future internet

Article

Name-Based Security for Information-Centric Networking Architectures

Nikos Fotiou * and George C. Polyzos

Mobile Multimedia Laboratory, Dept. of Informatics, School of Information Sciences and Technology, Athens University of Economics and Business, Patision 76, 10434 Athens, Greece; polyzos@aueb.gr

* Correspondence: fotiou@aueb.gr

Received: 10 September 2019; Accepted: 30 October 2019; Published: 1 November 2019

Abstract: Information-Centric Networking (ICN) is an emerging communication paradigm built around content names. Securing ICN using named-based security is, therefore, a natural choice. For this paper, we designed and evaluated name-based security solutions that satisfy security requirements that are particular to ICN architectures. In order to achieve our goal, we leverage identity-based encryption, identity-based proxy re-encryption, and the emerging paradigm of decentralized identifiers. Our solutions support outsourcing content storage, content integrity protection and content authentication, and provenance verification, as well as access control. We show that our solutions have tolerable storage and computation overhead, thus proving their feasibility.

Keywords: identity-based encryption; Information-Centric Networking; proxy re-encryption; trust delegation

1. Introduction

Information-Centric Networking (ICN) architectures promise efficient content lookup and dissemination [1], but at the same time, they present new security challenges. In this paper, we present solutions for securing content distribution in ICN taking into consideration the unique properties of this emerging paradigm. In particular, we consider the case of a content owner who wishes to share content with some users. Content items can be provided either directly by the content owner or by third parties. The latter network entities can be authorized by the content owner—for example, a Content Distribution Network (CDN) node—or they may act on their own, as in an in-network cache. We wish to provide solutions that assure that: (i) content has not been modified during distribution, i.e., provide content integrity protection, (ii) sensitive content items are received from their real owner or from authorized third parties, i.e., provenance verification, and (iii) a received piece of content is what a user really asked for by name, i.e., authenticity verification. Moreover, we want to enable access control independently from where the content came from (e.g., a cache).

ICN architectures use content names as the main element of their (inter-)networking functions. Therefore, *names* are a natural choice for building security solutions that achieve the desired properties. In the following, we consider hierarchical content names—for example, "Disney/little princess/episode 1," "User A/Pictures/Profile picture," and so on. Content names are globally unique (at least within the context of an application), and their "roots" are *content owner* specific.

Building security primitives based on names offers some significant advantages. Firstly, names can be human readable, therefore, they can be memorable (as opposed, for example, to RSA public keys). For this reason, it should be easier to disseminate them using out of band mechanisms, such as by printing them on a business card or including them in a slide presentation. Secondly, names can be predictable, as in it could be easy to predict the name of a content item that has not yet been created, such as the name of the next chunk of a live video stream. Thirdly, the namespace of an ICN architecture can be hierarchical, reflecting real world organization and business relationships, as well as content structures.

Security solutions relying on name-based primitives have some additional features compared to traditional ones used for the same purposes. For example, it is easier to construct ephemeral keys, it is easier to delegate trust, it is possible to verify content integrity and authenticity at the same time, and access control can be implemented even at untrusted network entities. On the other hand, these security solutions introduce a convenience-security tradeoff because secret keys are generated by an external secret key generator using some known generator-specific public parameters. Hence, we can either have a single (or a few) key generators and consider their parameters well known (which is more convenient but less secure) or use many key generators (even one per user) and develop a parameter look-up mechanism (which is less convenient but more secure).

The goal of this paper was to review identity-based encryption and identity-based proxy re-encryption; to present security solutions based on these primitives that achieve content integrity protection, content authenticity and provenance verification, and access control; and to develop a convenient parameter look-up solution based on the emerging *Decentralized Identifiers* paradigm [2]. The functionality of these solutions is already built-in in most ICN architectures, using traditional public-key encryption: In this paper, we use identity-based encryption, thus providing additional properties. In summary, in this paper, we make the following contributions:

- We provide security solutions and notions that extend legacy security systems with new properties and capabilities.
- We design our approaches to take advantage of the unique features of the ICN paradigm stemmed from its location independence properties.
- We leverage Decentralized Identifiers to enhance the security of the proposed approaches.
- We propose a solution to the key revocation problem.
- We evaluate the proposed solutions and prove their feasibility.

The remainder of this paper is organized as follows. In Section 2, we give some background information about identity-based Encryption, proxy re-encryption, and decentralized identifiers, and we present related work in this area. In Section 3, we present the design of our solutions, which we detail in Section 4. In Section 5, we evaluate the performance and the security properties of our solutions. We conclude in Section 6 with a summary, discussion, and directions for future work.

2. Background and Related Work

2.1. Identity-Based Encryption

An Identity-Based Encryption (IBE) scheme is a public key encryption scheme in which an arbitrary string can be used as a public key. An IBE scheme is specified by the four algorithms [3], Setup, Extract, Encrypt, and Decrypt, summarized as follows:

- Setup: Executed once by a *Secret Key Generator* (SKG). It takes as input a security parameter and returns a master-secret key (*master*) and some generator specific *public parameters* (PP). *Master* is kept secret by the SKG, whereas PP are made publicly available. *Master* is only known to the SKG and it is used only for *extracting* other secret keys (see next algorithm).
- Extract: Executed by a SKG. It takes as input PP, *master*, and an arbitrary string ID, and returns a secret key SK_{ID}.
- Encrypt: Executed by users. It takes as input an arbitrary string ID, a message M, and PP, and returns a ciphertext C_{ID}.
- Decrypt: Executed by users. It takes as input C_{ID} the corresponding private decryption key SK_{ID}, and returns the message M.

