering two clearly separated levels: (i) inner security, through the In-Graph Security (IGS) module, and (ii) outer security, with the Outdoor Front-end Security (OFS) module. As shown in Figure 3, the IGS module manages security in streams paths, defining a set of rules and indicating which actors are authorized to send/receive data to/from each single processing node. In other words, the IGS module allows developers to define paths in which some segments can be "secured" and some others can be "public" according to the kind of data treated and the specific application. To accomplish this task, the IGS module works in combination with the AR module, which is composed by the following components (shown in Figure 3):

Figure 3. Main building blocks of the proposed listener-based graph architecture with security models. Information streams between nodes can be "open" or "secured."



- the Graph State Database (GSDB) and the Node Registration and Queue Manager (NRQM) modules, which cooperate to manage the authorization policies adopted by the graph nodes;
- the Policy Manager and Storage Validator (PMSV) module;
- the Graph Rule Authorization Notifier (GRAN) module;
- the Persistent Security Storage Container (PSSC) module.

The other module related to security, namely, the OFS module, operates at the edge of the platform, after receiving input data from IoT networks, as well as before pushing out streams to final consumers. OFS has a crucial role, since it authorizes the interactions between the external entities and the frontier of the Cloud platform.

These basic security features in the core part of the platform have been further improved in (Davoli, Belli, Veltri, & Ferrari, 2018), where the concept of federated authentication is introduced. With this new feature, a federation of trusted entities can access to the Big Stream platform, through a federated access paradigm (namely Shibboleth) and to share their own nodes with the community. The adoption of a federation-based access paradigm allows to get rid of the need to store and maintain an account for each developer, reducing system vulnerabilities and making the architecture more scalable and accessible. Federated authentication is also adopted for the output stage of the Big Stream platform, in order to authenticate a final consumer requesting one or more streams coming from the higher layer nodes. The Big Stream architecture is thus enriched with a new module, denoted as Traffic Handler Orchestrator & Rapid Intervention (THORIN), deployed in the graph platform and connected to the AR module.

THORIN has been introduced to monitor the traffic on the overall platform, analyzing in real-time the behavior of the different components, and cooperate with the Application Register module aiming at:

- controlling the interactions between the external providers and the acquisition front-end nodes;
- analyzing the behavior of nodes in the graph to find potentially suspicious activities carried out by processing units;
- reacting to malicious activities and protecting the other active nodes, allowing them to safely continue their operations.

To reach these goals, THORIN implements a Shibboleth instance that allows different providers (linked to a wide federation) to connect and provide their processing nodes to the Big Stream platform.

Regarding its monitoring activity, THORIN operates according to the following heuristic principles, namely: (i) running an internal Java module that aggregates and performs statistics on data flows, as well as (ii) interacting with external heuristic services and (iii) analyzing behaviors and comparing URLs used by inner processing nodes with publicly available dangerous blacklists (e.g., spam URLs, malware and ransomware URLs, darknet onion-like URLs, etc.).

