Decentralized Authorization and Privacy-Enhanced Routing for Information-Centric Networks

Mariana Raykova Hasnain Lakhani Hasanat Kazmi* Ashish Gehani
SRI International

ABSTRACT

As information-centric networks are deployed in increasingly diverse settings, there is a growing need to protect the privacy of participants. We describe the design, implementation, and evaluation of a security framework that achieves this. It ensures the integrity and confidentiality of published content, the associated descriptive metadata, and the interests of subscribers.

Publishers can scope access to the content, as well as which nodes in the network can broker access to it. Subscribers can limit which nodes can see their interests. Scopes are defined as policies over attributes of the individual nodes. The system transparently realizes the policies with suitable cryptographic primitives. It supports deployment in heterogeneous mobile ad hoc environments where trust may derive from multiple independent sources. Further, no external public key infrastructure is assumed. We also report on the overhead that the security adds in actual deployments on Android devices.

1. INTRODUCTION

Information-centric networking (ICN) is a paradigm for content distribution and retrieval. It departs from the traditional source-destination data routing model by shifting to a framework where naming and routing are driven by the data content. The general premise is that data movement should be determined by the interests of nodes in the network and the extent to which the interests match descriptions of published content. This networking paradigm raises many questions about the naming of objects, caching and forwarding algorithms, granularity of data dissemination, scaling for large deployments, and interoperation with extant networks. These questions have motivated much research in the area [27]. In 2011, the United States Department of Defense's Advanced Research Projects Agency (DARPA) initiated the Content-Based Mobile Edge Networking (CBMEN) [10] program to develop an ICN solution that works on commodity mobile devices. By 2014, ICN technologies started being commercialized – for example, PARC is currently engaged with network device manufacturers and service providers for productizing its Content Centric Networking prototype [11].

An important question that is inherently related to the new ICN routing mechanisms is the privacy of the data in the network [1]. Unlike solutions with point-to-point communication paths, in the case of ICN the metadata needed for the routing algorithm is directly related to the content of the published data objects. This is because routing decisions are based on matches between content descriptions and nodes' interests. While information about these matches has to be revealed in order to enable the networking functionality, the ideal privacy goal for ICN is to reveal nothing more about the content in the network than the output of the matching algorithm that determines the routing. However, current solutions reveal much more than this by providing metadata describing the content as well nodes' interests in the clear.

We propose a *privacy-enhanced* information-centric networking solution for publish-subscribe systems. We employ cryptographic techniques, such as multi-authority attribute-based encryption (MA-ABE) [18], in order to provide the publishers of content with access control for their data. We leverage similar mechanisms to provide flexible protection of the metadata in the system as well. While the guarantees that we obtain for the privacy of the metadata do not achieve the ideal goal posed above, they provide a reasonable trade-off between privacy protection and usable efficiency for the system, given the current state of cryptographic tools.

Content sharing at the mobile edge is poised to rise dramatically with the recent introduction of Android Beam [5] and iOS AirDrop [3]. Consequently, we focus on the mobile ad hoc network (MANET) setting for our ICN architecture, where parties can join and leave the system arbitrarily and we cannot rely on persistent communication channels between any parties in the network. Such a setting is relevant to many real world scenarios for ICN applications. For example, when emergency responders must communicate at the scene of a disaster, conference attendees create virtual groups to share research data, or sports fans share multimedia content at an event venue. However, the MANET setting makes our task for providing privacy guarantees much harder.

Even advanced cryptographic tools such as encrypted search schemes [8, 13] and functional encryption [9, 16] do not suffice for achieving our ideal goal. We do not pursue extensions of these tools in order to overcome existing deficiencies since this will result in high computational overhead that would hinder the usability of our system. Instead we

*While visiting SRI.

ACSAC '15, December 07 - 11, 2015, Los Angeles, CA, USA

ISBN 978-1-4503-3682-6/15/12...$15.00

DOI: http://dx.doi.org/10.1145/2818000.2818001

provide a mechanism that allows each participant to declaratively scope the nodes in the network that can serve as brokers on its behalf. The selected brokers see only hashed versions of the metadata, which is enough for them to be able to run a matching algorithm. Our system further provides data integrity for the content and metadata published in the network by applying cryptographic signatures on the data objects and providing certificates to the participants in the network.

A major question for any system based on cryptographic primitives that require private credentials is how to bootstrap trust in the system and distribute corresponding credentials. A single central trusted authority in a dynamically changing system with multiple parties is not a reasonable assumption. This is why we built a solution that distributes trust across multiple authorities and use cryptographic primitives designed for such a decentralized trust model. We do not assume that the participants in the system already have their credentials in advance. Instead, we develop credential distribution protocols assuming only short secrets shared out-of-band between parties and their trusted authorities.

The system we describe is an extension of the Haggle framework [21] that was originally designed for opportunistic networking. Its modular architecture supports development and integration of a wide range of underlying protocols, routing algorithms, caching schemes, and security designs. The solution we created for ensuring the authenticity and confidentiality of content and queries is agnostic to the other aspects of the system. It can be composed regardless of the routing and caching schemes used. We have evaluated the overhead that our security solution adds and report the results here.

We describe our goals in Section 2 before providing an overview of our approach in Section 3. Section 4 outlines our general security architecture, which can be used by most ICNs. Section 5 describes our specific design, including the protocols used. Section 6 explains the changes we made to a mature ICN to implement our concrete design. Section 7 reports on the overhead from security when it is added to a real world Android deployment. We describe our contribution in relation to previous work in Section 8 and conclude with its highlights in Section 9.

2. GOALS

ICNs route content based on an associated description (such as an identifier, a name, or tags) and the interests of nodes in the network. ICN routing algorithms thus depend on the ability to match descriptions of published data and the interests advertised by other nodes. In existing ICN solutions, requests for content as well as content descriptions are visible to any node in the network, which introduces a substantial privacy problem.

The goal of our work is to mitigate the privacy concerns of publishers that tag content and subscribers that request this content in an ICN. The problem of ensuring data confidentiality is addressed in traditional packet-based networks by using encryption for all data sent between a specific source and destination [24]. We note that we cannot hope to achieve the same level of privacy in the context of an ICN because this will render impossible most ICN routing functionality. Thus, our ideal goal will be a solution that hides everything about the published content and subscribers' requests except the routing decisions.

Communication networks allow data to be exchanged by multiple parties. To realize the utility of the network, we need to provide a means for users to access the content that they are authorized to receive while protecting the data from everyone else in the network. In traditional networking settings this problem can be solved using public key encryption. Since the intended recipient in known to the publisher, the sender can encrypt the content with the receiver's public key. In the ICN setting, however, a particular piece of content may have multiple recipients. Therefore, we need a more flexible mechanism to enforce access control at the point of encryption, before the potential receivers are known.