2.2. *Identity-Based Proxy Re-Encryption*

An Identity-Based Proxy Re-Encryption (IB-PRE) scheme is an extension to IBE that allows a third party to alter a ciphertext, encrypted using an arbitrary string ID1, in a way that another user that owns a secret key SK_{ID2} can decrypt it. Such a scheme specifies two new algorithms [4], RKGen and Reencrypt, in addition to the IBE algorithms already discussed:

- RKGen: it is executed by a user that owns a secret key SK_{ID1}. It takes as input PP, the secret key SK_{ID1} and an arbitrary string ID2 and generates a (public) re-encryption key $RK_{ID1 \to ID2}$.
- ReEncrypt: it is executed by a third party. It takes as input PP, a re-encryption key $RK_{ID1 \to ID2}$ and a ciphertext C_{ID1} and outputs a new ciphertext C_{ID2}.

The ciphertext generated by the ReEncrypt algorithm can be decrypted only using SK_{ID2}. The entity that performs the re-encryption learns nothing about the encrypted plaintext or about the secret keys that correspond to ID1 and ID2.

Figure 1 illustrates an IB-PRE example. There are three users each owning a name root, namely "User A", "User B", and "User C". "User A" and "User B" have received the secret key that corresponds to their roots from a SKG (Step 2). "User C" encrypts a message by executing the IBE Encrypt algorithm using "User A" as a key and stores the generated ciphertext in a storage entity. Supposedly, "User A" wishes to enable "User B" to access the ciphertext generated by "User C" in the previous step: She generates an appropriate re-encryption key and sends it to the storage entity. The storage entity is now able to re-encrypt the ciphertext in a way that can be decrypted using the private key of "User A".

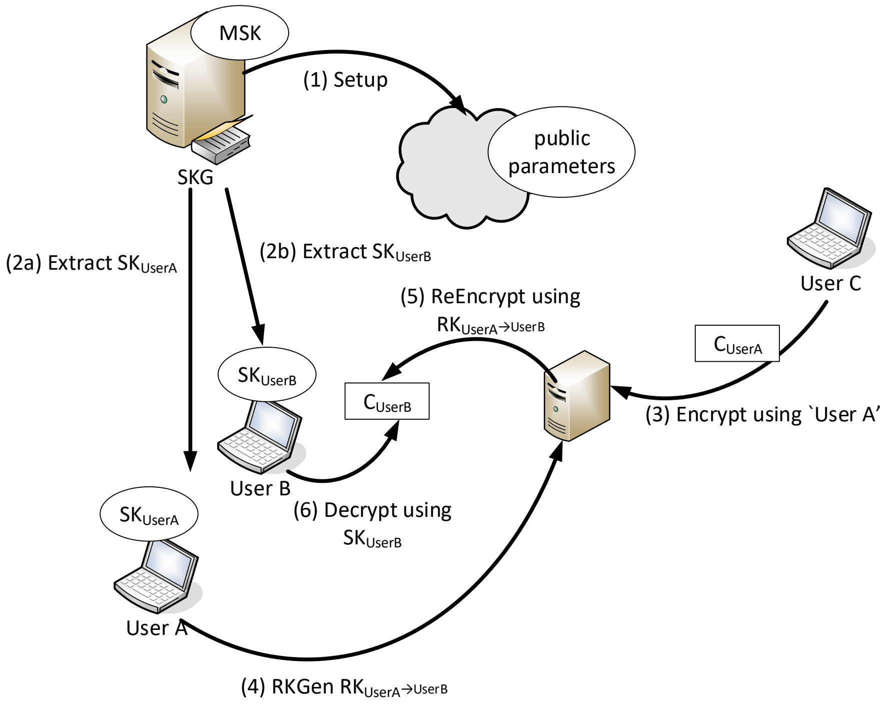

Figure 1. Identity-Based Proxy Re-Encryption (IB-PRE) example. MSK = Master Secret Key; SK = Secret Key; RKGen = Re-Encrypt Key Generation.

2.3. *Decentralized Identifiers*

Decentralized Identifiers (DIDs) are a new type of identifier that is globally unique, resolvable with high availability, and cryptographically verifiable [5]. From a high level perspective, a DID system can be viewed as a *registry* where key-value pairs are stored. The key part of this pair is the

DID, which can be any arbitrary string. The value is a *DID document*, i.e., a JSON-LD [6] encoded document that contains (among other information) pubic keys that can be used for linking a DID to that document, service endpoints related to that DID, as well as auxiliary information that can be used for verifying the integrity of the document (e.g., a digital signature) [2]. A registry can be implemented using blockchains, distributed ledgers, (decentralized) P2P networks, or other systems with similar capabilities. It should be noted that DID specifications do not dictate how a registry is implemented. Instead, they define the methods that it should implement.

Suppose that an entity (the prover) wants to prove to another entity (the verifier) that he is the owner of a DID. The verifier retrieves the corresponding DID document form the registry and extracts the public key associated with that DID. Then, using an "authentication" method that depends on the type of the public key, which is also defined in the DID document, authenticates the prover (e.g., the verifier generates a challenge, and the prover must provide a digital signature for that challenge).

