

IaaS-Aided Access Control for Information-Centric IoT

Giovanna Carofiglio*, Alberto Compagno*, Mauro Conti[†], Fabio De Gaspari*[‡] and Luca Muscariello*

*Cisco Systems, Paris, France - Email: first.last@cisco.com

[†]University of Padua, Padua, Italy - Email: conti@math.unipd.it

[‡]Sapienza University of Rome, Rome, Italy - Email: degaspari@di.uniroma1.it

Abstract—Information-Centric Networking (ICN) is a novel network architecture designed to replace the current IP-based architecture. In ICN, contents are identified based on unique names and can be cached in-network in order to immediately serve subsequent requests. These characteristics are particularly important in the context of the Internet of Things (IoT), where they allow to heavily reduce loads on the *things*, as well as increase content availability and decrease energy consumption.

In this paper we address the open problem of providing access control for IoT deployments in ICN. Our protocol takes advantage of the reliance of IoT on Infrastructure-as-a-Service and provides confidentiality, traitor tracing and revocation of an unbound number of users. At the same time, our protocol preserves both network caching and location-independent content retrieval. To the best of our knowledge, our protocol is the first access control protocol for ICN providing all these advantages. We provide an extensive analysis of the security properties of our protocol, as well as a thorough experimental evaluation. Our evaluation shows that our protocol does not pose significant overhead with respect to published content size nor computational time required. Considering a popular IoT camera, our evaluation shows that our protocol introduces 0.27% size overhead and 150ms of computational overhead for a 7 seconds stream.

I. INTRODUCTION

The Internet of Things (IoT) is widely regarded as the next step in the evolution of the Internet. However, IoT is also a cause for concern. Supporting IoT entails a considerable shift from today's Internet architecture, requiring ad-hoc solutions to the problems caused by the introduction of low-powered devices in an end-to-end based network. Moreover, widespread IoT deployment is contingent on finding ways to manage the huge amounts of data the devices will generate. Given the resource-constrained nature of the IoT devices themselves, processing and storing is generally offloaded to external platforms, as attested by the rise of new paradigms such as Edge/Fog computing [1] and the continuous thriving of Cloud computing [2]. However, offloading implies transferring the data to external platforms, which in the IoT scenario entails prohibitive bandwidth requirements. Moreover, the problem of bandwidth usage is further exacerbated in scenarios where there are multiple consumers, such as in the case of a public or semi-public IoT deployment offering live data to several users, alongside long-term data stored in the cloud.

This work is partially supported by the EU TagItSmart! Project (agreement H2020-ICT30-2015-688061), and the grant n. 2017-166478 (3696) from Cisco University Research Program Fund and Silicon Valley Community Foundation.

A natural fit to alleviate the problems of IoT integration is Information-Centric Networking (ICN) [3]. ICN is an alternative network architecture built upon *name-based* communication. In ICN all content is uniquely named: consumers request a content based on its name, which is used by ICN-enabled routers to automatically localise and return the content. When serving a request, ICN routers can opportunistically store the requested content in their local cache. Caching allows ICN routers to satisfy further requests for the same content directly, without the need to contact the producer again.

In-network caching and name-based content retrieval provide several advantages, especially in the context of IoT. First, ICN allows to reduce the load on the IoT devices due to network caching [4]. For an IoT deployment, reduced number of requests translates into lower energy consumption and longer lifespan. Secondly, ICN reduces bandwidth requirements and provides a natural mechanism to purge obsolete content from the network: the more a content is requested, the more nodes will eventually cache it, reducing the upstream bandwidth required to serve requests. Thirdly, ICN caches increase the availability of content in the network in case the producer is unreachable or overloaded, which can easily happen when dealing with resource-constrained devices. Finally, the problem of addressing the IoT devices, another major hurdle of host-to-host communication, is naturally eliminated in ICN.

From the point of view of security ICN introduces considerable challenges, particularly with regard to access control. Due to the distributed caching of ICN, a producer cannot directly control how a content is distributed in the network, nor who can or cannot retrieve it. Several encryption-based access control solutions have been proposed to tackle this problem, but they all suffer from one or more drawbacks. The first major problem relates to users' access revocation. Once a user has access to the decryption key for a protected content, it is challenging to devise a revocation system for an unbound number of users that is both effective and efficient. Secondly, it must be possible to identify users who disclose their decryption keys in order to take appropriate actions, which requires traitor tracing capabilities. Finally, these problems are exacerbated by the requirement of preserving efficient network caching. To the best of our knowledge, no previous proposal solves all the above-mentioned issues. Moreover, proposals that solve most of these issues tend to introduce high overheads.

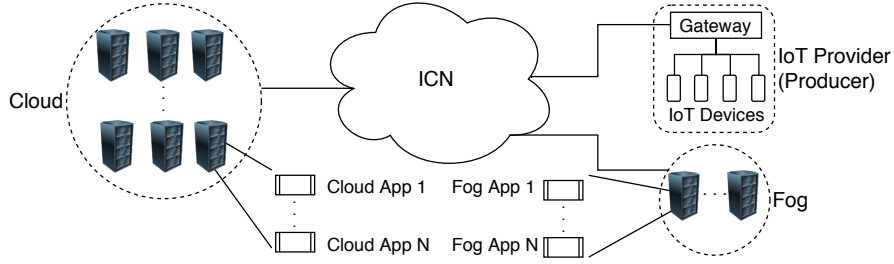


Fig. 1. System Model: IoT provider (IoT devices and gateway), IaaS provider (Cloud and Fog) and users (applications running in IaaS nodes). All actors are connected through an ICN network.