Another important requirement for ICNs, where content does not move directly from publishers to subscribers, is to guarantee data integrity. An ICN security solution should let subscribers verify the origin of the content that they receive as well as the integrity of the content. Together these will address the threat of maliciously injected data in the network, by binding the content to its publisher and guaranteeing non-repudiation.

Many of the nodes in a mobile ad hoc ICN may not have direct or stable connections. This is why we need routing algorithms that only rely on credentials that each node receives when it joins the network. In particular, routing should not require direct point-to-point communication between publishers and subscribers. This rules out solutions where a publisher chooses a secret key to compute the tags for its content and the same secret key is required to generate the interest metadata that can be matched against the content tags.

Our ICN's security goals have the following requirements:
- *Content integrity* – any receiver can verify the source and integrity of the obtained content.
- *Content confidentiality* – publishers can limit access to the content that they publish.
- *Metadata confidentiality* – minimize the exposure of the contents' descriptive metadata and nodes' interest.
- *Intermediated operation* – data exchange should not require direct interaction between publishers and subscribers.

3. OVERVIEW

We propose a security solution for ICN publish-subscribe systems. It addresses the requirements described in Section 2 by implementing the following extensions.

Data integrity. We employ cryptographic signatures [22] to provide data integrity for the content sent in the network. Each node signs content that it publishes and provides a certificate from an authority that binds the node's identity to a signature verification key. This allows the subscriber to check the authenticity and integrity of the content they receive.

Since we are dealing with a dynamic network structure where participants join and disconnect at an arbitrary rate, and there may not be time for nodes to get certified by a widely trusted authority in the network within this short period, we introduce a trust chaining approach that expands the set of nodes that trust some data as it gets forwarded in the network.

Data confidentiality. We use attribute-based encryption (ABE) [23] as a mechanism for protecting the content in a setting where the potential receivers are not known at the time of publication. The ciphertext-policy attribute-based encryption [7] primitive embeds an access policy directly

into each ciphertext, and associates a set of attributes with each decryption key. A key can decrypt a ciphertext if and only if its attributes satisfy the encryption policy. Nodes receive keys containing appropriate attributes (such as their organization, position, or role) from authorities. Thus, each publisher can encrypt each piece of content with a different access control policy that limits the set of nodes that can decrypt and learn the content based on their attributes.

Since we are dealing with a setting where it is difficult to agree on a single trusted authority that issues attributes for all the users, we use multi-authority attribute-based encryption (MA-ABE) [18] that decentralizes the trust by having several independent authorities that can issue decryption keys corresponding to different attributes. This allows maximum flexibility for the publisher.

Metadata confidentiality. The routing decisions in our ICN solution are made based on metadata, which consists of the tags describing the published data content and the interests declared by subscribers. The routing mechanism of the system aims to move content towards subscribers interested in it. This is achieved through hop-by-hop matching of content tags and subscribers' interests. These matching decisions are an inherent privacy leak, but are necessary for the functioning of the system. While there are cryptographic techniques [9, 16] that can be used to minimize this leakage at the cost of substantial computational overhead, we adopt an alternative approach that achieves a trade-off between privacy and efficiency. We allow publishers and subscribers to scope the nodes that can make routing decisions for their data by scoping the access to the associated metadata using MA-ABE in a manner similar to the way it is used for content encryption.

4. ARCHITECTURE

We present the architecture for our ICN solution. It aims to accommodate the dynamic nature of our system where nodes join and leave the network at arbitrary rates. Such a network cannot guarantee a stable connection between any two nodes that can accommodate several rounds of communication.

4.1 Participants

We construct an ICN solution for a decentralized mobile publish-subscribe system. In this model there are four roles for nodes in the network: *publishers*, *subscribers*, *brokers* and *authorities*. Each node in the network can assume more than one of these roles in the different stages of the protocol. *Publisher* nodes add content to the network along with descriptive content tags that will be used for routing. *Subscriber* nodes periodically broadcast node descriptions that include their interests. These descriptions are used by other nodes to identify which content matches a remote node's interests. *Broker* nodes facilitate the data routing in the network. They are intermediate nodes that forward content between publishers and subscribers, based on matches between the content tags and node interests. In addition to these three roles for ICN participants, some of the nodes in the system will serve as *authorities* that provide the credentials necessary for the cryptographic protocols in the system.

4.2 Co-Certification

Similar to other systems, the root of trust in our ICN solution is an authority that issues credentials for the participants. However, since we consider a distributed setting where we cannot guarantee that there is a single central authority trusted by everyone, we have developed a system that allows multiple independent authorities so trust can be spread across them.

Every node in the network chooses its trusted authorities before it joins the network. The initial trust relationship between a node and its authorities is established out-of-band. In this phase the node shares a secret key with each of its trusted authorities. This key can be derived from any shared information, such as a PIN, passphrase, or biometric scans. The node will later use the shared key in the communication protocol to send secure credential requests to the corresponding authority. The authorities in our ICN solution can serve both the role of certificate authorities (that issue identity certificates to nodes) as well as authorization authorities (for the ABE scheme, granting encryption and decryption attributes to nodes).

The identities of nodes in the network are established through the certificates that they have obtained. A node will trust only content coming from parties in the network that it trusts – that is, parties that possess certificates issued by authorities trusted by the node. Since we may have multiple authorities in the network, we require at least one certification from a common trusted authority to establish a trust relationship between two parties. Thus, a publisher and subscriber must be *co-certified* – that is, they have at least one certification authority in common.

4.3 Content Access Control

We use MA-ABE to restrict access to content published in the network. Publishers encrypt content with an access policy before sending it to a remote node. The policy specifies who can access the data, or more specifically what combination of attributes are required for gaining access. Note that each authority has a unique identifier, which determines the set of attributes for which it can issue encryption and decryption keys. The name of each attribute links it to the authority that issued it.

With this approach each publisher can construct a policy for its data requiring that any party that can decrypt the ciphertext has to possess a set of attributes issued by authorities that the publisher trusts. The publisher needs to know only the attributes that it uses in its policy.

The distribution of the MA-ABE decryption keys is part of the ICN solution. Each node requests encryption and decryption attributes from the authorities that it has established trust relationship with. It uses the shared secret key with the authority to establish secure communication for these requests. Nodes may request encryption and decryption attributes either on demand as they need them to encrypt and decrypt content or with pre-provisioning – that is, requesting encryption and decryption attributes as soon as they join the network.

4.4 Metadata Access Control

In our ICN framework the routing of content is determined by matching content tags and nodes' interests. Subscribers propagate their node descriptions containing a list of interests to other nodes that they encounter. Similarly publishers share content with interested remote nodes that they encounter. Since nodes aggregate the interests of others', they can serve as brokers to route content from publishers to subscribers. In order to perform its task, a broker must able to

check for matches between content tags and node interests. From the point of view of maintaining nodes' privacy, we want to limit the information that brokers can learn about the content and node descriptions that they receive. Ideally, the brokers would learn just the outcome of the match and nothing more.