Compared to traditional security certificates, DIDs (and DID documents) have some significant advantages. Firstly, a DID acts as an indirection point towards some public information (i.e., the DID document, which may include a certificate), and any modification to this public information does not affect the DID. Secondly, DID documents are stored in a registry (maintained by a 3^{rd}party), and before any other operation, a verifying entity may lookup this registry and obtain the latest version of the DID document. This property is of importance when it comes to revocation. Thirdly, DIDs offer "self-sovereignty," i.e., DID owners (and only them) can freely manage their DIDs and the corresponding DID documents.

2.4. Related Work

Name-based security solutions have been considered by many related research efforts. Zhang et al. [7] leverage identity-based encryption and identity-based signatures to provide content integrity and confidentiality protection for the CCN architecture. The difference between the content integrity mechanism presented in that paper and our solution is that with our solution only the content owner can generate a digital signature. This has the advantage that delegated storage entities (e.g., CDN nodes) cannot modify a stored item, but it has the disadvantage that it cannot be used for "live" content stored in a location outside the administrative realm of the content owner. The content confidentiality solution presented by Zhang et al. does not provide forward secrecy. Although we do not propose any particular confidentiality mechanism, our solution can be used with the Transport Layer Security protocol -like handshakes protocols that achieve ephemeral content encryption keys.

Hamdane et al. [8] achieve similar goals by using hierarchical identity-based encryption. The advantage of hierarchical identity-based encryption is that once the secret key that corresponds to a content name prefix has been generated, the keys for the names that use this prefix can be generated without the involvement of the *SKG*. This comes with additional computational and storage overhead. In this paper, we allow each content owner to maintain his own *SKG*, hence interacting with it does not impose significant network overhead.

Wood and Uzun [9] use proxy re-encryption to implement an end-to-end content encryption mechanism for CCN that does not prevent content caching. The difference between this approach and our solution is that with our solution a storage endpoint can re-encrypt a content item for a user only if the user is authorized to access it, whereas in the construction presented in [9] a storage endpoint can perform this operation for all items; hence, if access control is required the storage endpoint must be trusted. Similarly, Suksomboon et al. [10] use proxy re-encryption to provide access control by semi-trusted proxies. Their approach has the same limitation as [9].

Bernardini et al. [11] use proxy re-encryption to build a privacy enhancing mechanism for the NDN architecture. With their solution, they perform hop-by-hop name and data encryption and re-encryption, in this way hiding user interests. In this paper, we do not provide any privacy-enhancing solution; however, we believe that our constructions are compatible with the solution in [11].

Attribute-based encryption (ABE) [12] is a type of encryption that is very similar to IBE and for this reason it has been considered by many related ICN-specific efforts (e.g., [13–15]). However, when it comes to our goals, ABE has bigger computational and storage overhead without offering any additional security advantage. In any case, we believe that the solutions presented in this paper can be easily adapted to use ABE instead of IBE.

3. System Design

3.1. System Entities and Namespace Structure

Our system design considers real world entities that own some content (*content owners*) and want to share it with other users. A content owner owns some content name roots. Users are generally anonymous. However, there are cases where users have to be identified (e.g., in order to apply access control); in these cases, user names, similar to content name roots, are considered. Content items are stored in network locations that may or may not belong to the administrative realm of the content owner. Content items are requested and retrieved using standard ICN functions.

Our design assumes that the namespace of content names is organized in Direct Acyclic Graphs (DAGs), where the root of each graph, i.e., the name root, is globally unique. The granularity and the semantics of the names are application-specific. Throughout the paper, for illustrative purposes, we consider the case of a file sharing application. Suppose a user in this application, that owns the content name root "User A": the namespace in that case can be organized as in Figure 2. In this example, "User A/Videos/Holidays" and "User A/Videos/Graduation/Party/Chunk 1" are two valid names that share the same root, as well as the same name "prefix". As we will discuss in the following sections, a security solution can be applied to a whole name prefix.

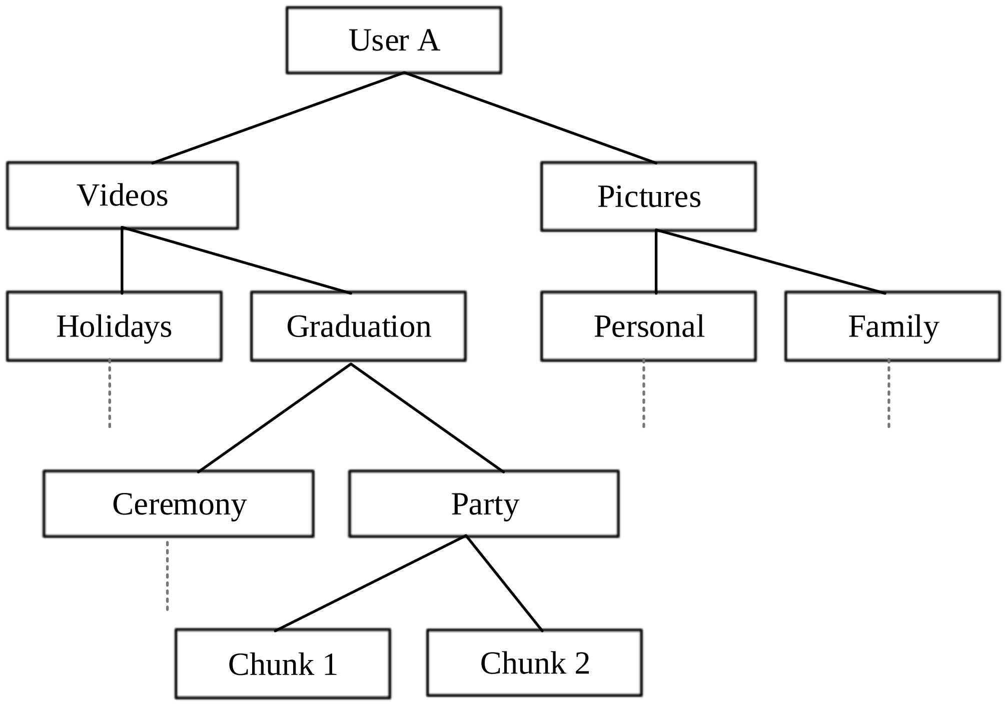

Figure 2. Namespace organization for a sample file sharing application.

