Secure Sharing of Spatio-Temporal Data through Name-based Access Control

Laqin Fan, Lan Wang
Department of Computer Science, University of Memphis, USA
{lfan1, lanwang}@memphis.edu

***Abstract*—As more and more mobile data is collected continuously in space and time for a variety of purposes such as health monitoring and driving behavior tracking, people are increasingly concerned about their privacy when sharing their data. To minimize privacy leakage, data owners often want to restrict the access to their data based on space and time constraints while meeting each data user's specific needs. In this paper, we introduce an access control system over Named Data Networking (NDN) that allows data owners to schematize and control data access at a fine granularity with respect to time, space, and user needs. More specifically, we designed spatio-temporal access control policies using hierarchically structured names, extended Name-based Access Control to support spatio-temporal policies, and incorporated publish-subscribe functionality for real-time data sharing. Moreover, we developed an NDN-based prototype based on our design and evaluated its performance in sharing both real-time and historical data.**

***Index Terms*—Access Control, Named Data Networking, Name-based Access Control, Spatio-Temporal Data**

I. Introduction

An ever increasing amount of spatio-temporal data is being collected, stored, and shared to improve our lives. For example, modern smart home systems utilize sensors to collect data and provide services in our homes. Mobile health apps generate health data continuously for fitness assessment and medical diagnosis. Insurance companies use in-car trackers to monitor people's driving behavior. As such data may contain sensitive information, e.g., medical conditions, visited locations, and personal activities, the data owners often desire to restrict access to their data based on space, time, data type, and data analysis requirements. For example, Bob may allow his coach to access his physical activity data only when he is at a gym, and allow a researcher to monitor his heart rate only during the daytime when he is not at home. Several access control models have been proposed to support more sophisticated policies, such as RBAC [1], KP-ABE [2], and CP-ABE [3]. However, RBAC defines access permissions based on users' roles without considering contextual information such as time, space, and data type. Attribute-based access control schemes such as KP-ABE and CP-ABE do consider data and user attributes, but they have many open issues such as auditability, attribute storage/sharing, and scalability in large systems [4].

Named Data Networking (NDN) [5] is a data-centric Internet architecture that gives each piece of data a unique, hierarchically structured, and semantically meaningful name. It retrieves data by name directly, and secures the data by binding the data content and its name using a cryptographic signature from the data producer. The basic components in the NDN design, i.e., hierarchical naming structure, data-centric security [6], and name-based data/key distribution, make it easy to define and enforce fine-grained access control policies, as demonstrated by the Name-based Access Control (NAC) scheme [7]. NAC achieves end-to-end confidentiality and enables automatic key distribution, but its current design supports only time-based access control policies. In this paper, we design fine-grained spatio-temporal access control policies using hierarchically structured names, extend NAC to support spatio-temporal policies, incorporate publish-subscribe functionality for real-time data sharing, and develop an NDN-based prototype. Finally, we evaluate the prototype's performance using Mini-NDN [8], an emulation framework for NDN.

II. Background and Related Work

A. Existing Access Control Schemes

The purpose of Access Control [9] is to protect data and resources from unauthorized users. Below we discuss two commonly used approaches: role-based access control (RBAC) and attribute-based access control (ABAC).

An RBAC scheme contains four basic components: users, roles, permissions, and sessions [1]. Each user is assigned one or more roles, and each session is a mapping between a user and an activated subset of the user's roles. Permissions are associated with roles. If a user with a particular role is given permission to access a dataset, then that user can access all the data in the dataset regardless of the data attributes (e.g., time and location) associated with each piece of data.

Goyal et. al. proposed an ABAC scheme called ***Key-Policy Attribute-Based Encryption (KP-ABE)*** to share data selectively [2]. With this scheme, data is encrypted with a set of *data attributes*, and each user gets a private key encoded in his/her access control policy expressed as a set of conditions over some data attributes. For example, conditions over time, location, and activity attributes may be "hour >= 8:00 and hour <= 20:00", "location != home", and "activity == smoking". These conditions can also be combined using logical operators. A user can decrypt a piece of encrypted data if and only if the set of attributes associated with the data satisfy the access control policy encoded in the user's private key. Another ABAC scheme is ***Ciphertext-Policy ABE (CP-ABE)*** proposed by Bethencourt et. al. [3]. Unlike KP-ABE, CP-ABE associates a set of *user attributes* with each private key, and encrypts data with an access control policy expressed

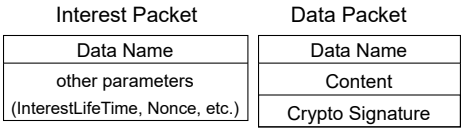

Fig. 1: NDN Interest and Data packets.