Contribution. Motivated by the advantages provided by ICN in the context of IoT, we propose a fine-grained access control protocol for IoT based on Infrastructure as a Service (IaaS)/Platform as a Service (PaaS). Our protocol takes advantage of the reliance of IoT deployments on IaaS infrastructure in order to preserve all desirable properties of ICN, while at the same time providing revocation and traitor-tracing capabilities. Moreover, our protocol can be easily integrated in existing deployments transparently with respect to users' applications. We base our construction on a combination of Ciphertext-Policy attribute-based encryption (CP-ABE) and layered encryption, which allows our protocol to scale to any number of users without significant overheads. Finally we note that, with some concessions detailed in Section VII-E, our construction can also be used as a general access control solution for ICN.

II. PRELIMINARIES

A. Information-Centric Networking

ICN is a novel network architecture proposed to address the shortcomings of the current IP based architecture. In ICN each content is uniquely identified with a *name*. Users retrieve content by issuing *interest* packets that carry the unique content name. Routing in ICN is entirely based on content names, rather than on the two particular endpoints of a communication. ICN-enabled routers can store contents in their local cache, allowing for immediate satisfaction of subsequent requests for the same content. These two characteristics allow for location-independent content retrieval: it is irrelevant who satisfies an interest. The only concern is the authenticity of the content, which is guaranteed by means of signatures on the content packets by the producer.

ICN architectures provide several advantages, in particular with regard to bandwidth efficiency and producer load. Indeed, network caching allows both to reduce the overall network bandwidth required to satisfy a request, as well as to reduce the total number of requests received by the producer. Moreover, ICN network caching provides increases content availability, since it allows retrieval of content even in case a producer is offline.

B. Ciphertext-Policy Attribute-Based Encryption

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [5] is a cryptographic primitive which allows to associate

an access structure to a ciphertext, and only decryption keys which satisfy the access structure allow for the decryption of the content. Specifically, upon generation, a CP-ABE decryption key is associated with a number of *attributes* from an attribute universe $\mathcal{U}$. During encryption, a plaintext is associated with an access structure $\mathbb{A}$ over the attributes in $\mathcal{U}$. $\mathbb{A}$ defines the *authorised sets* of attributes $U_j \subset \mathcal{U}$: the sets of attributes a CP-ABE key must have to successfully decrypt the ciphertext.

Formally, a CP-ABE scheme is composed of four algorithms:

Setup: Takes as input the security parameter and outputs the master key MK and the public parameters PP .

Key Generation(MK, S): Takes as input the master key MK and the set of attributes S describing the key to be generated. It outputs the key SK , associated with attributes S .

Encrypt($PP, m, \mathbb{A}$): Takes as input the public parameters PP , the plaintext message m and the access structure $\mathbb{A}$ over the attribute universe. Returns the ciphertext c , encrypted such that only users whose attributes satisfy $\mathbb{A}$ can decrypt the message.

Decrypt(PP, c, SK): Takes as input the public parameters PP , the ciphertext c and the private ABE key SK . Returns the plaintext m if the set of attributes associated with SK satisfy the access structure $\mathbb{A}$ associated with the ciphertext c , or $\perp$ otherwise.

A *Traceable* Attribute-Based Encryption scheme (TABE) [6] provides, in addition to the properties of classical ABE, the ability to trace the owner of a given secret ABE key. A TABE scheme is formally defined by a tuple of five algorithms: the four algorithms described above, where Key Generation takes an additional ID_u parameter (representing the ID of the user), plus a fifth *Trace* algorithm.

Trace(SK): Takes as input the secret key SK of user u . Returns the identity of the user ID_u , or $\perp$ if the key cannot be traced.

III. MODEL, ASSUMPTIONS AND GOALS

In this section we present our system model, the security model and assumptions underlying our work, and the desirable properties our access control protocol should satisfy.

A. System Model

We model our system with three separate actors, connected by an ICN network as illustrated in Figure 1: IoT provider (or *Producer*), IaaS/PaaS provider (from here on, *IaaS*) and users. The IoT provider owns and manages a number of IoT deployments, generating data for subscribed users. The IoT deployments require IaaS/PaaS platforms to collect and process the data generated by the things. As other works in the literature suggest [7], [8], [9], we assume the IoT devices are securely connected to the network through a gateway, equipped with moderate processing power. The IaaS provider offers to third parties a platform to run applications and the means to access the data generated by the IoT provider. For instance, IaaS includes Fog providers who deploy and manage a set of Fog nodes, and Cloud providers. Users deploy apps in the IaaS and access data generated by the IoT devices based on a subscription with the IoT provider. In this setting, we propose an access control protocol aimed at providing confidentiality for the IoT data with respect to the IaaS provider, the network, and unauthorised users. Note that with some minor modifications (see Section VII-C) our protocol can be adapted to a scenario where no gateway is present.

B. Security Model

We assume there is no trust between the IoT provider and any other actor in the system. As in most related works [10], [11], [12], we model the IaaS provider as *honest-but-curious*. Users can be malicious and can collude to access the IoT data. We assume that IaaS and authorised users do not collude, and that the IaaS cannot access data from running applications. To this end, techniques like [13], [14] can be employed, which provide confidentiality and remote attestation by building on top of Trusted Platform Module (TPM) [15]. We place no restrictions on collusion between IaaS and unauthorised users. Finally we assume that, for content of particular interest, the IaaS may collude with revoked users.

C. Desirable Properties

An ideal ICN access control protocol should provide a number of properties:

- **Access Control.** Only authorised users can access protected data.
- **Revocation.** It must be possible to revoke access to previously authorised users.
- **Traceability.** To provide accountability, it must be possible to link a leaked access token to its owner.
- **Caching.** All authorised users must be able to access any of the copies of the content cached in the ICN network.
- **Location-Independent Routing.** It does not matter where a request from an authorised user is satisfied.
- **Offline Access Control.** The access control mechanism must not require the producer to be online to function.