The ideal solution for handling the metadata is to have it encrypted with a scheme that allows the computation of a matching function on the ciphertexts of tags and interests. This would not allow the brokers to learn anything more about the encrypted tags and interests. Cryptographic techniques such as functional encryption [9, 16] or searchable encryption [8, 13] are potential candidates for a solution. Functional encryption provides the ability to generate decryption keys that reveal only the evaluation of a particular function on the encrypted data, rather than the whole plaintext. However, we chose not to pursue a solution based on this since the only known construction of this primitive [16] is extremely computationally expensive, which is untenable due to power and processing limitations of mobile devices.

Searchable encryption allows ciphertext to be queried without performing decryption. It provides functionality for generating search tokens for keywords. The tokens can then be used to test whether a ciphertext contains the keywords. The issue with known constructions is that the search tokens need to be generated using the private parameters for the scheme. In our publish-subscribe setting this means that subscribers will either need to know the private parameters for the encryption used by the publishers (which does not satisfy our trust model), or will have to request search tokens from the publishers (which is also not viable since subscribers and publishers may not be able to communicate directly).

We construct an alternative solution for the metadata in the system that does not achieve the goal of the ideal from a privacy point of view, but it provides a reasonable tradeoff between confidentiality and efficiency guarantees. We replace the content tags and the subscribers' interests with their hashes using a single function that is fixed for the whole system. The hash values of the metadata hide the exact content but reveals equality patterns across content tags and subscribers' interests. In particular, this allows an adversary to compare the same tags on different pieces of content and in different nodes' interests.

In order to mitigate the additional leakage from the metadata represented as hashes, we allow publishers and subscribers to restrict the nodes that can act as brokers for their content and interests. We realize this functionality using MA-ABE again. A publisher encrypts content tags under a policy that specifies which nodes are allowed to serve as brokers for the data. Similarly, subscribers use MA-ABE to encrypt their interests with a policy that specifies which nodes can serve as brokers on their behalf. Nodes that receive published content and interests can check for matches if and only if they are in the sets of authorized brokers for both the publisher and the subscriber.

5. DESIGN

We designed an access control architecture suited for publish-subscribe ICNs. Our goal was to minimize the assumptions about pre-shared key material between participants. In contrast to many systems, we also implement the protocols needed for credential distribution. Finally, we provide simple flexible primitives for applications to specify how to secure the actual distribution of their content.

Data Forwarding. The ICN forms an overlay and is agnostic to the underlying network transport – that is, node to node communication can use any protocol, such as UDP, TCP, or even UDP broadcast, over WiFi, Bluetooth, or other radio types. Since publishers and subscribers may not be able to communicate directly, a forwarding mechanism is needed. Subscribers flood their node descriptions to their neighbors. Data objects are propagated towards nodes with interests that match the content tags. We note that the nodes that receive forwarded published data objects are not necessarily subscribers. They may also be intermediate brokers that have propagated node descriptions. The choice of whether to cache a data object at a node depends on the application of a local utility function to the content tags, and the state of the local cache.

Cryptographic Primitives. Our constructions use multi-authority attribute-based encryption (MA-ABE), which consists of the following algorithms: $\mathsf{Setup}_{\mathsf{ABE}}(1^\lambda)$, which generates constants for the scheme using security parameter λ; $\mathsf{Encrypt}_{\mathsf{ABE}}(\mathcal{P}, m)$, which encrypts a message m under a policy $\mathcal{P}$; $\mathsf{Decrypt}_{\mathsf{ABE}}(\mathsf{ct}, \mathsf{SK}_{\alpha_1,\ldots,\alpha_n})$, which decrypts a ciphertext ct if and only if the attributes $\alpha_1, \ldots, \alpha_n$ of the decryption key satisfy the encryption policy $\mathcal{P}$ used to produce ct.

We also use a signature scheme which has the following algorithms: $\mathsf{Gen}(1^\lambda)$, which generates private signing key $\mathsf{SK}_{\mathsf{sign}}$ and public verification key $\mathsf{VK}_{\mathsf{sign}}$; $\mathsf{Sign}(\mathsf{SK}_{\mathsf{sign}}, m)$, which produces signature σ for a message m; and $\mathsf{Verify}(\mathsf{VK}_{\mathsf{sign}}, \sigma, m)$, which verifies the signature σ for m. The signature scheme guarantees that no party that does not have the secret signing key can produce a valid signature that can be verified with the corresponding public verification key.

5.1 Data Objects

Applications publish and receive data through our communication protocols in units called *data objects*. They have the following structure:

$$\{\ \mathsf{MetaData}\ \}[\ \mathsf{Content}\ ]$$

which consists of two parts – the metadata that is used for forwarding the data object, and the content that is used by the receiver.

The metadata of a data object consists of several tags that can serve different purposes. One of the tags specifies the type of the data objects. The main types of data objects are published data, node descriptions containing the interests of nodes, and messages used in credential distribution protocols.

Tags in the metadata are used by the forwarding algorithms. These can be tags describing the content of a published data object or a subscriber's interests in a node description. We note that tags and interests are not unique identifiers. Different data objects can share some of the tags that describe their content. Further, subscribers may have intersecting sets of interests in their node descriptions.

We can also use the metadata to emulate point-to-point communication between two parties. This is done by having the sender tag its messages with the identity of the receiver, while the receiver includes in its node description interests associated with its identity. The forwarding mechanism moves data objects from node to node based on matching data object tags to node interests, as described above. As a result, the data objects will be forwarded from the sender to the receiver, even if the connectivity is intermittent. Such

point-to-point communication will be used in the protocols for credential distribution, as described in Section 5.2.

Metadata Confidentiality. Routing requires that brokers must be able to compare tags and interests in order to decide whether to forward a data object. At the same time we want to minimize the information that a broker learns about the data object if it is not authorized to access the content.

As discussed in Section 4.4, we adopt a solution that offers a tradeoff between privacy and the efficiency of matching metadata. The metadata used for forwarding consists of interests, object type $\mathsf{tag}_{\mathsf{ObjType}}$, and other tags, denoted by $t_1, \ldots, t_n$. These are all hashed with a function $H()$:

$$\mathsf{MetaData} = (\mathsf{tag}_{\mathsf{ObjType}}, H(t_1), \ldots, H(t_n))$$

When a user node **U** published a data object, it can scope the set of nodes in the ICN that are allowed to serve as brokers for it. This is done by defining the chosen set with a broker access policy $\mathcal{P}_{\mathsf{MetaData}}$, and then encrypting the metadata with this policy using MA-ABE. The resulting data object has the form:

$$\{\ \mathsf{ct}_{\mathsf{MD}} = \mathsf{Encrypt}_{\mathsf{ABE}}(\mathcal{P}_{\mathsf{MetaData}}, \mathsf{MetaData})\ \} \quad [\ \mathsf{Content}\ ]$$

Brokers attempt to decrypt the metadata of all data objects that they receive. When a broker is able to decrypt the metadata of a data object, it checks the extent to which the hashed content tags $H(t_a)$ match the hashed interests $H(i_b)$ of each node description with metadata that it was able to decrypt. If there is a sufficient match between the content tags and a remote node's interests, the broker forwards the data object towards the subscriber.