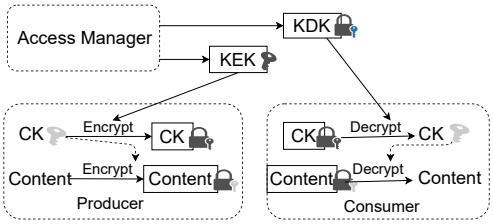

Fig. 2: NAC scheme

as conditions over some attributes, such as "age $>=$ 18" AND "position $==$ manager". A user can decrypt a piece of data if and only if the set of attributes encoded in his/her key satisfy the access control policy encrypted with the data.

In summary, KP-ABE takes into account data attributes in access control policies, while CP-ABE focuses on user attributes (so CP-ABE is conceptually very similar to RBAC). Theoretically, user attributes can be converted to data attributes and vice versa if needed, but each scheme may be more natural for a particular application depending on which type of constraints dominate the policies. A detailed discussion of ABAC is out of scope for this paper. We refer the readers to the survey by Servos and Osborne [4] for its open issues.

B. Named Data Networking and Name-based Access Control

Most of today's Internet applications are built on web protocol semantics of requesting data by names, but data names must be first translated to Internet addresses of specific end hosts first, then the applications fetch data from those hosts. Named Data Networking (NDN) [5] adopts the web's request-reply communication model, but the NDN request and reply work at the network packet granularity (Figure 1). A consumer first sends an Interest with the desired data name to the network. Routers then forward this Interest toward the producer. As soon as the matching data is found at an intermediate node (in its cache) or the original producer, the node will send a Data packet back to the consumer following the reverse path of the Interest. The consumer can verify the authenticity of the data by checking the signature in the data packet using a trust schema defined in the application. The trust schema consists of a set of trust rules specifying which keys can be used to authenticate other keys or data, as well as a trust anchor which is the public key of an entity trusted by the application [6].

Name-based Access Control (NAC) [10] utilizes NDN's hierarchical naming structure to express fine-grained access control policies and automate key distribution. There are three entities in NAC: access manager, producer, and consumer (Figure 2). The access manager represents the data owner

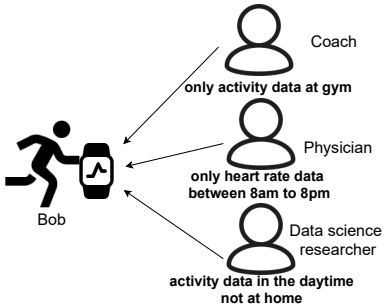

Fig. 3: An Example of mHealth Data Sharing

to specify access control policies and distribute keys. It first generates a pair of public and private keys as the Key-Encryption Key (KEK) and Key-Decryption Key (KDK), and encodes the access control policy in the KEK name. It then publishes the KEK as a Data packet. The producer fetches the KEK and generates Content Keys (CKs), which are symmetric data encryption keys, at the granularity specified in the access control scheme. For example, if the access control is at an hourly time granularity for Bob's activity data, a new CK will be generated for each hour of the data. Then the producer encrypts each CK using the KEK, and publishes the encrypted CK. Meanwhile, the access manager publishes the KDK by encrypting it with each authorized consumer's public key. After a consumer retrieves the encrypted data, it learns the CK's name from the data packet. It fetches the CK which is encrypted with the KEK. It then fetches the corresponding KDK and decrypts it using its private key. Next, it decrypts the CK using the KDK. Finally, it decrypts the data using the CK. *Since the key names follow the naming conventions defined in NAC, every step of the key distribution is automated.*[1]

Theoretically access control granularity can be associated with any data attribute, **but the existing NAC design focuses only on time-based constraints [10] or does not support such constraints at all [7]**. Our work addresses this limitation.

III. Design

The goal of our work is to develop an access control scheme for data owners to share their data using fine-grained policies that include time and/or location constraints. To facilitate our explanation, we introduce a mobile health data sharing application as an example (Figure 3). Suppose Bob uses his smartwatch to collect his daily activity data, e.g., heart rate, sleep patterns, and stress level. He wants to share the activity data generated in the daytime but not at home with a data science researcher, his heart rate data between 8am and 8pm daily with his physician, and his activity data generated in a gym with his coach. In this section, we describe our namespace design and spatio-temporal access control policies, and illustrate how the policies can be enforced cryptographically.