In particular, the requirement of *Traceability* rules out naïve use of hybrid encryption. Indeed, this would allow a malicious authorised user to leak the symmetric key to an unauthorised user, without possibility of identifying the traitor.

IV. THE ACCESS CONTROL PROTOCOL

In this section we propose an access control protocol which satisfies all the properties defined in Section III-C. In particular, in Section IV-A we discuss the main idea behind our protocol, Section IV-B presents the details of our construction, in Section IV-C we analyse the traceability property of our protocol and in Section IV-D we discuss revocation.

A. Main Idea

The main idea behind our protocol is to exploit the reliance of IoT on IaaS/PaaS platforms in order to split the information required to access protected contents. In particular, we separate decryption information between the IaaS platform and the users, so that no single party alone has enough information to decrypt the content. Moreover, we further use the IaaS platform as a kind of trusted platform module, in order to securely implement hybrid encryption and to enable revocation. Our construction employs CP-ABE to encrypt a symmetric key, which in turn is used to encrypt the content to be published, providing fine-grained access policy enforcement. In order to avoid the pitfalls of hybrid encryption, we augment this basic construction by means of layered encryption based on a secret shared between the producer and the IaaS provider to further encapsulate the encrypted content. The use of layered encryption on the content means that, even if a malicious user leaks the symmetric key, this alone is not enough to decrypt the content. Figure 2 provides a high level, simplified view of our construction.

B. Notation, Construction and Correctness

In this section we provide the details of our construction. The notation used is presented in Table I. The access control protocol is executed between three entities: (1) the producer P (IoT provider); (2) the IaaS provider $IaaS$; (3) the user $u_i \in U$. It is important to note that all content publication operations are executed by the gateway of the IoT deployment, not by the IoT devices themselves. A sequence diagram summarising the interaction between the three actors is illustrated in Figure 3.

Bootstrap The Producer P generates the system master ABE key and public parameters: $MK, PP \leftarrow Setup_{ABE}(\lambda)$. The Producer also generates the symmetric key K_{IaaS} , and shares it with the IaaS provider.

User Registration The producer P generates the private ABE key for user u_i based on the user's attributes γ_{u_i} and the special attribute γ_K : $SK_{u_i} \leftarrow KeyGen_{ABE}(MK, \gamma_{u_i}, \gamma_K)$ and gives it to the user together with PP . γ_K is a special attribute required to decrypt the symmetric key K_m , which we explain hereafter.

Content Publication Given a plaintext content m , P generates the symmetric encryption key $K_m \leftarrow \mathcal{K}$ and encrypts the content: $c \leftarrow Enc(K_m, m)$. P also generates the second layer key r_m . The producer encrypts r_m using ABE with access structure τ_m , $r'_m \leftarrow Enc_{ABE}(PP, r_m, \tau_m)$, and K_m with access structure τ_{K_m} , $K'_m \leftarrow Enc_{ABE}(PP, K_m, \tau_{K_m})$.

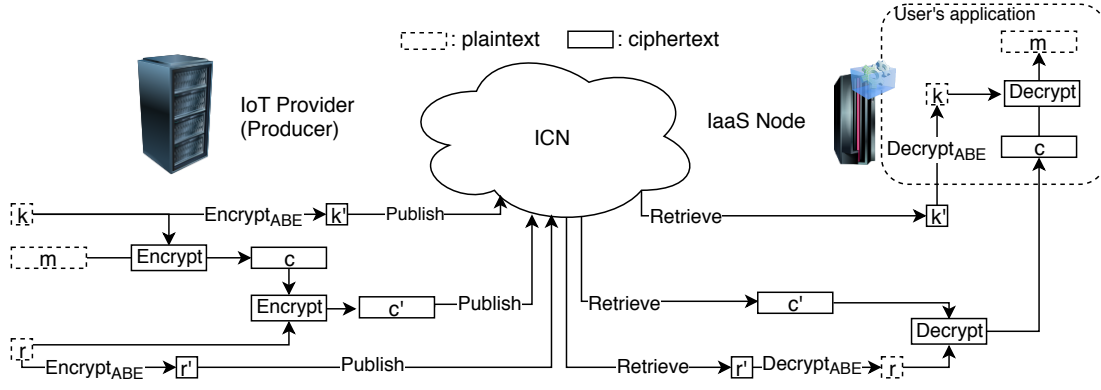


Fig. 2. High-level, simplified view of our protocol. The Producer publishes the encrypted content in the network, along with the two encapsulated encryption keys r and K .

TABLE I
TABLE OF NOTATION

P	Producer (IoT provider).
$u_i \in U$	User u_i from the group of users U .
m	Plaintext content.
Enc, Dec	Symmetric encryption/decryption.
Enc_{ABE}, Dec_{ABE}	CP-ABE encryption/decryption.
SK_P, PK_P	Private and public key of the Producer.
MK, PP	Master and public CP-ABE keys.
K_{IaaS}	Symmetric key shared between P and IaaS
K_m	First layer symmetric key for content m
SK_{u_i}	Private CP-ABE key of user $u_i \in U$.
r_m	Second layer symmetric key for m .
ω_{r_m}	Expiration information of r_m .
τ_m	Access structure for content m .
τ_{K_m}	Access structure for key K_m .
$\gamma_{u_i} = \{\gamma_{1,u_i}, \dots, \gamma_{j,u_i}\}$	Attributes of user $u_i \in U$.
γ_K	Attribute required to satisfy $\tau_{K_m} \forall m$.