3.2. The Key Escrow Problem and a DID-Based Solution

Identity-based security schemes often suffer from the so-called key escrow problem, i.e., the fact that a third party (the SKG) knows all secret keys. Even if we assume that this party is reliable and does not misuse the generated secret keys, these systems still are threatened by security breaches that would reveal the *master* key (note that this is the secret key of the SKG, used to generate all user secret keys). An obvious solution to this problem is to use multiple SKGs; each user could use the SKG they trust, or even their own. Having multiple SKGs, however, leads to another problem: How are PP disseminated? In order to solve this problem, we rely on DIDs. Our design assumes a DID registry that is not part of the ICN network; instead, entities interact with that registry using legacy networking protocols.

This name roots are treated by our system as DIDs, and they are associated with a DID document stored in a registry. Each DID document contains the PP required for performing the corresponding IBE and IB-PRE operations (related to a content name with that root). Therefore, in order for any third party to reliably learn some PP, it simply has to lookup the registry. A DID document in our system also contains a public key used for authenticating the root owner when he wants to modify the document in the registry. As we discuss in Section 5, we implemented the registry functionality using the Hyperledger Fabric blockchain technology.

Back to our file sharing example, suppose that an entity wants to perform some cryptographic operations related to the content name "User A/Videos/Graduation". The root of this name is "User A", hence its searching of the registry for the document associated with the DID "User A": This document contains the desired PP.

At this point, we should note that even if per-user SKGs are used, the problem of disseminating PP is not equivalent to the problem of disseminating users' RSA pubic keys (as in the current Internet). This is true because even if per-user SKGs are used, the same PP may be used for generating multiple keys. The security solutions presented in the following section use content names as keys; all these keys share the same PP, i.e., the PP of the content owner. Therefore, if the PP of a content owner are known, these solutions can be used without interacting with the DID registry. So, back to our file sharing example, if a user has retrieved the PP that correspond to "User A/Videos/Graduation", he can use them for performing cryptographic operations with the name "User A/Videos/Holidays".

3.3. Name Versioning

Content names are used in our system as public keys which are associated with a secret key. As in any crypto system, a secret key may be breached, hence the corresponding public key has to be modified (we discuss key revocation in more detail in Section 5.3). In order to enable content name modification, we consider the name "versioning". A name version is a simple number appended to a content name when necessary. For instance, and as we discuss in more detail in the following section, when a user requests a content item by its name, a network node that hosts this item may reply indicating in its response the version of the name it hosts, e.g., a user may request 'User A/Videos/Graduation", and a network node may reply with "User A/Videos/Graduation:1". In that case, the latter name will be used as the public key in all subsequent cryptographic operations. A content owner may include in the corresponding DID document all deprecated version numbers (akin to a key revocation list).

Similar to this concept, there are scenarios where ephemeral names are required (e.g., a network entity may be authorized to host a content name for limited time). In these cases, a timestamp can be appended to a content name. The semantics of this timestamp are that "this name and its corresponding secret key can be used up until the specified timestamp" .

4. Name-Based Security Solutions

4.1. Content Integrity Protection and Content Authentication

The integrity of a content item is protected by a digital signature generated by the content owner. Providing that the underlay IBE scheme is chosen-ciphertext attack (CCA) secure, a digital signature is generated as follows [16] (Figure 3a):

1. The content owner selects a secure hash function and calculates the item's hash h.
2. She creates $SK_{content\_name/h}$ by invoking the *Extract* algorithm of the SKG, i.e., she creates the secret key that corresponds to the content name appended with h.

The latter secret key is the (public) digital signature of the item identified by *content_name*. A digital signature can be trivially verified by anybody using the following steps (Figure 3b):

1. Calculate the hash h of the item covered by the signature.
2. Select a random number r.
3. Use the IBE *Encrypt* algorithm to encrypt r using as a key "$content\_name/h$" and as PP the content owner's PP.
4. Verify that the ciphertext produced in the previous step can be decrypted using the IBE *Decrypt* algorithm and the signature as a key.

It can be observed that $SK_{content\_name/h}$ is treated as a public information, and this solution does not require any secret key associated with the content. Hence, keeping track of the current version of the content name is not required by this solution.

An interesting property of this solution is that it can be used to verify content authenticity. With this solution, signatures can only be generated by the content owners, since the signature generation relies on the *Extract* IBE algorithm that can be executed only by the SKG. Therefore, a signature provides an unforgeable means of linking content items to their owners. As a result, a user can be sure that the content item she received is indeed what she asked for.