[1]Note that we have been describing the original NAC design that distributes the KDKs using consumers' public keys. It is also possible to use attribute-based encryption to distribute the CKs [7].

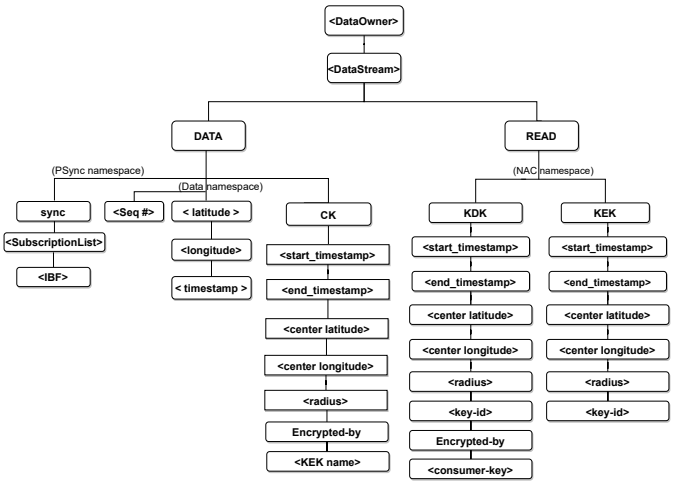

Fig. 4: Naming Scheme of Spatio-Temporal Access Control

A. Namespace Design

In NDN, data naming follows a hierarchical tree structure which has been used widely to support application development, routing scalability, automatic data authentication, and access control. Our namespace design has several requirements: naming data semantically and meaningfully, indicating data ownership, implying data encryption and decryption relationship, and expressing access control policies with spatio-temporal restrictions. Figure 4 shows our namespace design. We use the "`DATA`" sub-namespace for application data and content keys, and the "`READ`" sub-namespace for access control.

1) Data Namespace: Under "`DATA`", there are several branches for the actual application data, pub-sub functionality, and content keys (CK). The application data name starts with "`<latitude>`" and "`<longitude>`" which are geographic coordinates in decimal format to represent the data collection location. It ends with a "`<timestamp>`" name component to indicate data collection time, and is represented in ISO 8601 format, i.e., "`YYYYMMDDThhmmss`". For example, one piece of Bob's activity data may be named "`/Bob/activity/DATA/35.111287/-89.928124/20201020T123000`".

For real-time data distribution, we employ the publish-subscribe functionality from PSync [11] which uses the "`sync`" name branch for its protocol messages. Because PSync assumes a sequential data namespace, we have to add a branch under "`DATA`" with a sequence number name component "`<Seq#>`" to encapsulate the actual data which does not use a sequence number in its name. With PSync, the data consumers can subscribe to the name prefixes of interest. Once the data producer generates new data, the subscribers will be notified of the new data name. For example, an app running on Bob's coach Alice's smartphone can subscribe to Bob's activity data, i.e., "`/Bob/activity/DATA/`". Once a new piece of data is produced by Bob's smartwatch, a Sync message will be sent to Alice's phone containing the new data name, e.g., "`/Bob/activity/DATA/509`". Alice's smartphone app can use that name in an Interest to fetch the new data with the sequence number 509 which encapsulates the actual data "`/Bob/activity/DATA/35.111287/-89.928124/20201020T123000`".

The CK namespace is for the data producer to name the content keys it generates for data encryption (Section II-B). In our design, CKs can be granular in both time and location, which is expressed in the CK name with a time period and a spatial area. Each CK is used to encrypt all the data generated in the time and area indicated in the CK's name. We identify a time period by a "`<start_timestamp>`" and "`<end_timestamp>`", and represent a spatial area using a center point and a radius (i.e., "`<center_latitude>/<center_longitude>/<radius>`"). In addition, a CK name includes additional name components, "`<ENCRYPTED-BY>`" and "`<KEK-name>`", to indicate which KEK is used to encrypt the CK.

