Access-Controlled In-Network Processing of Named Data

Claudio Marxer
University of Basel
Switzerland
claudio.marxer@unibas.ch

Christopher Scherb
University of Basel
Switzerland
christopher.scherb@unibas.ch

Christian Tschudin
University of Basel
Switzerland
christian.tschudin@unibas.ch

ABSTRACT

In content-based security, encrypted content as well as wrapped access keys are made freely available by an Information Centric Network: Only those clients which are able to unwrap the encryption key can access the protected content. In this paper we extend this model to computation chains where derived data (e.g. produced by a Named Function Network) also has to comply to the content-based security approach. A central problem to solve is the synchronized on-demand publishing of encrypted results and wrapped keys as well as defining the set of consumers which are authorized to access the derived data. In this paper we introduce "content-attendant policies" and report on a running prototype that demonstrates how to enforce data owner-defined access control policies despite fully decentralized and arbitrarily long computation chains.

CCS Concepts

•Networks → Network protocol design; •Security and privacy → *Security protocols;*

Keywords

information centric networking; named data networking; named function networking; in-network computations; security; privacy; access control

1. INTRODUCTION

In the last years, extensive research was carried out in the field of Information Centric Networking. Motivated by the attempt to hide network locations but name content directly, an alternative network model to state-of-the-art computer networks has emerged. All developments into this direction have in common that a data consumer utilizes a lookup-primitive to request location-independent named content from the network. Named function networking has been introduced as an extension enabling the network to serve named content which has never been disposed before. To be more specific, the lookup-primitive is enhanced to deal with content names bound to the result of an arbitrary computation on named content. To the consumer triggering a lookup, it is transparent if the content was already available in the network or produced on-demand.

Production of content out of content is performed in-network without necessarily involving the original publisher of the processed data. This decentralized approach has turned out to be a good design decision as long as the processed data is not access-controlled. However, no access control solutions exist for in-network processing of access controlled content. Therefore, in this paper we study in-network processing of access-controlled content and contribute a protocol to secure data of this kind. According to that protocol, content is accompanied by meta-information which enables result producing entities to implement access control pursuant to the policies of the processed (primary) data. Content production chains – producing results out of results ... out of published content – are feasible without involving the original publisher in the access control part of every computation step. In fact, data production is completely consumer-driven and executed in a decentralized fashion just by making use of content and meta-information as provided through the protocol.

To balance the privacy policies of the data publisher with the computational needs of the consumer, any user can successfully lookup exactly those computation results which he could also produce locally (just by making use of content he is authorized to access).

The remainder of this section contains an introduction to Information Centric Networking (ICN) and Named Function Networking (NFN). In Section 2 we discuss challenges related to access control and security in the context of ICN as well as NFN and elaborate access control principles to implement a system as envisioned above. We evaluate our proposal in Section 3 and act out a use case for clarification. In Section 4 and 5 we discuss our contribution and come up with a conclusion.

1.1 ICN - Information Centric Networking

State-of-the-art computer networks such as the current Internet are constructed in a host-centric manner. In contrast, the principle idea of *Information Centric Networking* (ICN) is to treat information as the pivotal entity [7]. The most obvious impact of this paradigm shift is that all content carried by the network is directly identified by a unique name while "traditional networking" is about uniquely named hosts. In the currently most widespread ICN derivatives a name consists of hierarchically structured components (e.g.

ICN'16 September 26-28, 2016, Kyoto, Japan

ACM ISBN 978-1-4503-4467-8/16/09.

DOI: http://dx.doi.org/10.1145/2984356.2984366

`/this/name/has/five/components`). Such an ICN only distinguishes between two packet types, namely *interests* and *content objects*. A content object carries certain content uniquely and immutably bound to a name. To request a certain content object, an interest packet containing its name is sent out to the network.

A forwarder who handles such interests mainly consist of a *content store* (CS), a *forwarding information base* (FIB) and a *pending interest table* (PIT). If an interest arrives on a certain interface, the requested content object is directly returned in case it is found in the content store. Otherwise the interest is further propagated by sending it to neighboring nodes. In this case a PIT entry is created to remember the name and the incoming interface of the interest. if there exists already a PIT entry for the given name, the interface is solely added to that entry because the interest is already further propagated. To avoid extensive interest flooding, only some interfaces are selected for propagation. This is supported by the FIB which contains mappings of a name prefix to an interface. The interest is only sent towards an interface if the corresponding prefix matches. In case a forwarder receives a content object, the PIT is checked for entries containing the corresponding name and if matching the content object is downstreamed. In any case the content object is added to the content store.