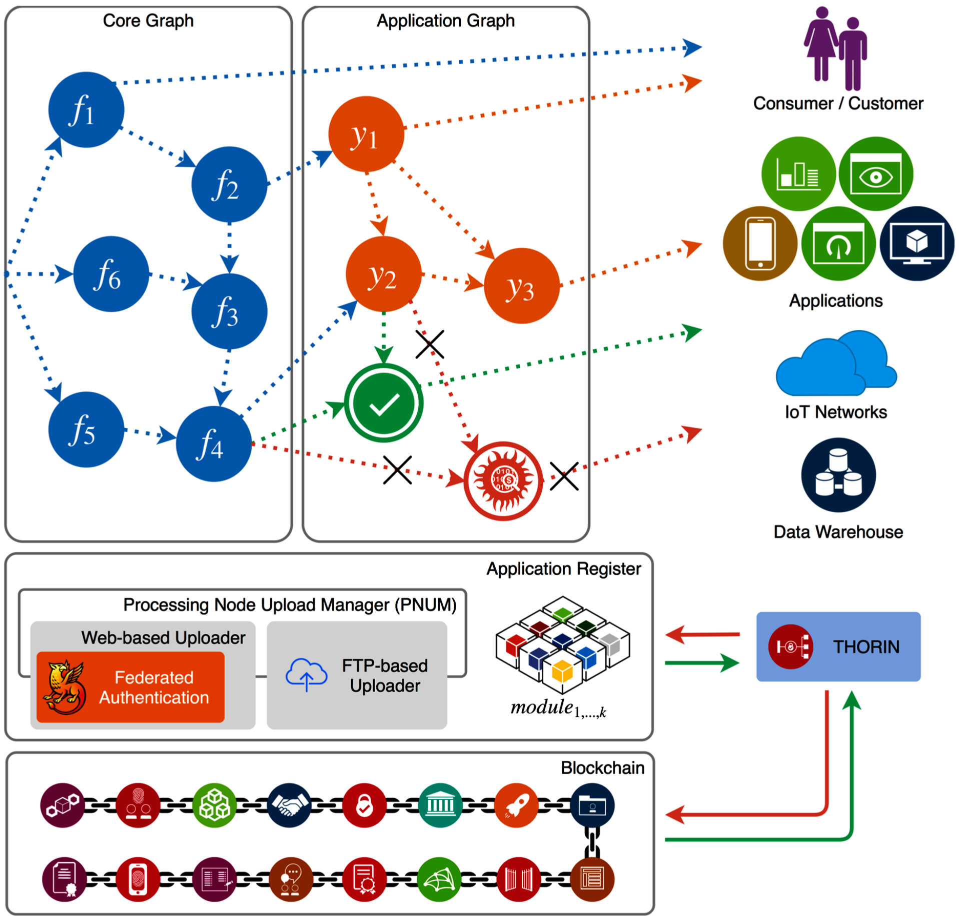
Alongside the security modules proposed previously, an interesting approach that can be applied to a Cloud-based architecture like the one oriented to Big Stream, is the adoption of the blockchain technology. Due to the fact that the Big Stream-based architecture has a graph structure, it can be suitable to involve the blockchain in the security management tier of the architecture itself. More in detail, as shown in Figure 4, the Big Stream architecture can be improved as follows. The Cloud-based architecture maintains a blockchain composed by blocks with different meanings.

When a new node request to join the platform, the federation to which it belongs to releases it a Universally Unique Identifier (UUID) that will be stored inside the new block, in turn added to the blockchain and verified with a PoW by the nodes pertaining to the blockchain itself. In this way, the masquerade identity information is persistently stored in the chain and shared among all the participants of the federation, and cannot be hijacked by anyone.

When an external developer produces a new processing unit and he/she decides to add it to the Big Stream architecture, the already described PNUM module collaborates with the blockchain and calculates a hash value for the newly provided processing unit; then, the PNUM module creates a new smart contract containing the UUID of the developer and the previously calculated hash value, together with a timestamp corresponding to the time instant in which the processing unit has been included in the Big Stream architecture. In this way, the architecture is furtherly secure, since the information on the uploader and the Java unit are shared and stored in several ledgers, assuring the absence of alterations and, at the same time, speeding up the operation of management of new nodes in the system.

The adoption of the blockchain also allows to improve the second uploading module, the one allowing the upload of new processing units through the FTP protocol: in this case, it is possible to furtherly secure the insertion operation requiring a confirmation of the current operation to the uploading developer. This can be done through the request of a One Time Password (OTP), as well as another random information, created through the usage of a hardware tamper-proof storage unit (e.g., Trezor). This approach enforces the system security and helps in tracing possible attackers trying to hijack the system essaying different OTP combinations in a certain time period.

Figure 4. Security modules, involving both THORIN module and a blockchain component, collaborating in order to counteract internal and external security threats in the proposed Big Stream platform.



The blockchain also welcomes the blocks originated by the PMSV module, which, in turn, stores the policies defined by the owner of the processing units not only inside the PSSC module, but also inside a new block composed by a hash value representing the specific policy, and a timestamp corresponding to the time instant in which the policy has been firstly defined. Hence, this allows the Big Stream architecture to trace the modifications of these policies each time that these happen, having the PMSV module in charge of creating a new block for each modification, containing, among all the information, the current modification timestamp. In this way, the blockchain allows to maintain and extract the complete modification history, being sure that these informations have not been hijacked in any way.