2) Access Control Namespace: Under "`READ`", there are two sub-namespaces used by the access manager to publish data consumption credentials, i.e., the KEKs and KDKs. In NAC, access control policies are expressed in KEK and KDK names. We extend the naming convention of NAC by adding spatial restrictions. Each KEK is named "`<DataOwner>/<DataStream>/READ/KEK/<start_timestamp>/<end_timestamp>/<center_latitude>/<center_longitude>/<radius>/<key-id>`", which identifies the data protected by this KEK, i.e., the data is generated by "`<DataOwner>`" with the type "`<DataStream>`" in the time period and spatial area as indicated in the KEK name. KDKs have the same name structure as KEKs except for two differences: (a) the name component "`KEK`" is replaced with "`KDK`", and (b) the KDK name is appended with an authorized consumer's key, "`<DataOwner>/<DataStream>/READ/KDK/<start_timestamp>/<end_timestamp>/<center_latitude>/<center_longitude>/<radius>/<key-id>/ENCRYPTED-BY/<consumer-key>`", since the KDK is encrypted for a specific consumer. Our naming scheme is also applicable to policies which contain only temporal or spatial constraints - we use the symbol "*" to populate the spatial or time name components in this case.

B. Fine-Grained Access Control

Fine-grained access control ensures that users are granted access to the minimum amount of data to meet the users' needs. In our work, we provide a precise way to express access control policies using spatio-temporal constraints. The temporal constraint is specified using a start time and end time, e.g., "`between 8am and 12pm from 09/01/2020 to 09/05/2020`". The spatial constraint is represented by a center GPS coordinate and a radius, e.g., "`center_latitude: 35.121196, center_longitude: -89.938124, radius: 50m`" can be used to locate the fitness center at University of Memphis (Figure 5). Figure 6 shows our policy structure, where "`Data Prefix`" is the name prefix of the shared dataset, "`TimeLocation`" contains a time schedule and a spatial bound, and "`User ID`" is the name prefix of the data consumer.

Policies with temporal constraints may have overlapping time intervals. For an instance, Bob may want to share

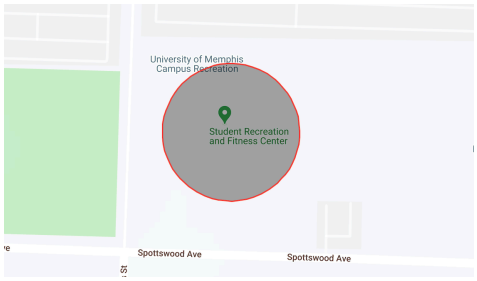

Fig. 5: An Example Spatial Bound

Policy
- Data Prefix: <name prefix>
- TimeLocation:
  - startDate:<YYYYMMDDT000000>
  - endDate:<YYYYMMDDT00000>
  - startHour:[0,24]
  - endHour:[0,24]
  - center: <GPS coordinates>
  - radius:(in meters)
- User ID: <name prefix>

Fig. 6: Spatio-temporal Access Control Policy Structure

his activity data at gym "`between 8am to 12pm on Sept 1, 2020`" with coach Alice, but share the data "`between 8am to 8pm on Sept 1, 2020`" with physician Dave. In this case, the time interval will be divided and multiple KEK/KDKs will be generated to avoid conflicts (Figure 7). More specifically, the access manager will generate two KEKs, one for "`8am-12pm`" and the other for "`12pm-8pm`". It will encrypt the KDK for "`8am-12pm`" using Alice's key and Dave's key separately, but encrypt the KDK for "`12pm-8pm`" using only Dave's key so Alice cannot decrypt the data from 12pm to 8pm.

Currently the granularity of a spatial area is fixed at a specific level, e.g., building, neighborhood, or city, so we do not have overlapping spatial constraints. However, if we allow arbitrary spatial areas or mix different granularities, then there can be overlapping spatial areas. This is an open issue for future research.

C. Granular Content Key

In NAC, the data producer encrypts data with a symmetric key (CK), and users that get the CK can decrypt the data. The granularity of CK depends on the granularity of the access control policies. If the minimum unit of temporal access control is one hour, the CK has to be changed at least hourly. For spatial access control, the CK granularity needs to be at least the same as the spatial access control granularity, which can be at the building level, neighborhood level, city level, etc, depending on the configuration. For policies with both temporal and spatial restrictions, the CK granularity is the product of the time granularity and location granularity. Figure 8 shows CK names with different granularities over time with a fixed spatial bound at the building level.

Let's consider the spatio-temporal policy in Figure 9. The granularity for temporal restriction is hourly, while spatial restriction is at a building level. Suppose Bob's smartwatch produces his activity data every second. Based on the policy and data generation rate, the CK can change every minute, two minutes, or hour. The CK granularity plays an important role in data sharing. If a CK is generated per minute, then the CK "`/Bob/activity/-DATA/CK/20200901T083000/20200901T083100 /35.114112/-89.943279/100/<key-id>/`" covers 60 data points produced between "`20200901T083000`" and "`20200901T083100`". When this CK is compromised, those 60 data points will be leaked. However, if a CK is generated every second, only one data point will be leaked if a CK is compromised. On the other hand, the finer CK granularity incurs higher cost to generate and distribute the CKs.