The access structure τ_{K_m} requires the attribute γ_K to be satisfied, and is necessary to prevent IaaS from recovering K_m . The producer further concatenates $r'_m = (r_m || \omega_{r_m})$, and encapsulates the result with K_{IaaS} , the IaaS shared key: $\rho'_m \leftarrow Enc(K_{IaaS}, \rho_m)$. Then, the producer signs the encrypted ρ'_m . The use of layered encryption on r_m ensures that no user can recover it without interacting with IaaS. We explain the role of the expiration information ω_{r_m} when we discuss revocation in Section IV-D. Finally, the producer computes $c' = Enc(r_m, c)$ and publishes c', K'_m and ρ'_m in the ICN network.

Content Retrieval The application of an authorised user u_i who wants to access a protected content m , retrieves c', K'_m and ρ'_m from the ICN network. If the attributes of SK_{u_i} satisfy the access structure τ_{K_m} , the application can recover $K_m = Dec_{ABE}(SK_{u_i}, K'_m)$. The application then forwards a request to the local IaaS node to compute c on his behalf. From SK_{u_i} , the application generates a *delegate key* SK_{del,u_i} which includes the subset of attributes $\{\gamma_{u_i} | \gamma_{u_i} \in SK_{u_i}\} \setminus \gamma_K$, and passes it to the IaaS node along with ρ'_m . It is worth noting that the IaaS cannot use SK_{del,u_i} to decrypt K'_m , since it does not have the attribute γ_K . The IaaS node verifies the signature on ρ'_m , recovers $\rho_m = Dec(K_{IaaS}, \rho'_m)$ and validates the

expiration information ω_{r_m} . Upon validation, the IaaS node uses SK_{del,u_i} to recover $r_m = Dec_{ABE}(SK_{u_i}, r'_m)$, which will work *iff* γ_{u_i} satisfies τ_m (i.e., user u_i is authorised). The IaaS node returns to the user's application $c = Dec(r_m, c')$ if Dec_{ABE} was successful, and $\perp$ otherwise. Finally, the user's application recovers the plaintext $m = Dec(K_m, c)$.

C. Traceability

Traceability is a challenging issue in the context of CP-ABE, particularly for the case of black-box traceability [16]. Such schemes generally impose high overhead with respect to both the size of ciphertext and keys, and the computational complexity of the construction, reducing the practical use of the scheme. In our design we integrate traceability, while at the same time maintaining a low overhead. In order to achieve this, we designed our scheme to provide full traceability based on the weaker white-box traceability property, which is considerably more lightweight than black-box schemes. In our construction we have two ABE decryption operations, one performed by the user on K'_m and one performed by the IaaS on r'_m . Since under our assumptions the IaaS node follows the protocol, only a *well-formed* [6] decryption key SK_{del,u_i} can be used to decrypt r'_m . Therefore, a malicious user who wishes to provide access to an unauthorised user must leak a well-formed version of his private key, otherwise the decryption of r'_m by the IaaS would fail. Thus, we can assume that malicious users must leak a well formed key, and therefore we can safely employ an efficient, white-box traceable CP-ABE construction [6] and still provide full traitor tracing.

D. Revocation

Our system model considers a subscription-based system: users obtain a subscription for the IoT data, valid for a time period T . As a consequence, revocation happens at predefined times, rather than at arbitrary points in time. Since the expiration date of a user's key is known a priori, it can be embedded as an attribute γ_{exp} in the key itself. Respectively, an additional condition in the access structure (τ_m and τ_{K_m}) is introduced, such that only a key with expiration attribute greater than the publication date D can satisfy the policy.

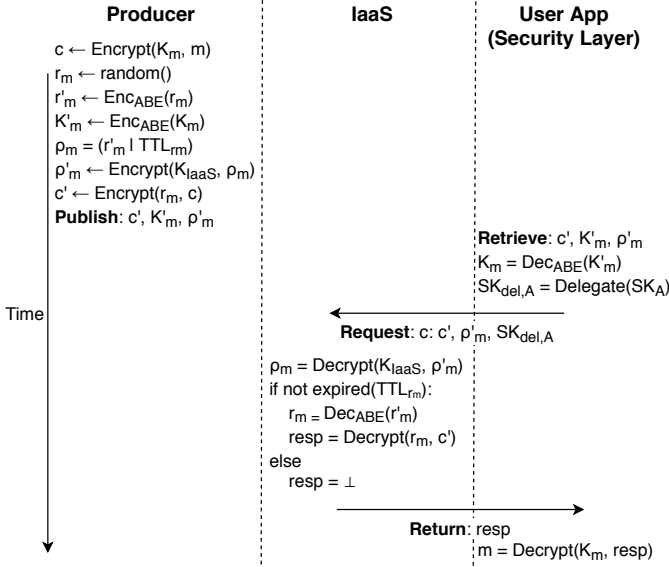


Fig. 3. Sequence diagram of the proposed protocol. The application running in the IaaS node requests the content m to the security layer, which implements our protocol.

The producer can revoke access to a user whose subscription expired by simply republishing ρ'_m with an updated date D . It is worth noting that the producer does not need to send individual updates to the users, but rather simply to publish the updated ρ'_m once in the network. Given that revocation happens at predefined times decided by the producer, that ρ'_m is small and that no interaction between the actors is required, revocation events pose no scalability issues in our protocol and their overhead is negligible. In Section VII-D we further discuss possible extensions to the revocation phase of the protocol.