The blockchain also works as a shared ledger, as previously highlighted, allowing each processing unit to verify, at each time, if a particular stream can be processed in a specific way and/or if this stream can be forwarded, once processed, to a particular next-hop processing node or not; this is possible by simply conferring to the blockchain the task to verify these operations.

Moreover, each processing node may participate in a shared community control, by creating new blocks containing a log of the operation made on a data stream, the UUID of the originator processing unit and the UUID of the destination one. This has a twofold gain: on one side, it is possible to timely trace each operation occurred in the Big Stream architecture, while on the other side it is not needed to store the complete original processed stream, but only its hash value and the UUID of the Java node that processed it, in this way saving time and not exposing sensible information, thus maintaining data integrity and confidentiality.

In case the blockchain management shows signs of slowing down, it is possible to automatically balance the load on the blockchain itself, switching from one unique blockchain to several ones, each one devoted to a specific task (e.g., a blockchain devoted to the maintenance of the processing nodes, one devoted to the policies, one devoted to the processing logs, etc.).

CONCLUSION

In this paper, an analysis of the security issues in a Cloud-oriented architecture for the management of Big Stream applications in IoT scenarios has been proposed. After a brief introduction on the state of the art of both IoT and Cloud concepts, a particular attention has been devoted to security aspects and, in particular, to the outcoming blockchain technology, and on its integration in general IoT scenarios. Then, the Big Stream paradigm has been recalled, describing its main characteristics and a recently proposed Graph-based Cloud architecture targeting IoT-based networks. The implementation of each operative module of the platform (Acquisition Module, Normalization Module, Graph Framework, Application Register) has been introduced, together with a panoramic on the weight that the adoption of security measures may have on each of them, thus describing the introduction of security-related modules, in particular the IGS module, acting inside the graph, the OFS module, operating at the boundary of the architecture, and the THORIN module. Finally, an enhancement of the architecture based on the blockchain technology, which can be employed in the proposed graph architecture to improve general security and speed up the management of new nodes, has been proposed.

REFERENCES

Aamodt, A., & Nygard, M. (1995). Different roles and mutual dependencies of data, information, and knowledge — An AI perspective on their integration. *Data & Knowledge Engineering*, *16*(3), 191–222. doi:10.1016/0169-023X(95)00017-M

Afzal, B., Umair, M., Shah, G. A., & Ahmed, E. (2017). Enabling IoT platforms for social IoT applications: Vision, feature mapping, and challenges. *Future Generation Computer Systems*.

Bacon, J., Evans, D., Eyers, D. M., Migliavacca, M., Pietzuch, P., & Shand, B. (2010). Enforcing End-to-End Application Security in the Cloud. In Middleware 2010 (pp. 293-312). Springer Berlin Heidelberg. doi:10.1007/978-3-642-16955-7_15

Belli, L., Cirani, S., Davoli, L., Ferrari, G., Melegari, L., & Picone, M. (2016). Applying Security to a Big Stream Cloud Architecture for the Internet of Things. *International Journal of Distributed Systems and Technologies*, *7*(1), 37–58. doi:10.4018/IJDST.2016010103

Belli, L., Cirani, S., Davoli, L., Melegari, L., Mònton, M., & Picone, M. (2015). An Open-Source Cloud Architecture for Big Stream IoT Applications. In I. Podnar Žarko, K. Pripužić, & M. Serrano (Eds.), *Interoperability and Open-Source Solutions for the Internet of Things* (Vol. 9001, pp. 73–88). Springer International Publishing; doi:10.1007/978-3-319-16546-2_7

Belli, L., Cirani, S., Ferrari, G., Melegari, L., & Picone, M. (2014). A Graph-Based Cloud Architecture for Big Stream Real-Time Applications in the Internet of Things. In *Advances in Service-Oriented and Cloud Computing* (Vol. 508, pp. 91–105). Springer International Publishing. doi:10.1007/978-3-319-14886-1_10

