NDN, Technical Report NDN-0023, 2014. http://named-data.net/techreports.html
Revision 1: July 22, 2014

An Endorsement-based Key Management System for Decentralized NDN Chat Application

Yingdi Yu
UCLA
yingdi@cs.ucla.edu

Alexander Afanasyev
UCLA
afanasev@cs.ucla.edu

Zhenkai Zhu
UCLA
zhenkai@cs.ucla.edu

Lixia Zhang
UCLA
lixia@cs.ucla.edu

ABSTRACT

In Named Data Networking (NDN), all data packets are authenticated with digital signatures. Thus a trustworthy key management system is required in all NDN applications for data validation. In this paper, we propose an endorsement-based key management system, which is inspired by the concept of Web-of-Trust, to secure ChronoChat, a serverless group chat application over NDN. With the endorsement-based key management system, users in a chatroom can collaboratively authenticate each other's membership in the chatroom. The system also leverages the synchronization mechanism provided in ChronoChat for efficient key/endorsement distribution and revocation. We further extend the key management system for user identity authentication in a chatroom to enable one user to authenticate another user's identity without resorting to any external public key infrastructure.

Categories and Subject Descriptors

D.4.6 [**Software**]: Security and Protection

General Terms

Security

Keywords

Named Data Networking, Security, Application

1. INTRODUCTION

Named-Data Networking (NDN) is designed to secure data directly by bounding together the content and the name of every data packet through a digital signature [15]. In order to verify this signature, NDN applications require a trustworthy key management system, i.e., a system that can reliably distribute public keys or public key certificates and provide proper public key authentication and revocation, etc.

In this paper, we explore the security of a recently developed serverless group chat application, called ChronoChat [18]. In ChronoChat, a chatroom is made of a group of participating users. Any user may send a chat message to the "virtual" chatroom at any time. Users synchronize their knowledge about the chat messages in order to be notified of any new chat message.

Trust is required in key management. With a trusted public key, one may effectively validate a data packet or authenticate another public key. NDN highlights *contextual trust* [9]. In ChronoChat, users in a chatroom extensively interact with each other through the synchronization. It would be natural for users to establish trust regarding to key management through their collaboration within the chatroom rather than resorting to some external third parties.

This community-based trust resembles the concept of Web-of-Trust (WoT), within which a user may establish trust on another user through a web consisting of introductions made by others. Such transitive trust requires well-defined semantics, otherwise an introduction made by a user may not be correctly interpreted by others.[1]

In this paper, we propose an endorsement-based trust model in which the concept of Web-of-Trust is explicitly expressed through NDN semantics. We also design a key management system based on the endorsement-based trust model, as a working and verifiable WoT system, to manage keys in ChronoChat.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the security semantics of NDN. We introduce ChronoChat and discuss its security requirements in Section 3. The endorsement-based trust model is presented in Section 4. We elaborate how to build an endorsement-based key management system to manage chatroom membership and authenticate user identity respectively in Section 5 and Section 6. A comparative analysis is made in Section 7 and related work is discussed in Section 8. We conclude the paper in Section 9.

[1]See a discussion in IETF mailing list as an example: http://www.ietf.org/mail-archive/web/ietf/current/msg82073.html

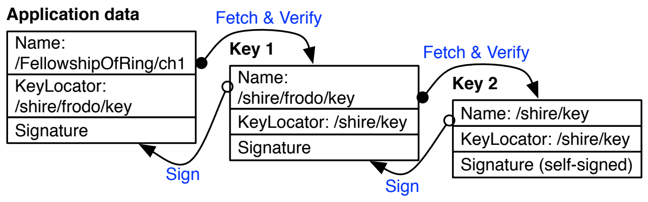


Figure 1: An example of data validation. The last key is directly trusted and is represented as a self-signed key.