Our experiments are carried out in a network consisting of NDN Forwarding Deamons (NFD) [2] and CCN-Lite [1] instances both implementing the Named Data Networking (NDN) concept [2] and its NDN-TLV packet format. The special feature of the NDN derivate is component-wise longest prefix matching for content store as well as PIT matching. Even though others implement exact matching of all name components there are no conceptional dependencies of either variant.

1.2 NFN - Named Function Networking

ICN is designed to perform effective content distribution. However, the authors of *Named Function Networking* (NFN) state that "amidst the cloud-computing area, only connecting users to information seems a halfway-vision" [8]. They propose to enable clients to encode arbitrary computations on content objects, string and integer values into a name and lookup a content object containing the result just by sending out such an interest. Names, interests and resulting content objects of this kind are treated by the network as common. For this work it is essential to point out that results are carried by content objects such that it is possible to formulate expressions which themselves incorporate results of other expressions.

To produce content objects containing computation results, the network must be augmented with special NFN-capable nodes which can execute expressions. In fact, the execution of an expression is completely transparent to the client. Under the orchestration of NFN nodes, the execution location is automatically chosen by the network.

Computation expressions are encoded in an extended version of λ-calculus [4] and packed into name components. However, λ-calculus is neither convenient to express application-specific code nor do larger expressions fit into a single interest packet. Therefore a special `call`-operator is added which enables to call a function provided in JVM byte code and distributed over the network as a content object (called a *named function*). We do not expose the actual names containing λ-calculus encoding (including the `call`-operator) but a more human-friendly form. For instance, in the following three examples the JVM byte code functions carried by the content objects `/word/count` and/or `/math/isPrime` are called:

```
/word/count("count these words")
/word/count(/text/to/count)
/math/isPrime(/word/count(/text/to/count))
```

The first name expresses a function call with a string as single parameter while in the second example the function takes the payload of the content object with the name `/text/to/count`. As the third example shows, a function can also take the result of another function as input data. Note that these examples are denoted in a human-friendly form while the actual wire format contains a λ-calculus expression packed into a name. In each example, at least one `call`-operator is used to call a higher level named function. Further details about the actual encoding of NFN expressions are provided by the authors in [8].

2. ACCESS CONTROL AND SECURITY IN THE CONTEXT OF ICN AND NFN

In today's connection-oriented Internet, transport layer security (TLS) is a standard solution for ensuring confidentiality [5]. Since TLS is designed for endpoint-to-endpoint connections, it can not be simply adopted to an ICN environment. To come up with a ICN-compatible solution, Smetters and Jacobson introduced the notion of *content-based security*: "Protection and trust [should] travel with the content itself rather than being a property of the connections over which it travels" [7][9]. The obvious solution following this spirit is to ensure confidentiality by securing the payload of content objects with symmetric key encryption: Access control is then implemented by providing the corresponding symmetric keys only to authorized consumers. When wrapping these keys with asymmetric key cryptography, they can be shipped directly through the ICN infrastructure itself. Since data authenticity and integrity are already addressed by network level packet signatures, no additional effort is needed. In the remainder of this section we discuss processing-related shortcomings of the aforementioned access control solution (wrt NFN) and state a complementary concept called *content-attendant policies*.

2.1 Problem Statement

Content-based security means to directly encrypt the payload of content objects *and* to provide the corresponding keys in a synchronized fashion. This requires some coupling between the data-publishing entity and the access-granting entity (or multiples of both sorts). Unfortunately, performing in-network content processing in conjunction with access control is not fully compatible with this model because content might be re-published by a processing entity in modified form. In consideration to the access-control policies of processed data, results must be made accessible following policies that are compatible with *all* policies attached to the input data, over all touched time intervals and all primary data producers. In a world where there would be only one place that can produce a derived result, one could assign the key and result publishing activity to a single entity. Conceptionally, we think this is not the way to go because