1) *Freshness of ρ'_m* : Since the revocation system is based on updating the published ρ'_m , we must ensure that users cannot simply reuse an old, previously published value to retain access after revocation. Indeed, an authorised user could potentially download all currently published ρ'_m , K_m and store them in his local storage. After revocation, the user could simply retrieve c' , send a request to the IaaS node with his (expired) key SK_{u_i} along with an old ρ'_m copy, and the node will return c , effectively breaking the revocation. Alternatively, since in our threat model the network could be malicious, routers could refuse to update ρ'_m , overriding the revocation. In order to prevent this attack, we concatenate the expiration information ω_{r_m} , which indicates a time-to-live value for r_m , to r'_m . Upon receiving a request from a user, the IaaS node verifies the signature on ρ'_m to authenticate the parameter received, validates ω_{r_m} , and returns c if they are valid, or $\perp$ otherwise. It is worth noting that network-based revocation mechanism based on expiration suffer from the same vulnerability, and are therefore not applicable to our system model.

2) *Impact on Caching*: The revocation procedure is designed to preserve efficient network caching. Published con-

tents are stored in the network as long as the ICN routers do not evict them from cache. Our protocol does not reduce in any way the lifespan of cached contents in the network. The only component that our protocol requires to be republished is the key ρ'_m . However, given the small size of the key, these republication events do not affect ICN caching in a meaningful way.

V. SECURITY ANALYSIS

In this section, we discuss the security property of our construction with respect to different attackers, both individual and colluding. In the following discussion, we make the distinction between *unauthorised users* and *revoked users*. We define unauthorised users as users who were never authorised to access any protected data, and revoked users as users who were previously authorised, but lost their access privileges.

A. General Security

Since K_m and r_m are both published after encapsulating them with ABE, given an unauthorised user the security of our construction relies on the security of the underlying CP-ABE scheme used. We developed our work on the CP-ABE scheme proposed in [6] (albeit any white-box traceable CP-ABE scheme can be used interchangeably in our construction), which in turn is derived from [17] and is proven selectively secure under the standard model. As a consequence, our construction is selectively secure against any (group of) unauthorised user.

B. IaaS Provider and Unauthorised Users

Under our assumptions, the IaaS provider always follows the defined protocol, but tries to obtain as much information about the protected content as possible. As in the general case, the IaaS provider has access to the public c' , K'_m and ρ'_m . Additionally, IaaS knows the symmetric key K_{IaaS} and learns the delegate key SK_{del,u_i} of the user during the content retrieval phase. Given this information, IaaS can strip the outer encapsulation layer of c' to obtain the ciphertext $c = Dec(r_m, c')$. In order to further decrypt c and recover m , IaaS needs to be able to obtain the symmetric key K_m , which is encrypted with ABE under access structure τ_{K_m} . However, since SK_{del,u_i} does not include the attribute γ_K required to satisfy τ_{K_m} , the IaaS node is unable to decrypt K'_m and therefore cannot recover the plaintext m . Furthermore, under our assumptions the IaaS provider can collude with unauthorised users. However, since unauthorised users do not possess any additional information with respect to the IaaS provider, the security considerations for this scenario are the same as those considering only IaaS.

C. Revoked Users

After revocation, a user retains his private ABE key and potentially all symmetric keys K_m for the contents he had access to during the subscription (we can assume the user stored the keys locally). With respect to new content published after the revocation, the user is in the same position as any

other unauthorised user since his private key SK_{u_i} is not valid anymore, and therefore the security in this case reduces to security in the general case. However, with respect to content published when he had an active subscription, the revoked user knows the symmetric key K_m and therefore, given c , he would be able to obtain the plaintext m . In order to recover c , the revoked user needs to strip the outer encryption layer. Since the IaaS provider rejects expired ρ'_m (see Section IV-D1) and since the revoked user's key SK_{u_i} is invalid for current ρ'_m , the revoked user is unable to obtain c . Therefore revoked users cannot obtain the plaintext for any content published either before or after their revocation.

D. Colluding IaaS and Revoked Users

The collusion between IaaS and revoked users poses a considerable challenge. First, we need to distinguish between two cases: (1) IaaS colludes with a revoked user, but still strictly follows the protocol; (2) IaaS colludes with a previously revoked user and does *not* follow the protocol. In case (1), the IaaS provider and the revoked user know the symmetric key K_m for all the content published while the user had a valid subscription (we can assume the user previously stored them). Therefore, in order to obtain m they need to recover r_m and strip the outer encapsulation layer of c' . However, since the IaaS provider is following the protocol (and therefore would reject expired ρ'_m) and the user is unable to decrypt ρ'_m (since he is revoked), they cannot obtain r_m , and hence the plaintext. In case (2), IaaS and revoked user know K_m and, since IaaS is not following the protocol, they can obtain r_m using an expired ρ'_m . At this point, the IaaS provider and the user can recover the plaintext m . However, there are few points worth noting relative to case (2):

- 1) The previously published content needs to be available in at least one ICN node's cache in order for the user to retrieve it. If the content is retrieved directly from the producer, P can re-encrypt and publish the content under a new key K_m .
- 2) The revoked user needs to store all keys K_m and all ρ'_m for each content he wishes to access after his subscription expires.
- 3) The IaaS provider must disregard the protocol and any (contractual) agreement with the producer P . If exposed, the IaaS provider will likely suffer economically, both from reputation damage and potential litigation.
- 4) Even if points (1), (2) and (3) hold true, the IaaS provider and the revoked user obtain access only to the content for which the user had access during his subscription.

Given the above, we don't believe this to be a significant issue in practice.