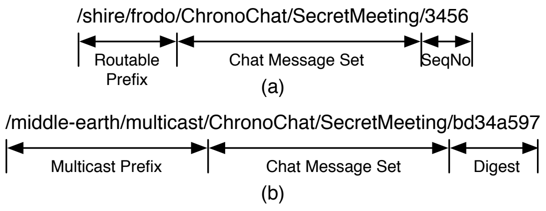

/shire/frodo/ChronoChat/SecretMeeting/3456
Routable Prefix
Chat Message Set
SeqNo
(a)
/middle-earth/multicast/ChronoChat/SecretMeeting/bd34a597
Multicast Prefix
Chat Message Set
Digest
(b)

Figure 2: Naming rules of ChronoChat: (a) an example of chat message name; (b) an example of sync interest name.

2. NDN SECURITY

NDN security is data-oriented. The content and the name of a data packet are bound together through a digital signature. A data consumer validates data packets according to a specific trust model, which usually answers two questions:

- Who can sign a particular data packet?
- How to authenticate the signer?

Assume that data packets are signed using public key cryptography. The answer to the first question is actually a set of conditions that a trusted signing key must satisfy. For example, in Figure 1, one may specify two rules: 1) any data packet published under the name prefix `/FellowshipOfRing` must be signed by a key with the name `/shire/frodo/key` and 2) any data packet published under the name prefix `/shire` must be signed by a key with the name `/shire/key`. There is always, of course, an implicit condition that a trusted signing key must be valid.

To answer the second question, a consumer may directly specify the public key bits of a trusted signing key. The signer of a data packet can be authenticated if the data signature can be verified using the public key. However, it is not always feasible for a consumer to have a priori knowledge about the signing key of all the received data packets. In this case, a consumer needs to retrieve the public key bits of a signing key according to the key-locator information specified in a data packet.

A public key that is retrieved from the network needs to be authenticated. In the example in Figure 1, a consumer directly trusts the key `/shire/key`. The consumer, when receiving a data packet signed by another key `/shire/frodo/key`, needs to retrieve the public key and authenticate it using the directly trusted key `/shire/key` according to the rules specified above. Once Frodo's key is authenticated, the consumer can use it to validate the original data packet.

Note that key authentication process is the same as the process of data validation. Actually, a consumer, by specifying answers to the two questions in its trust model, can explicitly define the expected key management.

A key, as a special type of content, is bound to a name, which is also called the *identity* of the key. How to interpret the identity of a key is up to the data consumer and also depends on the context. For example, a consumer may simply take the holder of a private key with the identity `/shire/frodo` as the legitimate owner of the namespace `/shire/frodo` and use the corresponding public key, with high confidence, to validate the data packets published under the namespace. Another user may interpret the identity `/shire/frodo` as someone who is member of the "the fellowship of the ring" and trust it to publish data packets under the prefix `/FellowshipOfRing`. Note that it requires a context to associate data packets with a key of an identity that is different from the data name prefix. In the example above, the consumer can only make the association within the context of "The Lord of the Ring".

With the digital signature and key name, a data packet that carries the public key bits of a signing key essentially becomes a certificate. We call a data packet that simply vouches for one's identity *identity certificate*.

3. SECURITY FOR CHRONOCHAT

We first briefly introduce ChronoChat and then discuss its security requirements.

3.1 ChronoChat

ChronoChat is a serverless group chat application in NDN. A chatroom is made of a group of users. Any user in the group may send a message to the chatroom at any time. All the chat messages sent to the same chatroom constitute a chat message set.

ChronoChat defines following naming rules for the data packets that carry chat messages [18]. Each chatroom has an name, e.g. `SecretMeeting`. Given a chatroom, the name of the chat message set is constructed by appending the chatroom name to the application tag `ChronoChat`, e.g., `/ChronoChat/SecretMeeting`. Each user has its own routable prefix, e.g. `/shire/frodo`. The name of a chat message produced by a user consists of three parts: Part (1) is the user's own routable prefix, so that interests for the chat message can be ef-