Since nodes may not have credentials when they make security requests and receive responses, the data objects that contain such messages are handled differently. Their metadata is not encrypted with MA-ABE. Instead they are routed point-to-point, revealing only the source and destination identifiers. No other information is present in the metadata, as described further in Section 5.2.

Content Confidentiality and Integrity. A user node **U** can limit which nodes in the ICN should be able to access the content they publish. This is achieved in a manner similar to the way brokers are selected. The publisher defines the set of authorized subscribers by specifying an access policy $\mathcal{P}_{\mathsf{Content}}$. This policy can be (and is likely to be) more restrictive than the broker access policy. The publishing node uses MA-ABE to encrypt the content with the specified access policy.

In order to guarantee the integrity of the content and the metadata, the publisher appends a signature. A publishing node thus emits data objects with the form:

$$\{\mathsf{ct}_{\mathsf{MD}} = \mathsf{Encrypt}_{\mathsf{ABE}}(\mathcal{P}_{\mathsf{MetaData}}, \mathsf{MetaData})\}$$
$$[\mathsf{ct}_{\mathsf{Content}} = \mathsf{Encrypt}_{\mathsf{ABE}}(\mathcal{P}_{\mathsf{Content}}, \mathsf{Content})]$$
$$\sigma = \mathsf{Sign}(\mathsf{SK}_{sign,\mathbf{U}}(\mathsf{ct}_{\mathsf{MD}} || \mathsf{ct}_{\mathsf{Content}})$$

Data objects used to convey security requests and responses to and from authorities are not encrypted with MA-ABE since nodes may not have sufficient credentials to do so. Instead, the integrity of the content is assured using the secret key shared by the user and authority and a MAC – that is, a keyed hash function. Similarly, the confidentiality of the content is ensured using symmetric encryption with the shared secret. This is described further in Section 5.2. In practice a key derivation function is used to obtain different keys for hashing and encryption from a single shared secret.

We note that for efficiency purposes, we encrypt a short symmetric key K with the MA-ABE scheme and then encrypt the actual message M using symmetric encryption with key K – that is, whenever we use $\mathsf{Encrypt}_{\mathsf{ABE}}(\mathcal{P}, M)$, the actual implementation is of the form $\mathsf{Encrypt}_{\mathsf{ABE}}(\mathcal{P}, K)$, $\mathsf{Encrypt}^{\mathsf{sym}}_{K}(M)$. This holds for the encryption of both metadata and content.

5.2 Bootstrapping Credentials

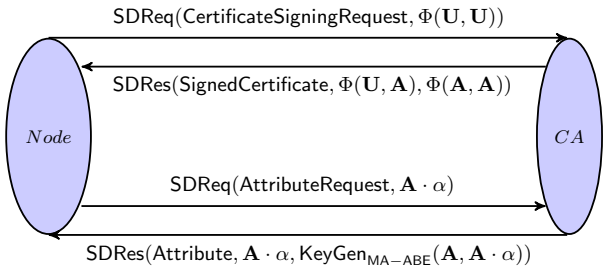

Figure 1: *Security data requests* for certification or attributes originate from user nodes. They are handled by authorities, which send *security data responses* with signed certificates or cryptographic keys corresponding to attributes.

Since the setting that we aim to address does not allow for centralized certification authorities, we adopt an approach where any party in the network can be reconfigured to serve as a certification authority. The only assumption about pre-shared keys in our system is the following: nodes in the system will have shared private keys with authorities from which they can request access credentials.

When a node **A** is configured in the authority mode of operation, it publishes a node description that includes an interest $\mathsf{tag}_{\mathsf{request},\mathbf{A}}$, which specifies that it wants to receive all credential requests for itself. A user **U** who has a shared key s with the authority **A** can generate a credential request M to the authority as follows: **U** derives a key K for a symmetric key encryption scheme $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$ and a key K' for a message authentication code MAC [6]. It then generates $\mathsf{SDReq}(M)$, a security data request from user **U** to authority **A** with message M, defined as follows:

$$\{\mathsf{tag}_{\mathsf{request},\mathbf{A}}\ \}$$
$$[\mathsf{ct} = \mathsf{Encrypt}_{\mathsf{K}}(\mathbf{U}, \mathbf{A}, \mathsf{tag}_{\mathsf{request,A}}, M), \mathsf{h} = \mathsf{MAC}_{\mathsf{K}'}(\mathsf{ct})]$$

At the same time, **U** floods its node description that includes an interest $\mathsf{tag}_{\mathsf{response},\mathbf{U}}$, which specifies that the user wants to receive data objects with responses to its credential requests. When the authority **A** receives the request, it publishes a data object $\mathsf{SDRes(M)}$, a security data response from authority **A** to user **U** with message M', defined as follows:

$$\{\mathsf{tag}_{\mathsf{response},\mathbf{U}}\}$$
$$[\mathsf{ct}' = \mathsf{Encrypt}_{\mathsf{K}}(\mathbf{A}, \mathbf{U}, \mathsf{tag}_{\mathsf{response}}, M), \mathsf{h}' = \mathsf{MAC}_{\mathsf{K}'}(\mathsf{ct}')]$$

Nodes use security data requests to obtain both certificates and ABE encryption and decryption attributes from the respective authorities. The security properties of $\mathcal{PKE}$ and MAC guarantee that nobody can issue requests on behalf of another user or receive the credentials of another user. They also ensure that tampering with the request and response messages will be detected.

5.3 Building Trust

To avoid attacks where malicious nodes masquerade as other nodes in the system, we need to establish cryptographic identities for the nodes in the network and then build trust in those. We use certificates as a means for node identification, where a node certificate contains a signature verification key that can be used to bind particular content to that node. There are two different types of certificates: self-signed certificates and certificates signed by a certificate authority (CA). The basic trust premise in our system is that a node trusts another node if and only if there is at least one authority that has certified both of them.