D. Access Control Policy Enforcement

We now use an example to illustrate how policies are enforced in our scheme. Suppose the policy in Figure 9 is assigned to coach Alice, which means user "`/edu/memphis/gym/coach/Alice`" is allowed to access Bob's activity data collected from the fitness center represented by "`center(35.114112, -89.943279)`" and "`radius(100m)`", between "`8am`" and "`12pm`" from "`09/01/2020`" to "`09/05/2020`" daily. Once the policy is configured, the access manager (e.g., a server) generates five pairs of KEK/KDK for the same time interval, same spatial area, but for different dates from "`09/01/2020`" to "`09/05/2020`". These five data consumption credentials can help Alice access Bob's activity data. Meanwhile, the CK's temporal granularity is chosen by the data producer, Bob's smartwatch, to be 1 minute (i.e., the CK changes every minute) with a fixed location (fitness center).

Figure 10 shows the naming convention for the data and corresponding keys.

When Bob's smartwatch generates a piece of activity data in the fitness center at 8:30:30 on Sept. 1, 2020, this data point will be named with Bob's coordinates "`35.113758, -89.937144`" and timestamp "`20200901T083030`". To control access to the data, a CK is used to encrypt it - the time interval and spatial area in the CK name need to cover the timestamp and coordinates of that data. If such a CK exists, it can be used directly. Otherwise, a new CK with a proper name needs to be produced by the smartwatch. In Figure 10, the CK name contains the time interval from "`20200901T083000`" to "`20200901T083100`" and the spatial area "`Fitness Center`", which can cover the data. After the data is encrypted using the CK, the encrypted data along with the CK name is published to the network.

To control access to the CK, the smartwatch uses a proper KEK to encrypt it, which means the time interval and spatial area in the KEK name cover the ones in the CK name. If such a KEK exists, it can be used directly. Otherwise, the smartwatch sends a KEK Interest with a proper name prefix based on the naming convention (Figure 4), appending the CK's time interval and spatial area, e.g., "`/Bob/activity/READ/KEK/20200901T083000/20200901T083100/35.114112/-89.943279/100/`". Once the

(a) KEK and KDK for access between 8am 9/1/2020 and 12pm 9/1/2020

/Bob/activity/READ/KEK/20200901T080000/20200901T120000/35.114112/-89.943279/100/<key-id>

/Bob/activity/READ/KDK/20200901T080000/20200901T120000/35.114112/-89.943279/100/<key-id>/Encrypted-by/<Dave-key>

/Bob/activity/READ/KDK/20200901T080000/20200901T120000/35.114112/-89.943279/100/<key-id>/Encrypted-by/<Alice-key>

(b) KEK and KDK for access between 12pm 9/1/2020 and 20pm 9/1/2020

/Bob/activity/READ/KEK/20200901T120000/20200901T200000/35.114112/-89.943279/100/<key-id>

/Bob/activity/READ/KDK/20200901T120000/20200901T200000/35.114112/-89.943279/100/<key-id>/Encrypted-by/<Dave-key>

Fig. 7: KEK and KDK Names with Time Interval and Location

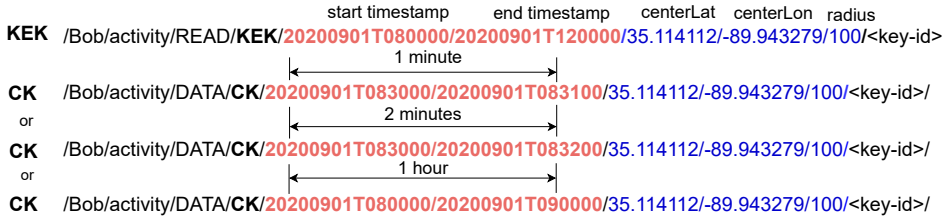

Fig. 8: An Example of Different CK Granularities based on an Access Control Policy

Dataset: /Bob/activity
TimeInterval_Start: 8am
TimeInterval_End: 12pm
StartDate: 09/01/2020
EndDate: 09/05/2020
Center_Lat: 35.114112
Center_Lon: -89.943279
Radius: 100
User: /edu/memphis/gym/coach/Alice

Fig. 9: An Example of Access Control Policy

access manager receives the Interest, it looks up its stored KEKs, and return an appropriate KEK to the smartwatch (if the KEK name does not match the Interest name, the actual KEK will be encapsulated in a Data packet that matches the Interest name). In Figure 10, the KEK name contains the time interval from "`20200901T080000`" to "`20200901T120000`" and the spatial area "`Fitness Center`", which can cover the CK. When the smartwatch encrypts the CK with the KEK, it appends the KEK name to the CK name. Upon receiving the encrypted CK, Alice's smartphone app can derive the KDK name from the CK data name by replacing "`KEK`" with "`KDK`", and appending "`Encrypted-by`" and the app's key name.