E. Colluding Authorised and Unauthorised Users

An authorised user u_A will always be able to provide access to restricted content to an unauthorised user u_U , either by providing u_U with his credentials or by leaking the protected content. In this case, the security goal we wish to achieve is

not confidentiality of the content, but rather *traitor tracing*, i.e. being able to identify who leaked the access credentials. Please note that, as is common in the literature [18], [19], [20], we consider the question of how the leaked key is obtained by the Producer out of the scope of the paper. With respect to traceability we aim to show that, in order to provide u_U with access to the protected content, u_A must either leak information that unambiguously identifies him, or leak as much information as the size of the content itself (at which point he can just leak the content). In our construction, an authorised user has access to the symmetric encryption key K_m and possesses a valid ABE key SK_{u_A} . The user can safely leak the symmetric key, since it would be impossible to trace it back to him given that all authorised users have access to it. However, leaking K_m is not enough to give access to an unauthorised user. In order to provide access to u_U , u_A would have to either leak his secret key, or leak c . However, leaking the secret key allows the producer to identify the malicious user (see Section IV-C) and leaking c requires as much data as the plaintext content itself.

VI. EVALUATION

In this section we present the results of our experimental evaluation with respect to the time and size overhead introduced by the cryptographic operations. All experiments are executed on a 2.5GHz quad-core i7 processor. The protocol is implemented in Charm [21], with computationally expensive routines implemented in C. ABE operations use a 160-bit elliptic curve group based on a supersingular curve with embedding degree 2, over a 512-bit finite field. We use two and three attributes *AND* access policy for τ_m and τ_{K_m} respectively. Symmetric encryption operations use 128-bit keys.

A. Published Size Overhead

We ran a number of different tests with varying size for the original unencrypted content. As illustrated in Figure 4, we can see that the size overhead introduced by our protocol becomes negligible as the size of the original content increases. Indeed, as detailed in Table II and illustrated in Figure 5, we see that the size overhead is high for very small contents, where the published blob is up to 3.37 times the original size for a content of 1KB. However as the content size increases, the size overhead ratio quickly decreases, reaching .16 for a 16KB content and falling as low as .0027 for a 1MB content. As depicted in Figure 5, this behaviour is due to the fact that the overhead introduced by ABE, which is the main contributor to the total overhead of our protocol for low content sizes, is independent of the size of the published content. For a detailed asymptotic analysis of the ABE scheme used in our protocol, we refer the reader to [6], [17]. For context, a 7 seconds footage from an outdoor Nest camera device providing a 1080p stream generates over 1MB of data [22]. Furthermore, as we discuss in Section VII-B, the publisher can use the same pair of keys K_m and r_m for multiple contents, until the overhead is below the desired threshold.

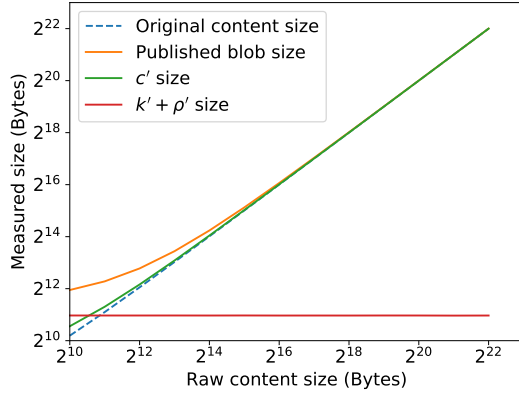


Fig. 4. Overhead breakdown for the different components of the published message

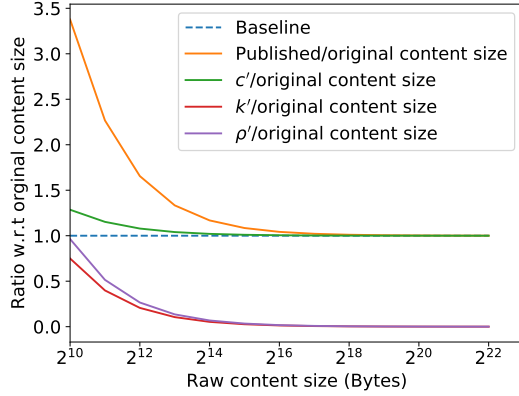


Fig. 5. Ratio of published content, c' , k' and ρ' over the original content size.

TABLE II
SIZE OVERHEAD

Content size (B)	Published blob	c'	k'	ρ'
2^{10}	3.3769	1.2855	0.7496	0.9632
2^{12}	1.6554	1.0787	0.2065	0.2657
2^{14}	1.1681	1.0202	0.0529	0.0682
2^{16}	1.0423	1.0051	0.0133	0.0171
2^{18}	1.0106	1.0013	0.0033	0.0043
2^{20}	1.0027	1.0003	0.0008	0.0011
2^{22}	1.0007	1.0001	0.0002	0.0003
2^{24}	1.0002	1.0000	5.23E-05	6.73E-05
2^{26}	1.0000	1.0000	1.30E-05	1.68E-05

Size overhead percentage for the total published blob, c' , k' and ρ' with respect to the original content size. All sizes are in Bytes.

B. Publication and Consumption Time Overhead

In addition to the size overhead, we also analysed the computational time overhead introduced by our protocol in the publication and consumption phases. As for the size overhead evaluation, we measure time overhead by running multiple tests with varying size of the original unencrypted content. For each original content size, we averaged the time required

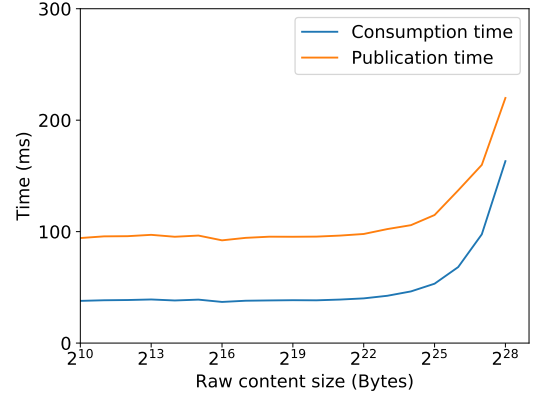


Fig. 6. Average time in ms required for content publication and consumption w.r.t. content size.