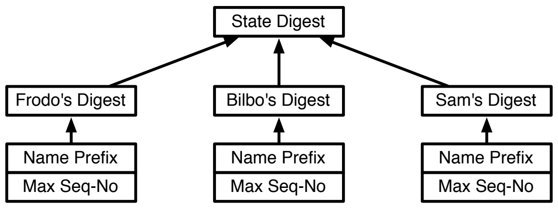


Figure 3: An example of digest tree

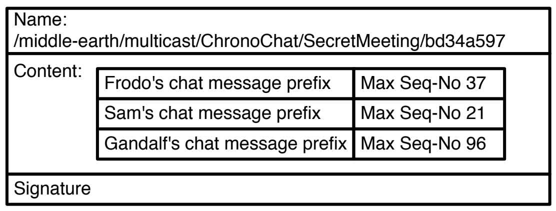

Name:
/middle-earth/multicast/ChronoChat/SecretMeeting/bd34a597
Content:
Frodo's chat message prefix
Max Seq-No 37
Sam's chat message prefix
Max Seq-No 21
Gandalf's chat message prefix
Max Seq-No 96
Signature

Figure 4: Example of a sync reply.

fectively forwarded toward the producer. Part (2) is the name of the chat message set, so that the interest can be demultiplexed immediately once it reaches the data source. Part (3) is a unique sequence number assigned to the chat message. The sequence number automatically increases by one whenever the user produces a new chat message. Figure 2(a) shows an example of chat message name.

Users synchronize their knowledge about a chat message set through ChronoSync [17], a distributed synchronization protocol over NDN. Using ChronoSync, users exchange their knowledge about the chat message set through an interest, called *sync interest*. Figure 2(b) shows an example of sync interest name. The name consists of three parts: 1) a multicast routing prefix, 2) the name of the chat message set, and 3) a crypto digest that represents the interest sender's knowledge about the chat message set. The crypto digest is calculated by compressing the user's knowledge through a digest tree as illustrated in Figure 3. Each leaf of the digest tree represents the status of a particular user, including the user's chat message prefix and the latest sequence number. The first two parts together is called the *sync prefix* of a chatroom. All the users in a chatroom should listen to the sync prefix, so that they can receive the sync interests from each other.

When a user receives an sync interest, the user can infer the status updates that the interest sender is missing (refer to [17] for more details about the inferring process) and replies the interest with a data packet, called *sync reply*, which contains all the missing updates. Figure 4 shows an example of sync reply. Leveraging the data multicasting feature of NDN, the sync reply is propagated back to all the users who send the same sync interest and trigger them to update their knowledge about the chat message set and fetch the missing chat messages.

3.2 Security Requirements in ChronoChat

In ChronoChat, both sync replies and chat messages need to be secured. If sync replies are not secured, an attacker may force users to fetch malicious data from a fake "user". An attacker, by maliciously introducing a huge number of new users, may also expand the digest tree to exhaust legitimate users' memory. If chat messages are not secured, attackers can easily launch impersonation attack.

Although both sync replies and chat messages are signed, they are not secured if users do not know the correct public keys to verify the signature. We assume that legitimate users in a chatroom, i.e., members of a chatroom, will honestly participate in the conversation. In other words, a member should honestly report the status updates of itself and other members in sync replies and publish chat messages under its own namespace. With this assumption, a key management system must satisfy following two requirements, so that the two types of data packets can be validated:

- The key management system should support membership authentication in a chatroom. As long as a user's membership is authenticated, the user can be trusted to publish sync replies.
- The key management system should also support user identity authentication. Since chat messages are published under each user's own namespace, a chat message can be validated as long as the public key of the publisher is securely bound with the corresponding namespace.

4. TRUST THROUGH ENDORSEMENT

In this section, we present the endorsement-based trust model, as our attempt to develop a systematic WoT-based solution with clearly defined trust relations. This trust model will be used to build a key management system for membership authentication and user identity authentication in ChronoChat, as we will demonstrate in the next two sections.

The foundation of the endorsement-based trust model is the direct trust between two users. Here we define the direct trust as: *a user* A *trusts another user* B*, without any premise, to publish data packets under a name prefix* P. Direct trust also implies that user A has already authenticated the public key of user B through some offline channels, so that user A can directly validate a data packet under the Name P if the data packet is signed by user B.