We do not assume that nodes in the system have received their certificates from trusted authorities in advance. When a node $\mathbf{U}$ joins the network, it has only a self-signed certificate $\Phi(\mathbf{U}, \mathbf{U})$. It sends to any trusted authority $\mathbf{A}$, a security data request $\mathsf{SDReq(M)}$ with message:

$$M = \{\mathsf{CertificateSigningRequest}, \Phi(\mathbf{U}, \mathbf{U})\}$$

The authority sends back a signed certificate for the user $\Phi(\mathbf{U}, \mathbf{A})$ in a security data response $\mathsf{SDRes(M)}$ with message:

$$M = \{\mathsf{SignedCertificate}, \Phi(\mathbf{U}, \mathbf{A}), \Phi(\mathbf{A}, \mathbf{A})\}$$

When two nodes encounter each other for the first time, they establish a connection and exchange their certificates. A remote user $\mathbf{U}'$ is added to the set of nodes trusted by node $\mathbf{U}$ if and only if it presents certificates that pass the following verification check:

$$\mathsf{Verify}(\Phi(\mathbf{U}', \mathbf{U}'), \Phi(\mathbf{U}', \mathbf{A})) \wedge \mathsf{Verify}(\Phi(\mathbf{U}', \mathbf{A}), \Phi(\mathbf{A}, \mathbf{A}))$$

where $\mathbf{A}$ is an authority trusted by the node $\mathbf{U}$.

A node trusts the content of a data object that it receives only if it is signed by a node that it trusts.

Content Re-signing.

When a node $\mathbf{U_i}$ forwards data objects:

$$\{\mathsf{ct_{MetaData}}\}[\mathsf{ct_{Content}}]\sigma_{\mathbf{U_{i-1}}} = \mathsf{Sign}(\mathsf{SK}_{sign,\mathbf{U_{i-1}}}, \mathsf{ct_{MetaData}}||\mathsf{ct_{Content}})$$

that are signed by another user $\mathbf{U_{i-1}}$ that is trusted by $\mathbf{U_i}$, it re-signs the data object with its own signature σ:

$$\{\mathsf{ct_{MetaData}}\}[\mathsf{ct_{Content}}]\sigma_{\mathbf{U_i}} = \mathsf{Sign}(\mathsf{SK}_{sign,\mathbf{U_i}}, \mathsf{ct_{MetaData}}||\mathsf{ct_{Content}})$$

This allows published data objects to transition across different trust domains.

Signature Chaining.

As a data object moves across the network, its lineage is recorded in its metadata. This can be used later to verify the source and the forwarding path of a particular data object. For this purpose we create *signature chains* $\langle c_0, \cdots, c_n\rangle$ which are computed as follows:

$$c_0 = \mathsf{Sign}(\mathsf{SK}_{sign,\mathbf{U_0}}, \mathsf{ct_{MetaData}}||\mathsf{ct_{Content}})$$
$$c_i = \mathsf{Sign}(\mathsf{SK}_{sign,\mathbf{U_i}}, c_{i-1}) \quad \text{for } 1 \leq i \leq n$$

where $\mathbf{U}_0$ is the publisher of the data object and $\mathbf{U}_1, \ldots, \mathbf{U}_n$ are the nodes that forward the data object. Thus, the data object forwarded from $\mathbf{U}_i$ to $\mathbf{U}_{i+1}$ is of the form:

$$\{\mathsf{ct_{MetaData}}\}[\mathsf{ct_{Content}}]\sigma_{\mathbf{U_i}}\langle c_0, \cdots, c_i\rangle$$

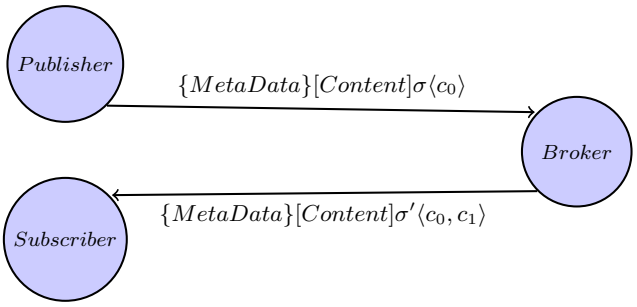

Figure 2: Signature chaining serves two purposes. First, it ensures that content flows through trusted paths from the publisher to the subscriber. Second, it provides subscribers with certified provenance.

5.4 Attribute Provisioning

We use MA-ABE to enforce access control over the content and metadata of data objects. Nodes obtain MA-ABE encryption and decryption credentials from the authorities that are responsible for the respective attributes. An authority $\mathbf{A}$ can issue credentials for attributes of the form $\mathbf{A} \cdot \alpha$. A node requests an attribute credential for $\mathbf{A} \cdot \alpha$ by sending a security data request $\mathsf{SDReq}(M)$ with message:

$$M = \{\mathsf{AttributeRequest}, \mathbf{A} \cdot \alpha\}$$

If the authority grants the request, it publishes a security data response $\mathsf{SDRes}(M')$ with message:

$$M' = \{\mathsf{Attribute}, \mathbf{A} \cdot \alpha, \mathsf{KeyGen}_{\mathsf{MA-ABE}}(\mathbf{A}, \mathbf{A} \cdot \alpha)\}$$

6. IMPLEMENTATION

The functionality of the security framework is designed to compose with the other features of the underlying ICN [25, 26]. Since content encryption occurs at file granularity, it is transparent to the routing, caching, and network coding components (that operate on data objects corresponding to encrypted files instead of the original ones). Control metadata is not encrypted, allowing delay-tolerant routing, utility-based caching, and context-aware network coding to operate unchanged. Since the hashing of tags and interests supports equality checks, threshold matching of content to interests continues unmodified at nodes authorized to serve as brokers. The inability of unauthorized nodes to broker matches results in the only change in system behavior – that is, since content must flow through a subset of all possible connections, routing robustness may be sacrificed.

We implemented our solution by extending the Haggle framework, as shown in Figure 3. The Haggle framework consists of two codebases – the Haggle kernel and a library `libhaggle` that ICN applications use. Each node on the network runs an instance of the Haggle kernel. Applications can publish data objects and express their interests by communicating with the local Haggle kernel. This is done via the shared library that communicates with the kernel through a socket. The kernel is responsible for transferring content between nodes, matching application interests to available content, and delivering content to applications.

The kernel on each node is configured with default security parameters (such as shared secrets, authority details, and access control lists) in its configuration file. The kernel is

responsible for handling key management and distribution of MA-ABE attributes, as well as handling trust management and verification of certificates and signatures. All of the security operations are transparent to applications. To allow publishers to restrict access to content, the kernel checks for *Access* metadata attached to data objects being published. This metadata is known as a *content attribute* in Haggle parlance, and differ from the MA-ABE attributes that serve as credentials. If the *Access* attribute is present, its value is used as the access policy for the data object. The content will then be encrypted with the access policy specified. At each receiving node, the kernel will check the encryption status of each data object, perform decryption if required, and deliver the plaintext data object to any local subscribing applications. Thus, access control is transparent to applications and publishers only need to add an access policy tag (using existing calls already present in libhaggle). Subscribers will receive decrypted content with no application modifications.

The Haggle kernel is an event-based architecture with multiple *managers* cooperating to provide various pieces of functionality. Each manager runs in its own thread and communicates with other managers by sending events through a shared queue. Each manager can also instantiate its own background worker threads for computationally-intensive operations. The `ApplicationManager` handles communication with local publishing and subscribing applications. The `DataManager` is responsible for data object storage, caching, and purging. The `NodeManager` manages remote node interests and matching. The `ForwardingManager` is responsible for routing decisions. The `ProtocolManager` handles low-level protocol communications between nodes. The `SecurityManager` is responsible for security. The layer-less architecture allowed us to make most of our changes in the SecurityManager, with minor modifications in other managers.