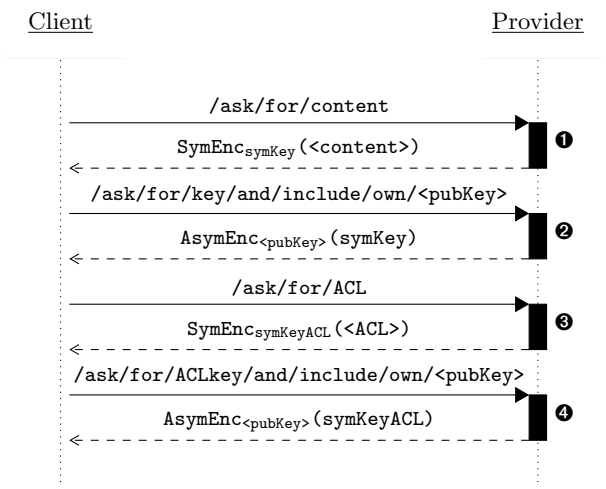

Figure 1: Flow of content from provider to client:
❶ Content (symmetrically encrypted)
❷ Symmetric key for content (asym. encrypted)
❸ ACL (symmetrically encrypted)
❹ Symmetric key for ACL (asym. encrypted)

the flexibility of decentralizing content distribution (ICN) and processing (NFN) should not be restricted. Moreover, it does not address the real problems: Who is responsible for synchronizing keys if the input content objects are managed by multiple parties? How does the key synchronizing system know which were the primary content sources from which some intermediate result was derived? These questions let us formulate an alternative decentralized approach which is based on *content-attendant policies*.

2.2 Content-Attendant Policies

In our approach of "content-attendant policies" we literally accompany each access-controlled piece of named information with a content object holding an *access control list* (ACL). The ACL itself is encrypted and access-controlled just as actual content. An access control list states to whom the actual content and the ACL itself can be propagated without violating the access-control policies. Technically, an ACL is a set of client identities for which access is explicitly permitted while all others are implicitly excluded.

In Figure 1 we specify a simple protocol which complies with the concepts of content-based security and content-attendant policies. Each client holds a public-private key pair and is identified by its public key. In step ①, a client expresses interest for a certain access-controlled and symmetrically encrypted content object. To fetch the corresponding symmetric key (step ②), the client propagates a respective interest which also includes his public key (identity). In case the provider's access control check is positive (matching public key against ACL), the requested symmetric key is returned (asymmetrically encrypted with the public key). Only the client (holder of the required private key) is able to decrypt the symmetric key and therefore also access the content. The remainder of the diagram is the part which is actually concerned with content-attendant policies. In step ③, the clients requests the ACL (symmetrically encrypted). Following the same procedure as in ②, the corresponding symmetric key is transferred to the client in step ④.

After receiving these four content objects and performing decryption, the client holds content and ACL in plain text. Thus, he is qualified to act as an independent provider of the information while enforcing the access-control policies. Since a single party can both be consumer and independent publisher of certain content at once, decentralized but still access-controlled propagation-chaining is possible. This fact becomes especially useful if we bring into mind that information might change its form during propagation. We consider "changing the form of information" very broadly and include conversion, reduction or even combination with other information. We equalize this notation with the terms *processing* as well as *producing computation results* to summarizes what we gain from content-attendant policies: Content-attendant policies in conjunction with content-based security enables to perform decentralized computation chaining of access-controlled named data.

2.3 Decentralized Meta-Data Derivation

In the previous paragraph we motivated to allow changing information's form during propagation. Let us look at an example: We consider a processing step which combines some information carried by two access-controlled content objects `A` and `B`. The result is published with a new name. To be compatible with the proposed protocol, it is necessary to generate and publish an attendant ACL and also synchronize encryption keys. We call these tasks *key synthesis* and *ACL merging*. In the following two subsections solutions are presented.

2.3.1 ACL Merging

In order to enable further policy-compliant propagation and processing of a computation result, a new ACL must be generated. In maximal consideration to the access-control polices of the content at the beginning of a computation chain (primary data), we propose the following procedure. Access should be restricted to exactly those consumers which are authorized to access both `A` and `B`. Conceptionally, this means to compose a new ACL from the overlap of the credentials in ACL_A and ACL_B. Technically, this is achieved by incorporating exactly those user identities which show up in the ACLs of both touched content objects and discarding all others. This way, a result is only made available to exactly those clients which were also able to receive `A` and `B` directly and perform the computation locally. Thus, it is ensured that no client gains additional information from computation results which he could not retrieve directly.