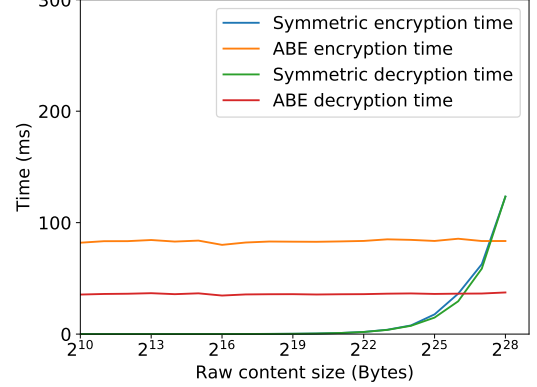


Fig. 7. Overhead breakdown for the different steps of the protocol

to publish and to consume the content over 100 iterations. As illustrated in Figure 6, both publication and consumption times remain considerably low. Indeed, even for content sizes in the order of hundreds of MB, both publication and consumption times remain well under the 300ms mark. It is interesting to note that there is a considerable difference between the time required for publication and the time required for consumption. Moreover, the difference appears to remain constant and is independent from content size. This behavior can be attributed to the specific ABE scheme we used [17]. Another interesting feature we can see from the graph is that for smaller content sizes ($2^{10} - 2^{22}$) the publication and consumption times remain approximately unchanged. However, for bigger contents both start to increase rapidly. This behaviour is easily explained by considering the computation time required by the different steps of the publication and consumption processes. As depicted in Figure 7, we see that the time required for ABE encryption/decryption remains constant, while the time required for symmetric encryption/decryption increases linearly and becomes significant between 2^{24} and 2^{28} . Overall, if we consider symmetric encryption as the baseline, i.e., the cheapest possible way to obtain confidentiality of the content, our protocol roughly doubles the computation time required (we require two layers of symmetric encryption).

Given that symmetric encryption does not provide most of the properties discussed in Section III-C and that the overhead of our protocol remains in the low hundreds of ms even for large contents, we believe our solution is widely applicable in real-life deployments.

VII. DISCUSSION

In this section, we provide a brief discussion on how our protocol can be transparently integrated with applications, the applicability of our construction in case of small content size and on possible modifications to adapt it to different scenarios.

A. Transparent Integration within Existing Stack

Our protocol can be transparently integrated in an IaaS deployment by introducing a new *security layer*, located between the network and application layers. The security layer intercepts interest requests from the application and issues the corresponding requests for the encrypted c' , K'_m and ρ'_m to the network layer. The only requirement for our protocol to be transparent to the application is that there exists a way for the producer to mark protected contents, along with an agreed protocol to identify K'_m and ρ'_m . In practical terms, this could be implemented by taking advantage of features similar to manifests in CCN. Upon publication, the producer issues a manifest file, indicating the fact that access to the content is restricted. The manifest should further include the name of main content c' , along with the names of the content blocks K'_m and ρ'_m . Upon manifest retrieval, the security layer issues separate interests for all the names, decrypts the content with the help of the local IaaS node, and returns the plaintext to the requesting application.

B. Dealing with Small Content

In some contexts, the size of published content can be very small. Since our protocol requires the publication of the encrypted first and second layer symmetric keys together with the content, in these scenarios the overhead of the protocol can become considerable. However, our protocol does not require the publisher to use a new key pair for every published content. The producer can reuse the same key pair K_m and r_m to encrypt multiple contents under the same access policy, until the overhead falls below a desired threshold. Effectively, this allows the producer to spread the cryptographic overhead over multiple publications, without degrading the security of the protocol.

C. Access Control without Gateways

Our system model assumes that the IoT devices are connected to the network through a gateway. In our construction, the gateway performs a number of cryptographic operations that cannot be performed by the IoT devices. In particular, ABE requires pairing operations and exponentiations, which are computationally expensive and not suitable for IoT devices. In order to adapt our protocol to run directly on the IoT devices, all ABE encryption operations can be moved to an external server managed by the IoT provider. The IoT

devices would generate random K_m and r_m and perform symmetric encryption on the content, like in the standard protocol. However, rather than encrypting K_m and r_m , the devices would forward them to the server, which would then compute r'_m and K'_m and publish them in the network.

D. Removing Revocation Time-Dependence

Cohherently with our system model, our protocol provides for revocation upon expiration of user subscription. If necessary, proxy re-encryption [23] or mediated encryption [24] can be integrated in our protocol in order to provide immediate revocation. However, this would considerably increase the reliance on IaaS, as well as introduce additional computational and/or storage overhead. Therefore, we don't see compelling reasons to use them beside in some particular cases.

E. Extension to general access control

Our scheme can be extended to become a general access control protocol for ICN by removing the reliance on external platforms. However, this requires presence of a Digital Right Management (DRM) system on the machine of the user in order to prevent him from accessing r_m . By executing the operations performed by IaaS in the original protocol in the DRM, our scheme maintains all its security properties. DRM systems are nowadays required by many popular platforms (e.g., Netflix, iTunes), therefore we believe this requirement would have limited impact on the adoption of our protocol.