When a user expresses its direct trust to others, the direct trust becomes an *endorsement*. An endorsement associates the endorsee (which is usually specified through its identity certificate) with a data name prefix. The endorser, by signing the endorsement, asserts that the endorsee can be trusted to publish data packets under the name prefix.

Note that the identity of the endorser and the endorsee do not have to be related to the data name prefix. In other words, *anyone* can endorse *any* others for *any* name prefix. Whether an endorsement is acceptable is determined by a data consumer's own trust settings and the context. Consumers with different trust settings or in different contexts may interpret the same endorsement in different ways.

Unlike some existing WoT systems in which the trustworthiness of a directly trusted user is expressed in terms of a value [5,10], we define the trustworthiness of directly trusted users in terms of three trust levels with explicitly specified semantics. Given a particular name prefix, the trust settings of a data consumer is made by classifying the consumer's directly trusted users into three trust levels:

- Level-1: a user is trusted by the consumer to only publish data packets under the name prefix;
- Level-2: in addition to Level-1 trust, a user is also trusted by the consumer to endorse another user's capability of publishing data packets under the name prefix;
- Level-3: in addition to Level-2 trust, a user is also trusted by the consumer to delegate another user the power of endorsement regarding to the name prefix.

The Level-2 trust enables a user to accept an endorsement from a directly trusted user, while the Level-3 trust allows a user to establish trust on an indirect endorser and accept its endorsements.

When a user expresses its trust settings to others, the trust settings become a set of endorsements for each directly trusted user. The trust level of each directly trusted user is also included in the endorsement.

Users need to collect the endorsements made by others to derive indirectly trusted users (including indirect endorsers). The mechanism of obtaining or distributing endorsements depends on specific applications. Ideally, within a given application context, one should collect as many endorsements as possible to obtain a complete view about the "Web-of-Trust".

In order to discover indirect endorsers within a particular context,[2] one needs to construct delegation chains using the Level-3 endorsements made by others. Each delegation chain starts from a directly trusted user at the trust Level-3. A delegation chain is extended through the Level-3 endorsements and ends at an indirect endorser who does not further delegate the power of endorsement. All users along a delegation chain are trusted regarding to the name prefix implied by the context.

[2]Indirectly trusted users can be easily derived from indirect endorsers

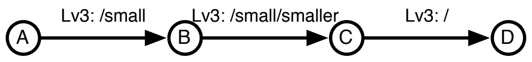


Figure 5: An example of endorsement delegation. Each arrow represents a Level-3 endorsement. The data name prefix included in each endorsement is labeled above the arrow. An endorsement also reflects the endorser's trust settings.

A user may change its trust settings occasionally and need to revoke its endorsements. An endorsement revocation should be distributed to all users that are using the endorsement. These users, on receiving the recovation, should remove the revoked endorsement from its delegation chains and re-evaluate the indirectly trusted users.

We would like to highlight three features of the endorsement trust model: contextual trust, decentralized trust, and redundant endorsement paths.

4.1 Contextual Trust

The name prefix specified in an endorsement provides context information for data validation. In such a context, the endorsee's identity certificate is implicitly associated with the data name prefix. As a result, a user may be allowed to publish data packets under specified name prefix with its own identity certificate. Although the name of a data packet may be irrelevant to the identity of the data publisher, a recipient of the data packet, with the corresponding context information, can still validate the data packets according to the publisher's identity.

Context also restricts the scope of trust. A user cannot be trusted when it is out of context. The context of a directly trusted user is specified in user's trust settings, while the context of an indirectly trusted user is derived as the intersection of the context of the user's endorser and the context of the endorsement. As a result, the context of users along a delegation chain is monotonically shrinking and is never beyond the context of the initial directly trusted user. Figure 5 shows an example of endorsement delegation. User A directly trusts user B for name prefix `/small`. When user B delegates the power of endorsement to user C, it further restricts the context of user C through specifying a more specific name prefix `/small/smaller` in the endorsement. Although user C can endorse user D for a broader context (`/`), the context of user D, from the per-

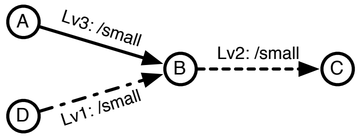


Figure 6: An example of different trust settings. An arrow represents an endorsement which also reflects the endorser's trust settings. All endorsements are in the same context `/small` but are at different trust levels.