Access Control. Using access control is transparent to end-user applications. Publishers only need to add an access policy tag. Subscribers receive decrypted content without any changes to applications.

Figure 3 illustrates the flow of a data object during publication. In step 1, a local application publishes a data object with an access policy using `libhaggle`. The `ApplicationManager` receives this, and notifies the `DataManager` (2). It is then inserted into the `DataStore` (3) and other managers are notified about this through the *Event Queue* (4). The `SecurityManager` checks for the presence of an access policy (5). If one is present, and the node has all the MA-ABE attributes needed to perform the encryption, an asynchronous task is created to Encrypt the content. Otherwise, a SecurityDataRequest is issued, and the encryption is performed upon receipt of the necessary attributes. During this encryption task, a symmetric encryption key is generated, and the content is encrypted with this key. This key is encrypted using MA-ABE and referred to as a *capability*. A new data object is created with the same metadata, the capability, and the encrypted content. This is then inserted in the *Event Queue* (6), where the `NodeManager` looks for remote nodes that are interested in this content (7). If any remote nodes are interested (8), `ForwardingManager` then makes routing decisions and asks the `ProtocolManager` to send the data using a link layer protocol (9). The *Protocol Manager* checks if the data object is destined for an application, in which case a plaintext version of the content is sent. If the data object is going to a remote node, the ciphertext

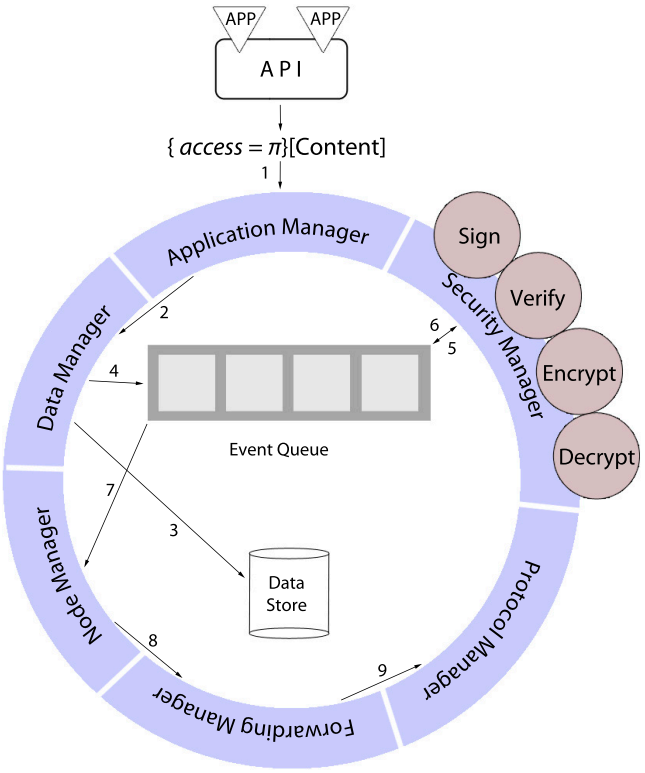

Figure 3: Haggle provides an event-based ICN kernel. Each feature is implemented in a manager running in a separate thread. Applications can scope access to published content simply by tagging it with an `Access` attribute (with π denoting the policy here).

version of the content is used.

A similar process happens at the receiving end. If an encrypted data object arrives and an application is interested in it, the *SecurityManager* enqueues the data object for decryption. A check is performed to see whether all the necessary MA-ABE decryption attributes are present. If not, a SecurityDataRequest is issued. After the relevant SecurityDataResponse is received, the symmetric encryption key is recovered from the capability via a Decrypt operation. The key is then used to symmetrically decrypt the content. A new data object is created with the same metadata, but with the decrypted content and without the capability tag. This plaintext is then made available to the interested application.

Integrity. The *SecurityManager* intercepts incoming node descriptions and extracts certificates present during the exchange of routing metadata. These certificates are verified using the mechanisms described in Section 5.3. When data objects are received from applications, the *SecurityManager* automatically signs them and updates the signature chain. When data objects are received from remote nodes, the *SecurityManager* performs signature verification and drops them if verification fails.

When a data object is to be sent, the SecurityManager checks whether a signature is present. If no signature is present, the sending is blocked, and it is enqueued for signing. The SecurityHelper will compute a signature and re-queue the data object for sending. When a data object is received,

Overhead	Time (s)		Space (KB)
	Linux	Android	
No security	0.107	0.793	51,794
Signatures	0.107	0.888	51,873
Encryption with Cached Access Policies	0.145	0.924	52,194
Encryption with Uncached Access Policies	0.422	2.391	52,194

Table 1: Time and space overhead of encryption and signing.

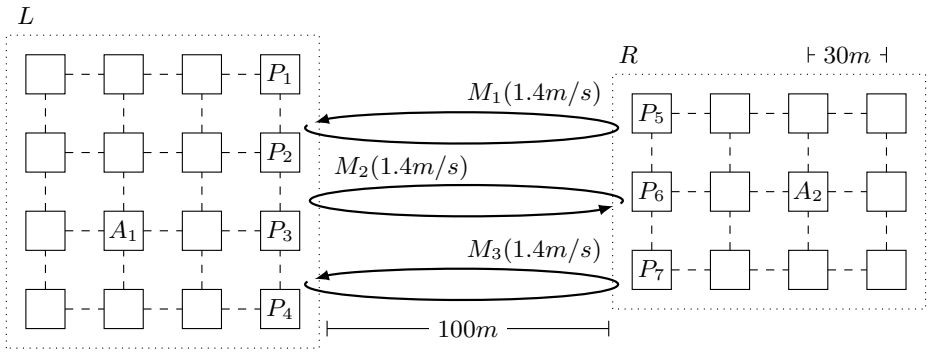

Figure 4: ***There are two groups of nodes arranged in a grid formation. Three data mules move back and forth to transfer data, staying for 60 seconds each time.***

the SecurityManager looks up the relevant node certificate in a store of verified certificates. If one is present, then a signature verification task is enqueued. The SecurityHelper will perform the signature verification and pass the data object on to other managers. If no verified certificate is present, or the verification fails, the data object is dropped.

7. EVALUATION

Our security design uses the computationally expensive MA-ABE primitive. Consequently, the result of transformations that use it are cached. We expect that this will ameliorate the impact on system performance with typical workloads. This was confirmed through empirical studies.

We evaluated the implementation in two environments. The first was a testbed of 30 Samsung Nexus S devices running Android Gingerbread (2.3). The second environment consisted of Linux containers, spawned by the network emulation framework, NRL's CORE 4.3 and EMANE 0.7.3 [2].