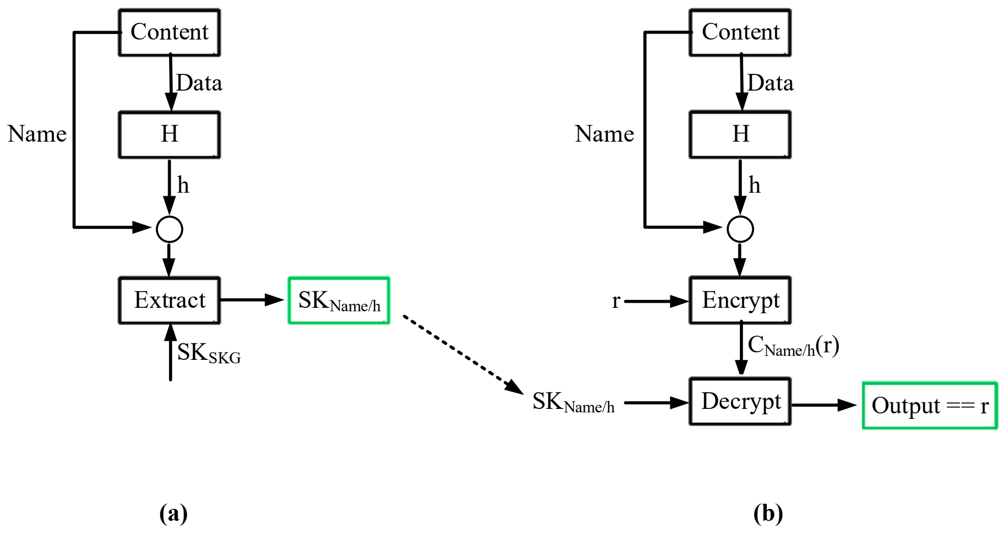

Figure 3. Digital signature: (**a**) creation; (**b**) verification.

In order to show the significance of this property, we present an authentication attack applicable to the NDN architecture. Suppose, a user requests the item "User A/Pictures/Picture1" (where "User A" is a name root owned by the content owner). The header of the corresponding data packet in the NDN architecture contains (among others) the following fields: the content name, a digital signature over the packet data and the content name, and a pointer to the public key of "User A". An attacker can modify the packet data, compute a new signature using his key, and replace the information in the header with the new signature and a pointer to the attacker's public key. The integrity of the new packet is still preserved, but this is not what the user really asked for. The reason why this attack is possible is because there is no binding between the public key of "User A" and the content name; in other words, a user by receiving only a public key (and not for example a digital certificate) cannot verify that this key indeed belongs to "User A". In our solution, this binding exists since the word

"User A" is both the root of the content item name *and* the public key of "User A". This attack is not specific to ICN: Recently researchers used a similar attack (and other techniques) to break Apple's iMessage messaging system [17].

4.2. Outsourcing Content Storage and Content Provenance Verification

Trust delegation is a typical application of IBE [16]. In the context of ICN, trust delegation is of significant importance, since a piece of content may be hosted by multiple entities. The solution presented here provides a secure way for a content owner to authorize an entity to host (i.e., store and provide on demand) part of the namespace she owns (hence, the content items associated with it). This is achieved as follows: firstly, the content owner uses her SKG and generates the secret keys that correspond to the content names (or the portion of the namespace) the authorized entity will store, then, the content owner distributes these keys to that entity. Back to our file sharing example: Suppose that "User A" wants to authorize "CDN-A" to store all family photos; he generates $SK_{UserA/Pictures/Family}$ and securely distributes it to "CDN-A". This key is then used for proving content provenance (see next). It should be noted that the rightful owner of these keys is the content owner and that the authorized entity simply acts on the owner's behalf, therefore, the fact that the content owner knows these keys is not considered key escrow. Moreover, key distribution should be secured (however, key distribution is out of the scope of this paper).

Content provenance verification is usually referred to as the "verification of the identity of the transmitting entity." However, in ICN, users should not care who sends a content item, as long as the content authenticity can be verified. In this work, we refer to content provenance verification as the process of verifying that *a transmitting entity is authorized by the content owner to host and share the requested item*. Therefore, we are not concerned with WHO the transmitting entity is. Content provenance verification, in this context, is useful in cases where a user wants to receive an item only from endpoints trusted by the content owner, for example, for accounting reasons, spam prevention, phishing protection, and so on. Provenance verification can be achieved using a simple challenge-response protocol and the secret key generated by the content owner. In the following, we give an example of a challenge-response routine that can be incorporated into a key agreement protocol (in our description we consider the TLS-like protocol presented in [18]). After all messages have been exchanged, a user is able to verify that she will receive the desired item from an authorized endpoint:

- The user sends to the network a *Hello* message indicating the portion of the namespace in which he is interested in (for example, in our file sharing example this could be "User A/Pictures/Family"). This message is forwarded using standard ICN mechanisms.
- A network endpoint, authorized to host (part of) this namespace, responds with a *Hello response*. If the endpoint is authorized for a less fine-grained portion of the namespace (for example, "User A/Pictures"), it indicates that in its response. Similarly, if the endpoint is authorized for a limited time, or it is authorized for a specific version of the namespace, it includes in its response the authorization expiration time (in the form of a timestamp) and/or the version number. Multiple endpoints may reply to a Hello message.
- The user selects a random number r, encrypts it using the IBE *Encrypt* algorithm, the content owner's PP, and an appropriate key (e.g., in our previous example, if an endpoint is authorized for a limited time, the encryption key would be "User A/Pictures/Family:timestamp"), includes this ciphertext to the next message of the key agreement protocol, and sends it to the network.
- An appropriate endpoint decrypts the ciphertext and sends the decrypted plaintext back to the user as part of the key agreement protocol.