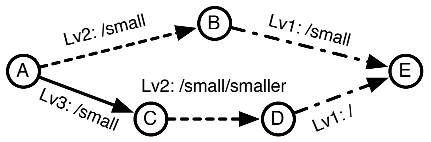


Figure 7: An example of multi-path delegation. An arrow represents an endorsement which also reflects the endorser's trust settings.

spective of user A, is still constrained within a narrow context `/small/smaller`.

4.2 Decentralized Trust

Instead of relying on the trust decisions made by a centralized system, a user in the endorsement-based trust model can customize its trust settings to reflect its own trust preference. A user can configure its own set of directly trusted users. This allows a user to boostrap trust from those users that the user knows privately.

Even for the same directly trusted user, different users may classify it into different trust levels. Figure 6 shows such an example. User B is classified into different trust levels by user A and D respectively. User D trusts user B as a normal data producer, while user A directly trusts user B to delegate the power of endorsement. As a result, user A will accept user B's endorsement for user C, but user D will reject user B's endorsement and treats user C as an untrusted user.

4.3 Redundant Endorsement Paths

Since the trust scope is restricted by the context rather than the name of endorser, users in the endorsement-based trust model are free to endorse any others. As a result, one can establish indirect trust on another through different endorsement paths. Figure 7 shows an example of multiple endorsement paths. Redundancy of endorsement paths can significantly improve the robustness of the key management system based on this trust model.

If a user can be indirectly trusted through multiplate paths, the trust scope of the indirectly trusted user is evaludated as the union of the trust scope of each endorsement path. In the example shown in Figure 7, the trust scope of user E should be evaludated by combining the trust scope of the two endorsement paths. Thus the trust scope of user E, from the perspective of user A, is `/small`.

5. MEMBERSHIP MANAGEMENT

Membership management associates the membership of a chatroom to the public key of participating users. In this section, we elaborate how to fit the membership management into the endorsement trust model.

Before designing an endorsement-based key management system for membership management, two questions need to be answerd:

- Who controls the membership of a chatroom?
- How to manage the membership of a chatroom?

For the first question, it would be natural to ask users in the chatroom to manage the membership by themselves, otherwise users are forced to trust some third parties outside the chatroom. For the second question, since users in a chatroom share the ownership of the chatroom, the membership of the chatroom should be collaboratively managed by all users. Centralized membership management is intentionally avoided, because the member who controls the membership can arbitrarily kick a member out of the chatroom and other users have to get the permission from the controller before introducing a new member.

We propose to manage the membership of a chatroom as follows: a new member can be introduced into the chatroom by any existing members; a user's membership is revoked if none of existing members vouches for it.

Next, we will first introduce the design of membership endorsement and then explain how to implement the decentralized membership management based on it.

5.1 Membership Endorsement

In order to fit the membership management into the endorsement trust model, we assume that

- To become a member of a chatroom, one must obtain at least one membership endorsement from an existing member.
- Existing members trust each other to endorse another's membership of the chatroom.

The second assumption implies that all the membership endorsements in a chatroom must be Level 3 endorsements. Because once a user has been admitted into a chatroom, the user aquires the same power of endorsement as the existing members. This can be viewed as

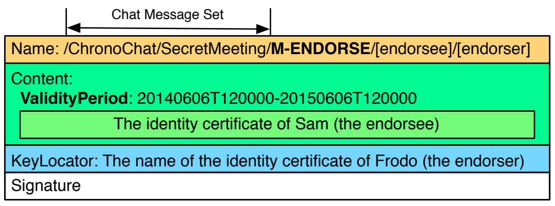

Chat Message Set
Name: /ChronoChat/SecretMeeting/**M-ENDORSE**/[endorsee]/[endorser]
Content:
ValidityPeriod: 20140606T120000-20150606T120000
The identity certificate of Sam (the endorsee)
KeyLocator: The name of the identity certificate of Frodo (the endorser)
Signature

Figure 8: An example of membership endorsement

the new member's endorser (which is an existing member) delegates its power of endorsement to the new member. As a result, trust regarding to membership endorsement is completely transitive in a chatroom.