Micro-Benchmark. We ran micro-benchmarks to measure the speed of cryptographic primitives used. On Linux, we also ran a version of the tests with the CPU limited to mimic the resource constraints of mobile Android devices. We used a simple two node test, with the user node publishing data objects, and the authority node subscribing to data objects. 512 KB files, 2048-bit RSA signatures, and access policies with eight MA-ABE attributes were used. 101 data objects were published. The results are shown in Table 1.

We can see that the time overhead added by signatures is minimal. Encryption has a more noticeable temporal overhead. However, when the capabilities have already been cached (as is the common case in practice), the overhead is much lower since the expensive MA-ABE operations do not have to be performed. A small amount of bandwidth overhead is introduced by the use of signatures since the node descriptions contain certificates. The encryption bandwidth overhead is from the capabilities that must be sent with each data object, and the requests for MA-ABE encryption and decryption attributes.

Macro-Benchmark. To obtain repeatable tests, we use emulation to model real-world mobile performance on a Linux server. The same source code was cross-compiled to ARM code that ran on Android, and x86 binaries that ran in Linux containers for the emulation. We used the CPULimit tool with a threshold of 30% to slow the performance of Haggle processes to match the performance of our Android devices, as reported in Table 1. We used CORE and EMANE for resource isolation, node mobility modeling, and network emulation. Haggle instances were started in virtual nodes that moved in the pattern described in the scenario below.

Scenario. The scenario setting is illustrated in Figure 4. The events run for 900 seconds with two authority nodes, A_1 and A_2, and seven publishers, $P_1 \cdots P_7$. At 1, 5, 11, and 16 seconds, one node in each group publishes a data object with an access policy involving both authorities. All nodes subscribe to these data objects, so we expect 112 intra-group deliveries and 136 inter-group deliveries. Starting at time 21, there are seven intra-group data objects published (with an access policy involving only one authority), with a total of 100 expected deliveries.

Results. Our emulation results are presented in Figure 5. We plot the number of data objects received at each point in time. We measured this for five different configurations: no-security, signatures, signatures-static, encryption, and encryption-static. The static cases refer to settings where nodes were pre-configured with the appropriate certificates and MA-ABE attributes, so they did not need to make SecurityDataRequests. These settings are useful in scenarios where the network topology is known beforehand.

In all cases except encryption, publishing can start from time 1, as there is no need to wait for MA-ABE attributes. Thus the 212 intra-group deliveries can complete quickly.

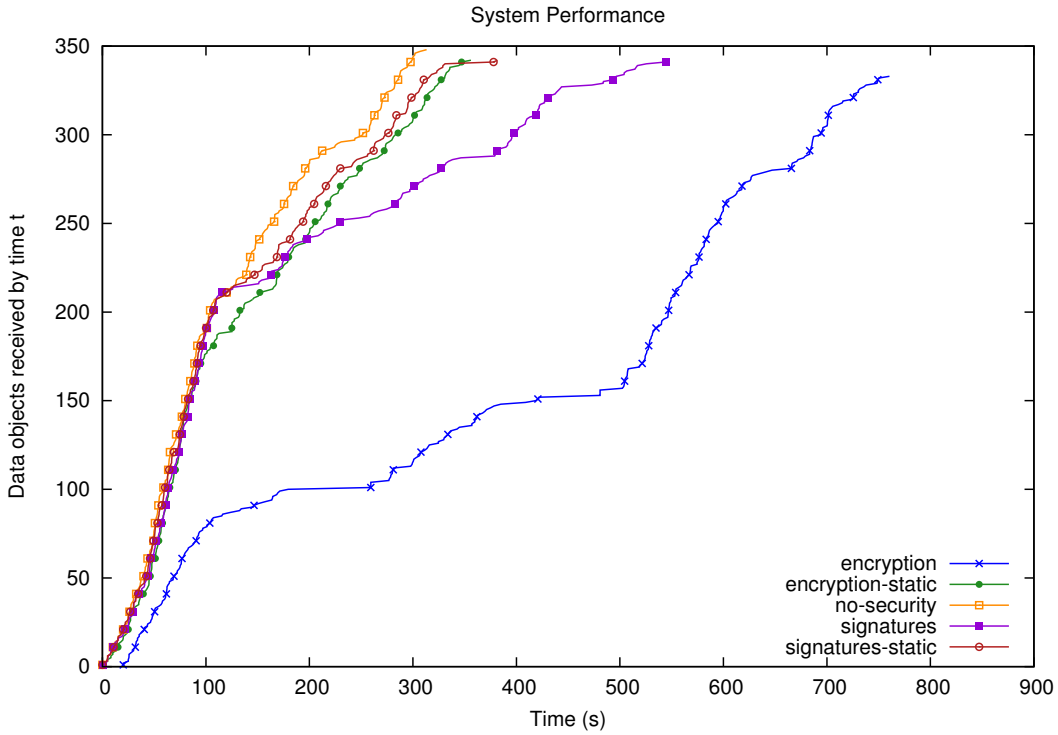

Figure 5: Data objects delivered over time.

The remaining 136 inter-group deliveries require the use of mules, so they take slightly longer to complete.

For encryption, there can be no publishing before time 21, due to the need for MA-ABE attributes. The initial 100 intra-group deliveries then complete gradually as publishers get the keys they need from the local authority and then so do the subscribers. There are no further deliveries until the 250 second point, by when mules have arrived with keys from authorities located in the other group. Publishing then begins, and intra-group deliveries commence. The mules make some more trips and by the 500 second mark the rate of delivery jumps as inter-group deliveries complete.

We can see that signatures and encryption affect delivery latency. This is mostly due to the SecurityDataRequests and SecurityDataResponses that have to travel through the mules. With static provisioning, this overhead is eliminated, and the only extra latency is due to the time and space overhead seen in the micro benchmarks in Table 1.

8. RELATED WORK

In the 1980s and 1990s, the Internet was primarily used for point-to-point communication. After the advent of the Web, it was increasingly used for content distribution. In 2000, TRIAD described the use of a *content layer* to handle routing, caching, and transformation [12]. Directory service responses were to be authenticated with digital signatures, while communication between endpoints was to be protected at the network layer, with an analog of IPSec. OceanStore proposed distributed access control using content encryption [20]. The assurance provided by our framework subsumes this. In particular, the integrity of metadata (used for content search) is protected with signatures, and access to content is limited via encryption. By 2009, CCN [11] and its derivatives [19] had introduced *flow balance*, with interests being consumed by data flowing back to subscribers. This provides resistance to denial of service attacks. In future work, we plan to leverage strong node identities to explore reputation tracking and anomaly detection for combating similar threats.