If the number that the endpoint sends in the last message is equal to the number the user generated, then the endpoint is considered authorized. It should be noted that the key agreement protocol specified in [18] does not require from the user to select a particular endpoint, on the contrary it allows users to initiate the key agreement protocol with one endpoint and finalize it with another, providing that these endpoints share the same secret key.

4.3. Access Control

For this solution, we leverage IB-PRE to provide access control on content items stored in network endpoints outside the administrative realm of the content owner, as in [19]. We assume that each content item is encrypted by the content owner using a symmetric encryption algorithm (each item with a different key) and each item is protected by an access control policy. Our goal is to allow authorized users to access the corresponding encryption key. Each access control policy is identified by a name similar to a content name (for example "User A/Friends") and it may protect multiple items. In order to achieve our goal, content owners encrypt the symmetric encryption keys with the IBE *Encrypt* algorithm using the name of the policy as a key; we refer to the latter ciphertext as $C_{policy\_name}(key)$. Moreover, for each authorized user, the content owner creates an IB-PRE re-encryption key of the form $RK_{policy\_name \rightarrow user\_name}$. Finally, the $C_{policy\_name}(key)$ ciphertexts, the re-encryption keys, and a list of authorized users (and their corresponding PP) are transferred to the content storage network endpoints. With this information, an endpoint can re-encrypt a $C_{policy\_name}(key)$ into $C_{user\_name}(key)$.

5. Implementation and Evaluation

ICN operations may take place in constrained devices (e.g., smartphones) or in nodes that experience high volumes of traffic (e.g., CDN nodes). For this reason, in order for a solution to be realistic and feasible, it should add tolerable storage and computation overhead.

5.1. DID Performance Evaluation

5.1.1. DID Registry Management Overhead

We implemented a proof of concept DID registry using Hyperledger Fabric (henceforth, we will simply refer to it as Fabric). Fabric is a private, permissioned blockchain technology, i.e., a blockchain system where membership is controlled. Fabric involves no monetary cost, low computational complexity, and insignificant delays (as opposed, for example, to public permissionless blockchains, such as the Ethereum [20]).

From a high-level perspective (For more details about the Fabric architecture, interested users are referred to [21]), Fabric allows special nodes, referred to as *endorsing peers* , to store and execute arbitrary programs, called *chaincode*. Chaincodes are replicated to many endorsing peers and the state associated with them is stored in a ledger distributed to a set of *committing peers* (In reality a peer can be at the same time both endorsing and committing).A chaincode is executed simultaneously and in parallel by many endorsing peers, and the final output of the execution is determined based on an *endorsement policy* defined by the chaincode owner. An endorsement policy can be, for example, that n out of m endorsing peers should execute the chaincode and produce the same output, otherwise the execution is not accepted. If the output of the execution has to be stored in the ledger, then it is forwarded to a special node referred to as the *orderer*. The orderer instructs committing peers to re-validate that the endorsement policy has been respected and to update the state of the chaincode in the ledger. Any "client application" can invoke a chaincode by interacting with a (single) *gateway* using the Fabric API. The gateway is responsible for: instructing all endorsing peers to execute the chaincode, collecting the final outcome, verifying the endorsement policy, and returning the outcome to the application.

In order to evaluate the overhead of our solution, we created a network composed of six peers, and we stored in all peers a chaincode that implements our registry. Our endorsement policy defines that at least three out six peers should produce the same output in order for that output to be accepted. The basic operations of our chaincodes are the following: check if a DID exists, retrieve the DID document that corresponds to a particular DID, and create a new DID document. Table 1 reports the amount of time required to perform these operations.

Table 1. Execution time of chaincode operations measured in seconds. DID = Decentralized Identifier.

Operation	Time in Seconds
Check if DID exists	0.0426
Retrieve DID document	0.030
Create DID document	2.59

As it can be seen except from the DID document creation operations, which requires $\sim$ 2.6 s, all other operations are executed in less than 50 ms. Our registry has been implemented using HashMaps therefore DID lookup complexity is O(1) no matter the number of entries. Furthermore, the size of DID document is a few bytes (and less than 1 KB); hence, a DID registry should be able to handle a very large number of documents (each corresponding to a name root).

5.1.2. DID Cryptographic Operations Overhead

In our DID registry implementation, only DID document owners are allowed to modify a DID document. In order to achieve this, a DID registry must authenticate a users based on the key included in the DID document.

W3C's (draft) DID specification defines that the keys used for user authentication must be compliant with the authentication cryptographic suites included in the "Linked Data Cryptographic Suite Registry" [22] . Currently, this registry includes four suites, and for our implementation, we have selected the "Ed25519 Signature 2018" suite [23]. This suite uses keys derived from the curve25519 elliptic curve and defines that authentication should be implemented using Ed25519 signatures. The size of the public key is 32 bytes and the size of a signature is 64 bytes.

User authentication is implemented using a simple challenge-response protocol, where the registry generates a random number and the user responds with the digital signature of that number (All messages are exchanged over TLS hence man-in-the-middle attacks are prevented).