2.3.2 Key Synthesis

Symmetric encryption keys must be reproducible over time. On one hand, the name-content binding must be unique in ICN. On the other, data and keys might be requested delayed and/or be served from different locations. Therefore, we propose that execution locations apply a commonly known seeded random key generator. This way, the task of generating reproducible keys is reduced to synchronizing seeds among authorized parties. Conceptionally, common information exclusively hold by all those parties are the keys of all touched input data. Thus, we propose to derive the seed from this information (e.g. hash of concatenation or XOR). Consequently, exactly those parties are able to reproduce a result's symmetric keys which can also produce (or read) the result.

3. ACCESS-CONTROLLED COMPUTATION CHAINING IN ACTION

Our evaluation takes place in a NDN-compatible environment. The nodes are either NFD instances (called *NDN nodes*) or CCN-lite instances with nfn-scala [3] (called *NFN nodes*). Wiring is simulated with UDP connections. Determined by nfn-scala, named functions are implemented in the Scala programming language and published in compiled form as JVM byte code. For symmetric key encryption we apply AES-256 while for asymmetric key encryption we deploy RSA with a key length of 1024 bit. In this section we introduce a use case and step through a computation chaining example which is covered by our setup and implementation.

3.1 Analyzing and Combining Running Data

We process personal time-location series as captured by common activity trackers. Each track consists of waypoints made up of a longitude, a latitude and a time stamp value. Furthermore, each track is dedicated to an owner who grants access permissions. An owner is identified by a string (e.g. `"alice"` or `"bob"`) while privileged clients are identified with their RSA public keys. A *data storage unit* (DSU) is a NDN node which serves tracks with corresponding ACLs and keys. A *data processing unit* (DPU) is a NFN-capable node equipped with RSA public-private key pairs to execute NFN computations on access-controlled data.

3.2 Computation Chaining Example

Assume that the client Charly is interested in the relative ranking of Alice and Bob who both participated the New York Marathon 2016. Because Charly does not want to compute the result locally, he triggers the computation chain described in Figure 2. The in-network computation directly expressed by Charly produces a result based on a comparison of intermediate results. These intermediate results (elapsed duration of both runners) are computed in separate processes by directly making use of the tracks as recorded by the runners and published by a DSU. For illustration purposes, we assume that the intermediate results are produced on another DPU as the ranking. In fact, this is determined by the network and transparent to the client.

3.2.1 Flow of Content and (Intermediate) Results

In step ①, Charly triggers the computation chain. He does this by expressing to call the named function published under the name `/make/ranking`. An event (`NYmarathon16`) and two participant identifiers (`alice` and `bob`) are passed as arguments. After all (sub-)computations are executed successfully, Charly receives the result delivered as a content object and symmetrically encrypted for security purposes.

The execution location of Charlies computation expression does not directly interact with the DSU but makes use of the named function `/get/duration` to obtain the intermediate results. In steps ② and ③, according interests are propagated to fetch the elapsed duration of Alice's and Bob's runs. Again, the result of these two sub-computations are symmetrically encrypted and handed back through the ICN infrastructure. In steps ④ and ⑤, it is shown that the responsible execution locations (DPU_2) need to fetch the symmetrically encrypted tracks of both runners (captured data) to satisfy the computation requests.

3.2.2 Flow of ACLs and Symmetric Keys

So far, we focused on fetching symmetrically encrypted results from a DPU and content from the DSU. As indicated in gray, there is more communication for synchronizing keys and distributing ACLs. To make this clear, Figure 3 shows the exchange of interests and content objects between DPU_1 and DPU_2 which were left beside in part ③ of Figure 2 ($\star$).

In step ①, DPU_1 requests the symmetric key for the computation result. As described in the previous section, the key is produced by a seeded generator in order to enable others to re-generate exactly the same key at any time (key synthesis). The seed is derived from the symmetric key for Bob's track because this is common information of all authorized parties (and only these).

In order to check if Charly is authorized the receive the symmetric key for the ranking, DPU_1 needs to merge the ACLs from the both intermediate results as described in the previous section. Therefore in steps ② and ③, DPU_1 requests (among other content objects) the ACL of Bob's duration as well as the corresponding symmetric key from DPU_2.