Our work differs from previous research on securing ICNs in several respects. We do not assume the existence of an external infrastructure for providing cryptographic keys [19]. Instead each node defines its own identity (via self-generated signing and verification keys) that is then attested by nodes that it deems to be authorities. Similarly, we do not rely on authorization services for access policies to be resolved [15]. We use a cryptographic primitive that supports flexible specification of access control policies during encryption. Resolution can thus occur as soon as content is available, rather than when a policy server becomes reachable. Moreover, we provide the first ICN implementation of cryptographic access control that supports the use of multiple, independent authorities. We enhance the privacy of individual interests, rather than just preventing surveillance at scale [4]. We do not rely on self-certifying names for integrity assurance [17]. By decoupling naming from content identification, we allow flexible resolution of interests to content. At the same time we provide strong assurance that a piece of content came from the claimed publisher. In particular, we do not require applications to verify the authenticity of content [14]. Finally, we provide the first design and implementation of certified content lineage, allowing subscribers to identify and verify both the origin as well as the path through which content arrived from a publisher.

9. CONCLUSION

We described the design, implementation, and evaluation of a security architecture for ensuring the integrity and confidentiality of content, as well as the metadata that describes it. The metadata is utilized by brokers that mediate between publishers and subscribers, based on the match between content tags and node interests. The protection of this metadata is necessary to mitigate attacks on the privacy of publishers and subscribers. It is worth noting that this problem is particularly challenging when every node can serve as a publisher, broker, and subscriber.

We utilize a recent cryptographic primitive, *multi-authority attribute-based encryption*, to reduce the access control problem to credential management. Nodes are issued credentials with their attributes (such as their name, organization, or location) from one or more authorities. Access policies can be declared as any Boolean combination of node attributes. This allows access to content and its descriptive tags to be scoped by publishers. Similarly, subscribers can cryptographically limit the nodes that can access their interests.

Our work is an extension of the Haggle opportunistic networking system. We report on the overhead added for ensuring the integrity and confidentiality of content on both Linux desktops and Android smartphones. We found that integrity protection does not add noticeable overhead. Typical confidentiality protection (where access policies have been previously defined) adds modest overhead. Publication with new access policies adds latency to the transfer of content. However, this does not reduce the quantity of data objects that get through in a real scenario with mobile nodes.

Acknowledgements. We thank Mark-Oliver Stehr for his insights on extending Haggle. This material is based upon work partially supported by the National Science Foundation under Grant IIS-1116414. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

10. REFERENCES

[1] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Borje Ohlman, A survey of information-centric networking, *IEEE Communications Magazine*, Vol. 50(7), 2012.

[2] Jeff Ahrenholz, Comparison of CORE Network Emulation Platforms, *29th IEEE Military Communications Conference*, 2010.

[3] Apple AirDrop, https://www.apple.com/ios/features/#airdrop

[4] Somaya Arianfar, Teemu Koponen, Barath Raghavan, and Scott Shenker, On preserving privacy in content-oriented networks, *ACM SIGCOMM Workshop on Information-Centric Networking*, 2011.

[5] Android Beam, http://developer.android.com/guide/topics/connectivity/nfc/

[6] Mihir Bellare, Ran Canetti, and Hugo Krawczyk, Keying hash functions for message authentication, *16th Annual International Cryptology Conference on Advances in Cryptology*, 1996.

[7] John Bethencourt, Amit Sahai, and Brent Waters, Ciphertext-policy attribute-based encryption, *28th IEEE Symposium on Security and Privacy*, 2006.

[8] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano, Public key encryption with keyword search, *23rd International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2004.

[9] Dan Boneh, Amit Sahai, and Brent Waters, Functional encryption: Definitions and challenges, *8th Theory of Cryptography Conference*, Springer, 2011.

[10] DARPA CBMEN, http://www.darpa.mil/NewsEvents/Releases/2013/08/21.aspx

[11] PARC CCN, https://www.parc.com/services/focus-area/content-centric-networking/

[12] David Cheriton and Mark Gritter, TRIAD: A new next-generation Internet architecture, 2000.

[13] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky, Searchable symmetric encryption: Improved definitions and efficient constructions, *13th ACM Conference on Computer and Communications Security*, 2006.

[14] Seyed Kaveh Fayazbakhsh, Yin Lin, Amin Tootoonchian, Ali Ghodsi, Teemu Koponen, Bruce Maggs, K.C. Ng, Vyas Sekar, and Scott Shenker, Less pain, most of the gain: incrementally deployable ICN, *ACM SIGCOMM Conference*, 2013.

[15] Nikos Fotiou, Giannis Marias, and George Polyzos, Access control enforcement delegation for information-centric networking architectures, *2nd ACM Workshop on Information-Centric Networking*, 2012.

[16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters, Candidate indistinguishability obfuscation and functional encryption for all circuits, *54th IEEE Symposium on Foundations of Computer Science*, 2013.

[17] Ali Ghodsi, Teemu Koponen, Jarno Rajahalme, Pasi Sarolahti, and Scott Shenker, Naming in content-oriented architectures, *1st ACM Workshop on Information-Centric Networking*, 2011.

[18] Allison Lewko and Brent Waters, Decentralizing attribute-based encryption, *30th International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2011.

[19] Van Jacobson, Diana Smetters, James Thornton, Michael Plass, Nicholas Briggs, and Rebecca Braynard, Networking named content, *5th International Conference on Emerging Networking Experiments and Technologies*, 2009.

[20] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao, OceanStore: An architecture for global-scale persistent storage, *9th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2000.

[21] Erik Nordstrom, Christian Rohner, and Per Gunningberg, Haggle: Opportunistic mobile content sharing using search, *Computer Communications*, Vol. 48, Elsevier, 2014.

[22] Ronald Rivest, Adi Shamir, Leonard Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, Vol. 21(2), 1978.

[23] Amit Sahai and Brent Waters, Fuzzy identity-based encryption, *24th International Conference on the Theory and Applications of Cryptographic Techniques*, 2005.

[24] Transport Layer Security, http://tools.ietf.org/html/rfc5246

[25] Samuel Wood, James Mathewson, Joshua Joy, Mark-Oliver Stehr, Minyoung Kim, Ashish Gehani, Mario Gerla, Hamid Sadjadpour, and J.J. Garcia-Luna-Aceves, ICEMAN: A system for efficient, robust and secure situational awareness at the network edge, *32nd IEEE Military Communications Conference*, 2013.

[26] Samuel Wood, James Mathewson, Joshua Joy, Mark-Oliver Stehr, Minyoung Kim, Ashish Gehani, Mario Gerla, Hamid Sadjadpour, and J.J. Garcia-Luna-Aceves, ICEMAN: A practical architecture for situational awareness at the network edge, *Logic, Rewriting, and Concurrency, Lecture Notes in Computer Science*, Vol. 9200, Springer, 2015.

[27] George Xylomenos, Christopher Ververidis, Vasilios Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos Katsaros, and George Polyzos, A survey of information-centric networking research, *IEEE Communications Surveys and Tutorials*, Vol. 16(2), 2014.