Biswas, K., & Muthukkumarasamy, V. (2016, December). Securing Smart Cities Using Blockchain Technology. In *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on* (pp. 1392-1393). IEEE. 10.1109/HPCC-SmartCity-DSS.2016.0198

Bonomi, F., Milito, R., Natarajan, P., & Zhu, J. (2014). Fog Computing: A Platform for Internet of Things and Analytics. In Big Data and Internet of Things: A Roadmap for Smart Environments (pp. 169-186). Springer International Publishing. doi:10.1007/978-3-319-05029-4_7

Bushmaker, J. (2018). *Cryptocurrencies*. Retrieved from https://www.investinblockchain.com/top-cryptocurrencies/

Cirani, S., Davoli, L., Picone, M., & Veltri, L. (2014, Jul). Performance Evaluation of a SIP-based Constrained Peer-to-Peer Overlay. In *2014 IEEE International Conference on High Performance Computing Simulation* (pp. 432-435). IEEE. 10.1109/HPCSim.2014.6903717

Cirani, S., Picone, M., & Veltri, L. (2013). CoSIP: A Constrained Session Initiation Protocol for the Internet of Things. In *Advances in Service-Oriented and Cloud Computing* (Vol. 393, pp. 13–24). Springer Berlin Heidelberg. doi:10.1007/978-3-642-45364-9_2

Cirani, S., Picone, M., & Veltri, L. (2015). mjCoAP: An Open-Source Lightweight Java CoAP Library for Internet of Things Applications. In Interoperability and Open-Source Solutions for the Internet of Things. Springer. Doi:10.1007/978-3-319-16546-2_10

Collina, M., Corazza, G. E., & Vanelli-Coralli, A. (2012). Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST. In *2012 IEEE 23rd International Symposium on Personal Indoor and Mobile Radio Communications* (pp. 36-41). IEEE. DOI: 10.1109/PIMRC.2012.6362813

Dastjerdi, A. V., Gupta, H., Calheiros, R. N., Ghosh, S. K., & Buyya, R. (2016). Fog computing: Principles, architectures, and applications. In Internet of Things (pp. 61-75). Academic Press.

Davoli, L., Belli, L., Veltri, L., & Ferrari, G. (2018). THORIN: An Efficient Module for Federated Access and Threat Mitigation in Big Stream Cloud Architectures. *IEEE Cloud Computing*, *5*(1), 38–48. doi:10.1109/MCC.2018.011791713

Dorri, A., Kanhere, S. S., Jurdak, R., & Gauravaram, P. (2017, March). Blockchain for IoT Security and Privacy: The Case Study of a Smart Home. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on* (pp. 618-623). IEEE.

European Union. (2016). *Interoperability of Heterogeneous IoT Platforms (INTER-IoT).* Retrieved from https://cordis.europa.eu/project/rcn/199587_en.html

European Union. (2017). *Aggregate Farming in the Cloud (AFarCloud).* Retrieved from https://cordis.europa.eu/project/rcn/216117_en.html

Fremantle, P., Aziz, B., Scott, P., & Kopecky, J. (2014, Sep). Federated Identity and Access Management for the Internet of Things. *3rd International Workshop on the Secure IoT*. 10.1109/SIoT.2014.8

Gaetani, E., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., & Sassone, V. (2017). *Blockchain-based database to ensure data integrity in cloud computing environments*. Academic Press.

Google. (2018). *Google Cloud IoT Core*. Retrieved from: https://cloud.google.com/iot/docs/

Hardt, D. (2012). *RFC 6749: The OAuth 2.0 Authorization Framework*. Retrieved from http://tools.ietf.org/html/rfc6749

IBM. (2018). *IBM Watson IoT*. Retrieved from https://www.ibm.com/internet-of-things

Kaa. (2018). Retrieved from https://www.kaaproject.org/

Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F., & Alonso-Zarate, J. (2015). A survey on application layer protocols for the internet of things. *Transaction on IoT and Cloud Computing*, *3*(1), 11–17.