IV. Evaluation

We have developed a working prototype [12] based on the NAC library [13] (note that we modified the naming scheme in the NAC library to follow our design). This prototype can be applied in many situations, e.g., a smart home owner shares sensor data with family members or friends, a personal server collects mobile health data and shares the data with caregivers or others, and a cloud service shares a large amount of data with its users. Moreover, it supports real-time data sharing. In this section, we analyze security properties and evaluate the performance of our spatio-temporal access control prototype.

A. Security Analysis

Man-in-the-Middle Attack: NDN secures data directly, allowing producers to sign data packets and consumers to verify the signatures to ensure data integrity and authenticity. An attacker cannot impersonate another node to produce the correct signature for a data packet containing a key or content. In addition, if the attacker modifies another producer's data, consumers will know the change by verifying the signature.

Denial-of-Service Attack: NDN is designed to be resistant to DoS attacks. First, data cannot be sent unless it is requested by a consumer, which limits data flooding. Moreover, NDN supports in-network caching so content and keys can be stored in cache, which helps mitigating interest flooding attacks.

Compromised Data Consumer: NAC can handle this issue by revoking the consumer's privileges through changing the KEK/KDK pair. Once a compromise is detected, the affected KEK/KDK pair (and CKs) will be updated and the compromised consumer will not be able to access future data.

Attacker as Data Consumer: We assume that, before issuing a valid certificate to a user, the system has validated the user's identity by verifying the user's email address, phone number, etc.. This certificate binds the user's public key and the key name using the trust anchor's key (or a delegate's key).

B. Performance Evaluation

We used Mini-NDN [8] to perform experiments that share data from storage and in real time. We build NDN topologies with hundreds of nodes running on a single machine. Those nodes are connected via virtual Ethernet interfaces.

Data Collection. We collected a sample GPS data stream which contains time and location attributes through the MD2K Cloud platform (CerebralCortex) [14]. There are a total of 2,000 data points with a data generation rate of 1 per second.

latitude longitude timestamp

Data Name /Bob/activity/DATA/**35.113758/-89.937144/**20200901T083030

start timestamp end timestamp centerLat centerLon radius

CK Name /Bob/activity/DATA/**CK/**20200901T083000/20200901T083100/**35.114112/-89.943279/100/**<key-id>/

start timestamp end timestamp centerLat centerLon radius

CK Data Name /Bob/activity/DATA/**CK/**20200901T083000/20200901T083100/**35.114112/-89.943279/100/**<key-id>/ENCRYPTED-BY/<KEK-prefix>

start timestamp end timestamp centerLat centerLon radius

KEK Name /Bob/activity/READ/**KEK/**20200901T080000/20200901T120000/**35.114112/-89.943279/100/**<key-id>

start timestamp end timestamp centerLat centerLon radius

KDK Name /Bob/activity/READ/**KDK/**20200901T080000/20200901T120000/**35.114112/-89.943279/100/**<key-id>/ENCRYPTED-BY/<Alice-key>

Fig. 10: KEK and KDK Names with Time Interval and Location

TABLE I: Evaluation Metrics

Metrics	Description
Number of CK	Number of CKs with different time-based granularities
Traffic Overhead	Number of packets transmitted in the system
Sync Delay	Time between when new data is updated by data producer and when notification is received by data consumer
Data Production Time	Time for CK generation (32-byte key), KEK retrieval, CK encryption (RSA2048), and data encryption (AES256)
Data Consumption Time	Time for CK retrieval, KDK retrieval, CK decryption (RSA2048), and data decryption (AES256)
Communication Delay	Time for a data consumer to receive a piece of data after sending the interest
Data Retrieval Time	Total time for a data consumer to access a piece of data successfully through spatio-temporal access control

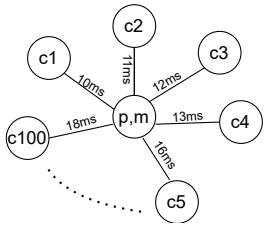

Fig. 11: Mini-NDN topology

Experimental Topology. We create a star topology with 101 nodes and 100 links (Figure 11), where we run a data producer and an access manager in the center node, and run data consumers in all the other nodes. The delay for each link is in the range of 10-20 ms.