The second assumption also implies that an existing member may endorse another existing member. Such an endorsement may reinforce the endorsee's membership.

We define an membership endorsement in terms of a data packet as shown in Figure 8. The data name serves as a brief description of the endorsement. The name starts with the name of the chat message set which provides the context of the endorsement. After that, a special name component `M-ENDORSE` is used to indicate that the data is a membership endorsement. The endorsee's identity certificate name is encoded into one name component, so is the endorser's. These two components are sequentially appended to the end of the data name.

The content of an endorsement contains two pieces of information: 1) the validity period of the endorsement and 2) the endorsee's identity certificate. The endorser, by signing the endorsement data, asserts that, in this particular chatroom, the endorser trusts the endorsee as a legitimate user during the specified period.

5.1.1 *Bootstrap*

As we introduced in Section 4.2, users in the endorsement trust model start with their own trust settings. Each user endorses the membership of its directly trusted users in a chatroom. A user, through the endorsements made by its directly trusted users, can authenticate the membership of some indirectly trusted users. The user may further extend the roster of a chatroom through the endorsements made by the indirectly trusted users.

The roster of a chatroom, from the perspective of a user, is extended in terms of a spanning tree which is rooted in the user's own key. Figure 9 shows examples of the endorsement tree of two members in a chatroom.

5.1.2 *Mutual Endorsement*

Endorsement is directional. Given an endorsement, the endorser takes the endorsee as a member in a chatroom, but the endorsee may not do so in the other way around. Since endorsement trees are extended from each user's own public key, different users' endorsement trees may contain different sets of nodes, i.e., the roster of the chatroom may be inconsistent among members in the same chatroom.

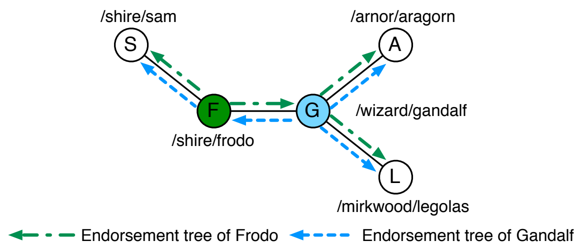


Figure 9: Examples of endorsement tree in a chatroom `SecretMeeting` with five members. Nodes in color are the roots of two endorsement trees respectively.

In order to maintain a consistent roster among users in a chatroom, we use *mutual endorsement* to further restrict the membership management. That is, any two members in a chatroom either directly endorse each other's membership or indirectly endorse each other's membership through a chain of direct mutual endorsements.

Mutual endorsement is a strong requirement in membership management. It prevents an user from being unilaterally introduced into a chatroom. When a member in a chatroom endorses a user outside the chatroom, the endorsement serves as a invitation. The invited user, on accepting the invitation, must endorse the inviter back as a confirmation.

5.2 Adding New Members

Introducing a user into a chatroom takes two steps: 1) an existing member invites the user; 2) the inviter distributes the membership endorsements in the chatroom if the invitation is accepted.

5.2.1 *Chatroom Invitation*

An invitation to a user implies that the inviter (an existing member) has already trusted the user to behave as a legitimate user in the chatroom. A successful invitation also implies that both inviter and invitee have already authenticated each other's identity through offline channels (e.g., through private email exchange, face-to-face meeting, etc.).