DIDs require the following cryptographic operations: key-pair generation, signature creation, and signature verification. We have implemented all these cryptographic using the TweetNaCl cryptographic library [24]. Table 2 measures the execution speed of each operation in a high-end PC (3.4 GHz intel-i7 CPU, 16 GB of RAM, running Ubuntu 16.04 Server), a first-generation Raspberry Pi (700 MHz BCM2835 CPU, 512 MB of RAM, running Raspbian GNU/Linux 9 distribution), and the Safari browser of an iPhone 7s (Running iOS 12.3.1) (For this measurement we used the JavaScript port of the library available here: https://tweetnacl.js.org).

Table 2. Execution time of cryptographic operations measured in ms.

Operation	PC	RPi	Mobile Phone
Key generation	13.6	126	32
Signature creation	13.2	126	18.9
Signature verification	24.1	231	22

As it can be observed, all cryptographic operations are executed very fast even by the constrained devices.

5.2. Performance Evaluation of the Proposed Security Solutions

In order to evaluate the storage and computation overhead of the proposed solutions, we implemented the IB-PRE scheme of Green and Ateniese [4] using the Charm cryptographic library [25]. In our measurements, we considered a setup that provides security equivalent to RSA with an 1024 bits key. The measurements related to the Encrypt, ReEncrypt, and Decrypt algorithms concern the encryption of a 128-bit random number (i.e., the length of the symmetric encryption key used for encrypting files). Table 3 presents the computation overhead of the IB-PRE algorithms. Measurements were obtained in an Ubuntu 16.04 machine, running in a single core of an Intel i7-4440 3.1 GHz processor with 16 GB of RAM, using Charm v0.43. Table 4 presents the size of the various variables used by the IB-PRE algorithms.

Table 3. IB-PRE algorithm computation times.

Algorithm	Time (ms)
Extract	4.0
Encrypt	6.2
RKGen	9.8
ReEncrypt	1.0
Decrypt	5.3

Table 4. Size of variables used by the IB-PRE algorithms.

Variable	Size (bits)
Public Parameters (PP)	2048
Master Secret Key ($master$)	160
Secret Key (SK)	1024
Ciphertext (C)	2048
Re-encryption key (RK)	2048

5.3. Security Evaluation

DID registry security is not in the scope of this work. A DID registry implementation should provide mechanisms that assure: (i) the uniqueness of DIDs, i.e., a user cannot register a DID (i.e., a name root) that has already been registered, (ii) DID document access control, i.e., only the owner of a DID document should be able to modify it, and (iii) user account management, i.e., DID owners should be able to recover/modify a lost/breached secret key (used for authenticating themselves to the DID registry).

The scheme used in our paper is CCA-secure, and as proved in [16], it can be used for generating digital signatures using the algorithm presented in Section 4.1. The digital signatures generated by this algorithm have all the properties of traditional digital signatures, i.e., unforgeability (only the content owner can generate a valid signature for the associated name), non-re-usability (the signature that corresponds to a content name cannot be used with another name), and non-repudiation (a content owner cannot deny having generated a valid signature).

As already discussed, a digital signature can only be generated by the content owner, and it cannot be generated even by authorized storage endpoints, i.e., network entities that know the secret key that corresponds to a content name. This has the advantage that users can make sure that they always receive authentic content, but on the other hand, it has the disadvantage that 3^{rd} parties, including authorized storage points, cannot sign "dynamic" content (e.g., a live stream).

Our access control solution is secure even in the presence of misbehaving storage endpoint. A storage endpoint cannot create a $C_{user\_name}(key)$ for unauthorized users since it cannot create the re-encryption key required for this process. Moreover, storage endpoints learn no secret information. For a more thorough security analysis of this solution, interested readers are referred to [19].

Key Revocation

Our solutions rely on the following secret keys: (i) *master* (i.e., the master secret key of a SKG), and (ii) the secret key associated with a content name. *master* is the most important key: any entity that learns this key can re-generate all the keys that the SKG has extracted. *master* keys must be securely stored by SKGs. In case a *master* key is lost, the corresponding SKG should re-execute the Setup IBE algorithm and produce new PP. Accordingly, the affected user(s) should update their DID documents and re-generate all secret and re-encryption keys. *master* is the only key that can be used for extracting other keys. On the contrary, the secret keys that correspond to a name cannot be used for generating other secret keys, even for names that use the breached name as a prefix. For example, the key that corresponds to "User A/Pictures" cannot be used for extracting the key that corresponds to "User A/Pictures/Holidays", even if these keys share the same PP. On the other hand, the loss of a secret key means that the corresponding name cannot be used as a public key. For this reason, every time a key is revoked, the version number of the corresponding name is increased, and a new secret key is generated.

6. Conclusions

In this paper, we briefly reviewed identity-based encryption and identity-based proxy re-encryption. We developed solutions for ICN architectures based on these primitives that achieve outsourcing of content storage, content integrity protection, content authenticity, and provenance verification, as well as access control. We argued that the use of name-based security contributes to some unique features of our solutions: It is possible to create ephemeral keys, it is trivial to verify content authenticity, trust can be easily delegated, and access control can be enforced even by untrusted network entities.

Named-based security solutions create a convenience-security tradeoff, i.e., a single PKG (or a small number of PKGs) can be considered (convenient but not secure), or multiple SKGs alongside a PP lookup system can be implemented (more secure but less convenient): We claim that it is possible and convenient to use the secure extreme point of this tradeoff by using per-user key generators. To this end, we proposed a parameter dissemination mechanism using the emerging DID paradigm. The performance evaluation of the implemented solution demonstrated that it is feasible for many applications.