Access Control Policy. The policy we set up in our experiments is applicable to all data consumers for easy deployment. It is "`access the GPS data stream generated between 8am and 12pm in Dunn Hall on 09/01/2020`".

Evaluation Metrics. Table I shows the metrics for non-real-time data sharing and real-time data sharing.

Evaluation Results. In our experiments, we vary the CK granularity to show the traffic overhead and the different number of CKs generated (see Figure 12(a)). When the CK changes every second, sharing real-time data and non-real-time data lead to 16,000 and 8,000 packets, respectively. Real-time data sharing generates more traffic than sharing historical data because the former involves PSync protocol messages. The numbers decrease to 12,000 and 4,000 when the CK changes every hour. A coarser granularity generates fewer CKs, but once a CK is compromised, more data will be exposed.

Figure 12(b) shows the average sync delay with different CK granularities. As we change the CK granularity, the average sync delay for each data point stays around 15ms, which is due to the one-way delay between the consumer and producer. Figure 12(c) shows that the average communication delay is about 30ms, which is around one round-trip delay between a consumer and a producer. Both results are expected.

We measure the total data production time for both real-time data sharing and historical data sharing. In Figure 12(d), the data production time for both data sharing modalities has no major difference given a CK granularity. However, as the granularity becomes coarser, the total data production time becomes less for both. The coarser CK can cover more data, so we can save time for CK generation. We can make similar observations at the the data consumer side - as the granularity becomes coarser, data consumption time becomes less for both (Figure 12(e)). The reason is a CK at a coarser granularity can be reused to decrypt more data, so we can reduce the time spent on retrieving and decrypting new CKs and KDKs.

Figure 12(f) shows the average data retrieval cost for accessing a data point. With real-time data sharing, it takes more time to access each data point than sharing historical data from storage due to the sync delay. Overall, using a coarser CK incurs less data production delay, consumption delay, and traffic overhead than using a finer CK.

The results presented in this section show that our spatio-temporal access control prototype is able to support sharing both historical data and real-time data with different time-based CK granularities. Most notably, there is a tradeoff between security and overhead, i.e., a finer CK granularity leads to more CKs generated and longer data retrieval time, but when a CK is compromised, less data will be leaked. In addition, given the same CK granularity, real-time sharing incurs more traffic overhead and data retrieval delay than historical data sharing due to the overhead of the sync protocol.

V. Discussion

In our naming scheme, we use location information in the data, CK, KEK, and KDK names. Since data is fetched by