VIII. RELATED WORK

There are a number of works related to our proposal. In [25] Da Silva et al. propose a proxy-based ABE access control scheme to achieve instantaneous revocation. The scheme allows fine-grained access control, and revocation is obtained by means of proxy re-encryption at the proxy machine. There are a number of issues with this work: first, the direct use of ABE on the published content heavily reduces the usability of the protocol. Secondly, given the small size of the access structure, users can store it locally before being revoked to retain access to protected contents. In [26] Misra et al. propose an access control mechanism based on the Broadcast Encryption scheme due to Tzeng et al. [27]. The proposed scheme provides access control for large multimedia contents which provides revocation and traitor tracing. However, the scheme requires the presence of a DRM on the users' machine, which can limit its applicability. Moreover, the scheme can only revoke a predefined number of users, and the computational and space overheads grow as this number increases. Finally, it is worth noting that security parameter used by the authors is much smaller than what is required to guarantee reasonable security. Increasing the size of the security parameter as required would correspondingly increase both computational and space overhead. In comparison, our scheme can revoke an unbounded number of users and the overhead is independent of the number of revocations. In [23], Liang et al. propose an attribute-based proxy re-encryption scheme, which allows fine grained access control as well as provides immediate user

revocation by employing PRE. The use of proxy re-encryption in addition to ABE increases the overhead of the scheme when compared to our proposal. In [24] Yang et al. propose a proxy-assisted approach for ABE where a proxy performs a partial ABE decryption, which is then completed by the user. The authors enable immediate revocation by means of an attribute revocation list maintained by the proxy. The protocol requires each proxy to maintain a revocation list, which needs to be updated by the producer at each revocation. In our scenario, this requires each IaaS node to maintain state for each user, and requires the producer to contact each IaaS platform every time a revocation is performed. Finally, both [23] and [24] are designed under a different system model, and are not directly applicable to the model defined in Section III-A

IX. CONCLUSION

In this paper we proposed a new, fine-grained access control protocol for IoT deployments over ICN. Our protocol preserves ICN network caching and location-independent content retrieval, while at the same time providing fine-grained access control, revocation and traitor tracing capabilities. We described and justified our security model and proved the security of our protocol within the model. Finally, we implemented and evaluated the proposed protocol, showing that both computational and size overhead remain low. Additionally, the protocol overhead can be further lowered by spreading it over multiple content publications, making it widely applicable in real-life IoT deployments.

REFERENCES

- [1] "Openfog architecture overview," <https://www.openfogconsortium.org/white-paper-reference-architecture/>.
- [2] "Cisco global cloud index: Forecast and methodology, 2015-2020," <http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf>.
- [3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT, 2009.
- [4] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, "Information centric networking in the iot: Experiments with ndn in the wild," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ser. ACM-ICN, 2014.
- [5] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE Symposium on Security and Privacy (SP '07)*, 2007.
- [6] J. Ning, X. Dong, Z. Cao, L. Wei, and X. Lin, "White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes," *IEEE Transactions on Information Forensics and Security*, 2015.
- [7] A. Dunkels, J. Alonso, T. Voigt, H. Ritter, and J. Schiller, *Connecting Wireless Sensornets with TCP/IP Networks*, ser. WWIC, 2004.
- [8] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "Iot gateway: Bridging-wireless sensor networks into internet of things," in *Proceedings of the 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, ser. EUC, 2010.
- [9] A. Compagno, M. Conti, and R. Droms, "Onboardicng: A secure protocol for on-boarding iot devices in icn," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ser. ACM-ICN '16, 2016.
- [10] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *2010 Proceedings IEEE INFOCOM*, 2010.
- [11] —, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS, 2010.
- [12] M. Li, S. Yu, K. Ren, and W. Lou, *Securing Personal Health Records in Cloud Computing: Patient-Centric and Fine-Grained Data Access Control in Multi-owner Settings*, ser. SecureComm, 2010.
- [13] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: A virtual machine-based platform for trusted computing," in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, ser. SOSP, 2003.
- [14] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards trusted cloud computing," in *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, ser. HotCloud, 2009.
- [15] "Trusted platform module," <https://trustedcomputinggroup.org/work-groups/trusted-platform-module/>.
- [16] Z. Liu, Z. Cao, and D. S. Wong, "Blackbox traceable cp-abe: How to catch people leaking their keys by selling decryption devices on ebay," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS, 2013.
- [17] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS, 2013.
- [18] B. Chor, A. Fiat, and M. Naor, *Tracing Traitors*, ser. CRYPTO, 1994.
- [19] D. Boneh and M. Franklin, *An Efficient Public Key Traitor Tracing Scheme*, ser. CRYPTO, 1999.
- [20] D. Boneh, A. Sahai, and B. Waters, *Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys*, ser. EUROCRYPT, 2006.
- [21] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, 2013.
- [22] "Nest bandwidth usage," <https://nest.com/support/article/How-much-bandwidth-will-Nest-Cam-use>.
- [23] K. Liang, L. Fang, W. Susilo, and D. S. Wong, "A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security," in *2013 5th International Conference on Intelligent Networking and Collaborative Systems*, ser. INCoS, 2013.
- [24] Y. Yang, J. K. Liu, K. Liang, K.-K. R. Choo, and J. Zhou, *Extended Proxy-Assisted Approach: Achieving Revocable Fine-Grained Encryption of Cloud Data*, ser. ESORICS, 2015.
- [25] R. S. da Silva and S. D. Zorzo, "An access control mechanism to ensure privacy in named data networking using attribute-based encryption with immediate revocation of privileges," in *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, 2015.
- [26] S. Misra, R. Tourani, F. Natividad, T. Mick, N. E. Majd, and H. Huang, "Accconf: An access control framework for leveraging in-network cached data in the icn-enabled wireless edge," *IEEE Transactions on Dependable and Secure Computing*, 2017.
- [27] W.-G. Tzeng and Z.-J. Tzeng, "A public-key traitor tracing scheme with revocation using dynamic shares," in *In Proceedings of Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems*, ser. PKC, 2001.