Our DID registration approach is based on the "first come, first served" principle. This approach can further be extended to prevent illegitimate name registrations. Moreover, in this work we considered the DID registry system and the ICN network as two distinct and decoupled entities. We believe, however, that ICN architectures have the potential to benefit DID registry implementations. An integrated ICN-DID architecture seems a very promising future research direction.

Author Contributions: All authors contributed equally to this work.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A Survey of Information-Centric Networking Research. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1024–1049. [CrossRef]
2. W3C Credentials Community Group. Decentralized Identifiers (DIDs) v0.13. Available online: https://www.w3.org/2019/08/did-20190828/ (accessed on 31 October 2019).
3. Ateniese, G.; Fu, K.; Green, M.; Hohenberger, S. Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. *ACM Trans. Inf. Syst. Secur.* **2006**, *9*, 1–30. [CrossRef]
4. Green, M.; Ateniese, G. Identity-Based Proxy Re-Encryption. In *Applied Cryptography and Network Security*; Katz, J., Yung, M., Eds.; Springer: Berlin, Germany, 2007; Volume 4521, pp. 288–306.
5. W3C Credentials Community Group. A Primer for Decentralized Identifiers, 2019. Available online: https://w3c-ccg.github.io/did-primer/ (accessed on 31 October 2019).
6. Sporny, M.; Kellogg, G.; Lanthaler, M. JSON-LD 1.0. Available online: https://www.w3.org/TR/json-ld/ (accessed on 31 October 2019).

7. Zhang, X.; Chang, K.; Xiong, H.; Wen, Y.; Shi, G.; Wang, G. Towards name-based trust and security for content-centric network. In Proceedings of the 2011 19th IEEE International Conference on Network Protocols, Vancouver, BC, Canada, 17–20 October 2011.
8. Hamdane, B.; Serhrouchni, A.; Fadlallah, A.; Fatmi, S.G.E. Named-Data security scheme for Named Data Networking. In Proceedings of the 2012 Third International Conference on the Network of the Future (NOF), Gammarth, Tunisia, 21–23 November 2012.
9. Wood, C.A.; Uzun, E. Flexible end-to-end content security in CCN. In Proceedings of the 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 10–13 January 2014.
10. Suksomboon, K.; Tagami, A.; Basu, A.; Kurihara, J. IPRES: In-device Proxy Re-encryption Service for Secure ICN. In Proceedings of the 4th ACM Conference on Information-Centric Networking, Berlin, Germany, 26–28 September 2017.
11. Bernardini, C.; Marchal, S.; Asghar, M.R.; Crispo, B. PrivICN: Privacy-preserving content retrieval in information-centric networking. *Comput. Netw.* **2019**, *149*, 13–28. [CrossRef]
12. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP '07), Berkeley, CA, USA, 20–23 May 2007.
13. Ion, M.; Zhang, J.; Schooler, E.M. Toward Content-centric Privacy in ICN: Attribute-based Encryption and Routing. In Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking, Hong Kong, China, 12 August 2013.
14. Li, B.; Verleker, A.P.; Huang, D.; Wang, Z.; Zhu, Y. Attribute-based access control for ICN naming scheme. *IEEE Trans. Dependable Secur. Comput.* **2016**, *15*, 194–206. [CrossRef]
15. Zhang, Z.; Yu, Y.; Afanasyev, A.; Burke, J.; Zhang, L. NAC: Name-based Access Control in Named Data Networking. In Proceedings of the 4th ACM Conference on Information-Centric Networking, Berlin, Germany, 26–28 September 2017.
16. Boneh, D.; Franklin, M. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology - CRYPTO 2001*; Kilian, J., Ed.; Springer: Berlin, Germany, 2001; Volume 2139, pp. 213–229.
17. Garman, C.; Green, M.; Kaptchuk, G.; Miers, I.; Rushanan, M. Dancing on the Lip of the Volcano: Chosen Ciphertext Attacks on Apple iMessage. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016.
18. Fotiou, N.; Xylomenos, G.; Polyzos, G.C. Securing Information-Centric Networking without Negating Middleboxes. In Proceedings of the 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Larnaca, Cyprus, 21–23 November 2016.
19. Fotiou, N.; Polyzos, G.C. Securing Content Sharing over ICN. In Proceedings of the 3rd ACM Conference on Information-Centric Networking, Kyoto, Japan, 26–28 September 2016.
20. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
21. Androulaki, E.; Barger, A.; Bortnikov, B.; Cachin, C.; Christidis, K.; Caro, A.D.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018. [CrossRef]
22. W3C Credentials Community Group. Linked Data Cryptographic Suite Registry. Available online: https://w3c-ccg.github.io/ld-cryptosuite-registry/ (accessed on 31 October 2019).
23. Sabadello, M. Ed25519 Signature 2018. Available online: https://w3c-dvcg.github.io/lds-ed25519-2018/ (accessed on 31 October 2019).
24. Bernstein, D.J.; Van Gastel, B.; Janssen, W.; Lange, T.; Schwabe, P.; Smetsers, S. TweetNaCl: A Crypto Library in 100 Tweets. In Proceedings of International Conference on Cryptology and Information Security in Latin America 2014, Florianópolis, Brazil, 17–19 September 2014.
25. Akinyele, J.A.; Garman, C.; Miers, I.; Pagano, M.W.; Rushanan, M.; Green, M.; Rubin, A.D. Charm: A framework for rapidly prototyping cryptosystems. *J. Cryptogr. Eng.* **2013**, *3*, 111–128. [CrossRef]