Kim, E., Kaspar, D., & Vasseur, J. (2012, Apr). *Design and application spaces for ipv6 over low-power wireless personal area networks (6LoWPANs)* (No. 6568). RFC 6568 (Informational). IETF. Retrieved from http://www.ietf.org/rfc/rfc6568

Kshetri, N. (2017). Can blockchain strengthen the internet of things? *IT Professional*, *19*(4), 68–72. doi:10.1109/MITP.2017.3051335

Lagutin, D., Visala, K., Zahemszky, A., Burbridge, T., & Marias, G. F. Roles and security in a publish/subscribe network architecture. In *2010 IEEE Symposium on Computers and Communications* (pp. 68-74). IEEE. 10.1109/ISCC.2010.5546746

Leavitt, N. (2013). Storage challenge: Where will all that Big Data go? *Computer*, *46*(9), 22–25. doi:10.1109/MC.2013.326

Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., & Njilla, L. (2017, May). Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (pp. 468-477). IEEE Press. 10.1109/CCGRID.2017.8

Malek, Y. N., Kharbouch, A., El Khoukhi, H., Bakhouya, M., De Florio, V., El Ouadghiri, D., Latre, S., & Blondia, C. (2017). *On the use of IoT and Big Data Technologies for Real-time Monitoring and Data Processing*. Academic Press. DOI: . doi:10.1016/j.procs.2017.08.281

Mera Pérez, D., Batko, M., & Zezula, P. (2014). Towards Fast Multimedia Feature Extraction: Hadoop or Storm. *2014 IEEE International Symposium on Multimedia*, 106-109. 10.1109/ISM.2014.60

Microsoft. (2018). *Azure IoT: the Internet of Things (IoT) for every business*. Retrieved from https://azure.microsoft.com/en-us/overview/iot/

Mineraud, J., Mazhelis, O., Su, X., & Tarkoma, S. (2016). A gap analysis of Internet-of-Things platforms. *Computer Communications*, *89*, 5–16. doi:10.1016/j.comcom.2016.03.015

Park, J. H., & Park, J. H. (2017). Blockchain Security in Cloud Computing: Use Cases, Challenges, and Solutions. *Symmetry*, *9*(8), 164. doi:10.3390ym9080164

Puthal, D., Mohanty, S. P., Nanda, P., & Choppali, U. (2017). Building Security Perimeters to Protect Network Systems Against Cyber Threats. *IEEE Consumer Electronics Magazine*, *6*(4), 24–27. doi:10.1109/MCE.2017.2714744

Puthal, D., Nepal, S., Ranjan, R., & Chen, J. (2016). Threats to networking cloud and edge datacenters in the internet of things. *IEEE Cloud Computing*, *3*(3), 64–71. doi:10.1109/MCC.2016.63

Raiciu, C., & Rosenblum, D. S. (2006). *Enabling Confidentiality in Content-Based Publish/Subscribe Infrastructures*. Securecomm and Workshops. doi:10.1109/SECCOMW.2006.359552

Sharma, P. K., Moon, S. Y., & Park, J. H. (2017). Block-VN: A distributed blockchain based vehicular network architecture in smart City. *Journal of Information Processing Systems*, *13*(1), 84.

Shelby, Z., Hartke, K., Bormann, C., & Frank, B. (2014). *RFC 7252: The Constrained Application Protocol (CoAP)*. Internet Engineering Task Force.

Trezor. (2018). *Trezor: The safe place for your coins*. Retrieved from https://trezor.io/

Triwinarko, A. (2002). *Elliptic Curve Digital Signature Algorithm (ECDSA)*. Makalah TA, Departemen Teknik Informatika ITB.

Uzunov, A. V. (2016). A survey of security solutions for distributed publish/subscribe systems. *Computers & Security*, *61*, 94–129. doi:10.1016/j.cose.2016.04.008

Yang, K., Zhang, K., Jia, X., Hasan, M. A., & Shen, X. S. (2017). Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms. *Information Sciences*, *387*, 116–131. doi:10.1016/j.ins.2016.09.020

Zaslavsky, A., Perera, C., & Georgakopoulos, D. (2013). *Sensing as a Service and Big Data*. Retrieved from http://arxiv.org/abs/1301.0159