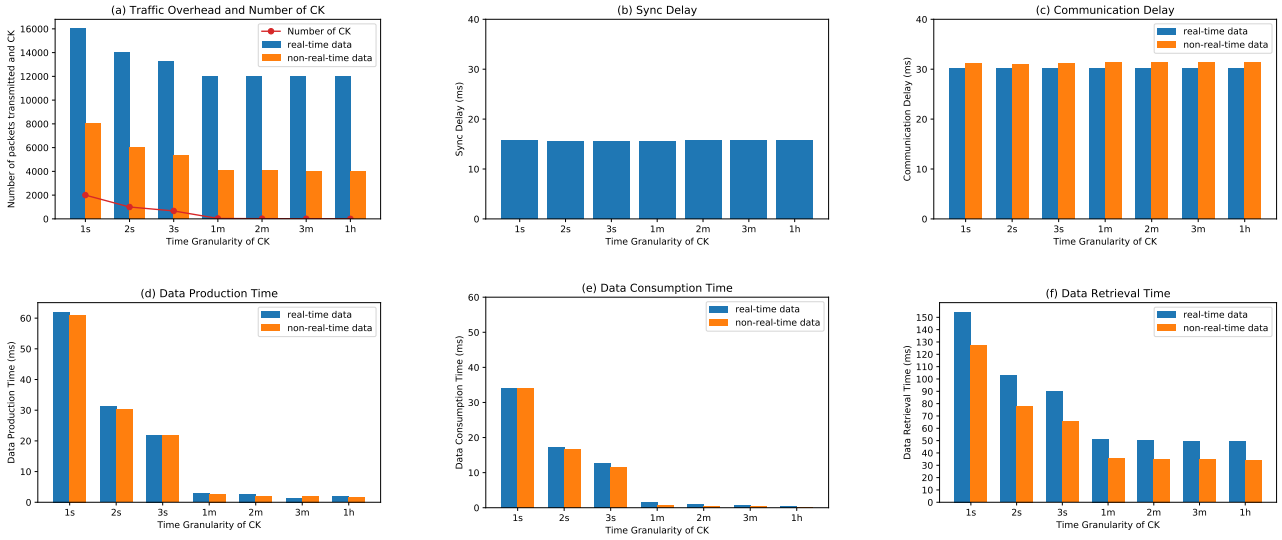

Fig. 12: Results of Performance Evaluation

giving the data name, attackers would be able to sniff the data packets and extract the location information through the name. To prevent location information exposure, we can obfuscate the name using an encryption function or hash function so that only authorized users are able to derive the real data name. Some name obfuscation solutions have been proposed for NDN (e.g., [15]) and we can adopt one of those schemes.

Location coordinates can be represented by latitude, longitude, and altitude. For simplicity, we only use latitude and longitude. Spatial areas can be represented by a circle, square, rectangle, or other geometric shapes. In our design, we use a circle with a center GPS point and radius to express location-based restrictions in CK, KEK and KDK names for its simplicity. When the shape of a spatial area is not exactly a circle, this could result in false positives. We will investigate other representations in future work.

VI. Conclusion and Future Work

We have presented a spatio-temporal access control scheme for NDN. This work extends the existing NAC naming scheme to support fine-grained spatio-temporal access control policies and incorporates publish-subscribe functionality to enable real-time data sharing. We evaluated a preliminary prototype through security analysis and performance analysis by running Mini-NDN experiments. For our next step, we will conduct more comprehensive performance evaluation with different spatial granularities and multiple access control policies. We will also develop a user-friendly interface for data owners to configure access control policies.

Acknowledgment

This work was supported by NSF Grants 1629769 and 2019085. We are grateful to Christos Papadopoulos, Kan Yang, and the anonymous reviewers for their valuable feedback.

References

[1] P. S. Sandhu, E. J. Coynek, H. L. Feinsteink, and C. E. Youmank, "Role-Based Access Control Models," *IEEE Computer*, vol. 29, pp. 38–47, February 1996.

[2] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and Communications Security*, 2006, pp. 89–98.

[3] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of IEEE Symposium on Security and Privacy*, 2007.

[4] D. Servos and S. L. Osborn, "Current research and open problems in attribute-based access control," *ACM Computing Surveys (CSUR)*, vol. 49, no. 4, pp. 1–45, 2017.

[5] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *ACM SIGCOMM Computer Communication Review*, July 2014.

[6] Z. Zhang, Y. Yu, H. Zhang, E. Newberry, S. Mastorakis, Y. Li, A. Afanasyev, and L. Zhang, "An Overview of Security Support in Named Data Networking," *IEEE Communications Magazine*, vol. 56, pp. 62–68, November 2018.

[7] Z. Zhang, Y. Yu, S. K. Ramani, A. Afanasyev, and L. Zhang, "NAC: Automating Access Control via Named Data," in *Proceedings of 2018 IEEE Military Communications Conference (MILCOM)*, 2018.

[8] NDN Project Team, "Mini-NDN," https://github.com/named-data/mini-ndn, (Accessed on 10/11/2020).

[9] P. Samarati and S. de Capitani di Vimercati, "Access Control: Policies, Models, and Mechanisms," *Foundations of Security Analysis and Design*, vol. 2171, pp. 137–196, October 2001.

[10] Y. Yu, A. Afanasyev, and L. Zhang, "Name-based access control," Technical Report NDN-0034, NDN, Tech. Rep., 2015.

[11] M. Zhang, V. Lehman, and L. Wang, "Scalable Name-based Data Synchronization for Named Data Networking," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, May 2017.

[12] L. Fan, "Prototype Implementation for Spatio-Temporal Access Control in NDN," https://gitlab.com/lfan1/location-nac, (Accessed on 3/7/2021).

[13] NDN Project Team, "NAC: Named-Based Access Control Library for NDN," https://github.com/named-data/name-based-access-control, (Accessed on 10/11/2020).

[14] MD2K, "CerebralCortex," https://github.com/MD2Korg/CerebralCortex, (Accessed on 10/11/2020).

[15] C. Ghali, M. A. Schlosberg, G. Tsudik, and C. A. Wood, "Interest-Based Access Control for Content Centric Networks," in *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, September 2015.