As Figure 2 shows, the signatures (computation expressions) for expressing interest in the actual content, the ACL or the symmetric keys are quite similar. There is simply an additional parameter to specify the ACL (`ACL`) or the keys (`KEY` and `ACLKEY`) instead of the actual result (no additional parameter).

4. DISCUSSION AND FUTURE WORK

In this section we discuss our contribution and name subjects for future work.

4.1 Decentralized Computation Chaining

As already demonstrated in the previous section, protocols following the principle of content-attendant policies allow to perform distributed computation chaining for access-controlled named data. In fact, there is no central party which "oversees" an entire computation chain or is involved in every processing step. Each "chain link" solely acts as a client of the previous ones and as a publisher for the following. Every computation step in a chain takes place in an arbitrary location which is equipped with sufficient credentials. Thus, content-attendant policies allow to express arbitrary complex computations on content with restricted access by still being data-centric rather than location-oriented.

Since data owners just define access-control policies for their published data but not for all possible processing results, these are derived at execution time by the executing party. We address this task under the term "ACL merging" and balance the data owner's and consumer's requirements as following: No access-controlled information should be leaked to unauthorized parties while a consumer should be able to trigger any in-network computation which he could also compute locally. In other words, a consumer can request exactly those computation results which solely incorporate access-controlled data he is anyway allowed to access.

4.2 Generalized Access Control Policies

The access-control policies in our examples forbid every party to read any processing result which could not also be produced locally. Even though this is a proper assumption for many applications, more flexibility desirable. For instance, it might be reasonable that a client is be allowed

Figure 2: Computation Chain: ❶ Client requests ranking from NY Marathon 2016 including Alice and Bob ❷❸ Triggering computation of Alice's and Bob's duration (intermediate results) ❹❺ Requesting the time-location tracks of Alice and Bob on which all computations are directly or indirectly based ⋆ Further details in Figure 3: Fetching ACLs and keys for each step are just foreshadowed in this Figure.

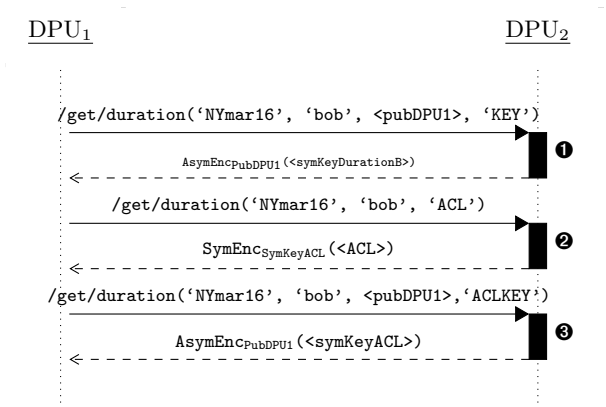

Figure 3: Complementation of Figure 2 (⋆):
❶ Symmetric key for intermediate result.
❷ ACL for intermediate result.
❸ Symmetric key for ACL.

to read a computation result (e.g. statistical evaluations) but not the primary data itself. Therefore, allowing more general access-control policies is subject of future work.

We think that in any more general approach the following difficulties show up in some form. First, named functions can be published by any (untrusted) third party. Thus an execution location needs a mechanism to ensure that third-party code really produces policy-complying results (e.g. if certain information must be reduced this must be verified). With policies as introduced in this paper, this is relatively trivial because the execution location simply needs to ensure that no unneeded access-controlled content is touched during execution (probably to maliciously publish it as part of the result). However, for more general policies, appropriate mechanisms are required. Second, when defining policies, a data owner might not explicitly oversee all computational needs which he can accept as "reasonable" (i.e. which do not reveal "too much" information). Still, ideal policies provide sufficient information for an execution location to properly assess *any* "reasonable" result request. Again, with access control policies as introduced in this paper (binary: read/denial) this is trivial while for the general case principles for designing appropriate policies must be developed.

A generalized solution for these issues might go into the following direction: ACLs define more fine-grained authorizations than just read-permission for a peace of named data as whole. For instance, access control policies for a time-location track might not just regulate access to absolute coordinates but also to relative ones. Relative coordinates are still sufficient to compute properties such as length or average speed but hide real spatial positions. In general, the idea is that polices not just regulate access to the content as whole but also to "filtered" versions which strip away certain sensitive information. Selectively sharing certain "attributes" and hide away others might go together with *attribute-based access control* [6].

4.3 Trust Model

Content-based security and content-attendant policies provide a solution for access control and content confidentiality. In support of this, the data owner defines an ACL which establishes several trust relationships. First, the owner explicitly allows each authorized client to propagate the (probably processed) content and relies on its trustworthiness to respect the access-control policies. Second, trust relationships between all authorized clients are assigned. These are necessary because a client in NFN can not assess the correctness of a result nor can't the provenience of a computation result (execution locations of all sub-tasks) be fully controlled. To find practical solutions for dependable results "in general" is subject of future work. For illustration, an intermediate DPU (consumer and publisher at the same time) is trusted by the data owner to enforce the polices but potentially also by any other authorized client not to deliver manipulated results.

Further, there need to be trust relationships between a client requesting a computation result and all providers of all called functions. The client has to rely that the named functions implementation is according to the specification such that the delivered results are correct. Note that there is no need for a trust relationship between data owner and function provider since it is up to the execution location to enforce the access-control policies (See 4.2).

4.4 Performance Considerations

A protocol as introduced in this paper causes a communication overhead due to accompanying meta-data and key synchronization but can also save computation power, shorten response times and time and reduce network load.

From the communication overhead point of view, it can be stated that distributing symmetrically encrypted content and separately handling the key access, doubles the number of interests. Additionally, two more interests per content object are propagated if a consumer makes use of content-attendant policies (e.g. to act as a producer for others). All in all, this means at most a four-fold increase in the number of interests. However, with application-specific optimizations, the communication overhead might be reduced. For example, if the applications access-control policies allow to encrypt the ACL and the accompanied content with the same symmetric key, the number of interests is at most three-fold.

On the other side, breaking up comprehensive computations into sub-tasks not only allows to re-use existing code but also to re-use results due to the caching capabilities of ICN. As a consequence, repeated computation effort is avoided and results might be available earlier. To organize the re-usability of access-controlled computation results as effective as possible, results and according ACLs are encrypted with one symmetric key for all clients. Thus, cached results and ACLs are adaptable by every authorized party. Only the asymmetrically encrypted keys must be produced individually for every client. Further, traffic is not just reduced because of ICNs' effective content/result distribution but also due to the fact that in-network computations might be performed close to the data source. For instance, if a named function (instead of a function on the clients machine) returns the largest entry of a huge vector, network load is reduced because the propagated result is significantly smaller than the functions input data.

5. CONCLUSION

In order to tackle confidentiality and access control in the context of in-network processing for ICN, we have introduced *content-based policies* as a complementary concept to content-based security. Combinable with concepts for other security guarantees such as integrity or authenticity, content-attendant policies constitute an essential building block for a comprehensive security solution in ICN. The contribution of the introduced concept to such a system is to enable flexible, distributed and location-agnostic processing on access-controlled named content. A system following the principle of content-attendant polices balances the privacy settings of the data-owners with the consumer's legitimate demands for results. A consumer can read exactly these results which he could also compute locally.

6. REFERENCES

[1] CCN-Lite, 2016. http://ccn-lite.net.

[2] Named Data Networking, 2016. http://named-data.net.

[3] NFN-Scala, 2016. https://github.com/cn-uofbasel/nfn-scala.

[4] Church, A. An unsolvable problem of elementary number theory. *American Journal of Mathematics 58*, 2 (April 1936), 345–363.

[5] Dierks, T. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, Oct. 2015.

[6] Hu, V. C., Ferraiolo, D., Kuhn, R., Friedman, A. R., Lang, A. J., Cogdell, M. M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST Special Publication 800*, 162 (2013).

[7] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies* (New York, NY, USA, 2009), CoNEXT '09, ACM, pp. 1–12.

[8] Sifalakis, M., Kohler, B., Scherb, C., and Tschudin, C. An information centric network for computing the distribution of computations. In *Proceedings of the 1st International Conference on Information-centric Networking* (New York, NY, USA, 2014), ICN '14, ACM, pp. 137–146.

[9] Smetters, D. K., and Jacobson, V. Securing network content. Tech. rep., PARC, October 2009.