We define a chatroom invitation in terms of an interest packet. The interest name consists of three parts: Part (1) is the invitee's routable name prefix, so that an invitation is effectively forwarded toward the invitee. Part (2) is a special name component `ChronoChat-Invitation` to indicate that the interest is an invitation. Part (3) is a name component which encodes a membership endorsement for the invitee. Figure 10

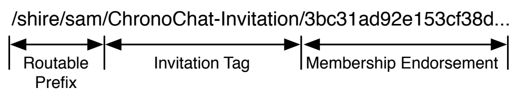


Figure 10: Example of invitation interest name. The last component encodes the membership endorsement for the invitee (e.g., Sam).

shows an example of invitation interest name. The first two parts together is called *invitation prefix*

ChronoChat users, by listening to the invitation prefix, can receive the invitation from others. Since the membership endorsement in the invitation is signed by the inviter, the invitee can validate the invitation using the inviter's public key which has been authenticated offline. The invitee, on accepting the invitation, should reply the invitation with a data packet containing the membership endorsement for the inviter. Thus both the inviter and the invitee are mutually endorsed.

With the direct mutual endorsements, the invitee's endorsement tree is extended to include the inviter, so does the inviter's. The inviter then exports all of its collected endorsements to the invitee, so that the invitee's endorsement tree can be further extended to all the other members in the chatroom. As a result, the invitee can immediately authenticate all the members in the chatroom.

5.2.2 *Endorsement Distribution*

The inviter should notify all the other members of the mutual endorsements made during the invitation phase, so that they can authenticate the invitee's membership. In order to deliver the endorsement notification as soon as possible, we utilize the synchronization mechanism for chat message notification.

In the chatroom's digest tree, a new leaf is created for each user to manage their endorsements. The name prefix of the new leaf is the user's chat message prefix appended with a special name component `ENDORSE`. A data packet published under this prefix is called *endorsment wrapper*. Figure 11 shows an example of endorsment wrapper. With the endorsement wrapper, endorsement notifications are distributed in the same way as chat message notification.

The inviter can encapsulate the mutual endorsements into an endorsement wrapper, all the other members, after fetching the wrapper, can authenticate the new user's membership immediately.

The membership endorsements between existing members are distributed in a similar way, except that a member ony needs to encapsulate its own endorsement in an endorsement wrapper. An membership endorsement received from an existing member also serves as a mutual

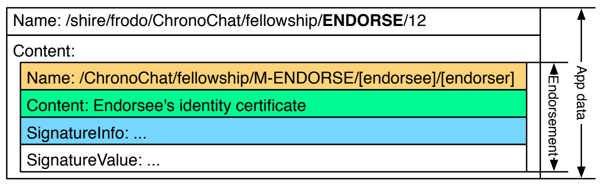

Name: /shire/frodo/ChronoChat/fellowship/ENDORSE/12
Content:
Name: /ChronoChat/fellowship/M-ENDORSE/[endorsee]/[endorser]
Content: Endorsee's identity certificate
SignatureInfo: ...
SignatureValue: ...
Endorsement
App data

Figure 11: An endorsement data encapsulated in an endorsement wrapper.

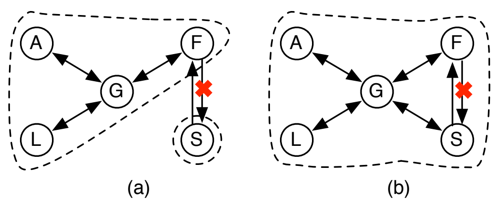


Figure 12: Result of endorsement revocation. User F revokes its endorsement on user S: (a) S is expelled from the chatroom; (b) S retains its membership due to its mutual endorsement with user G.

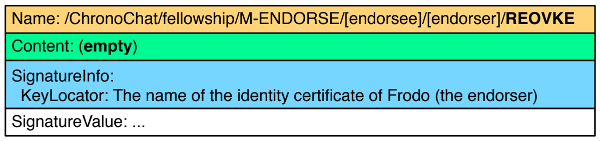
endorsement request from that member. The endorsee of the endorsement, if accept the request, can make a reverse endorsement as a confirmation.

5.3 Expelling Existing Members

Besides introducing new users into a chatroom, members should also be able to expel some members when necessary. As we described above, one retains its membership through the its direct mutual endorsements with some other existing members. Depriving a user's membership is equivalent to revoking the user's membership endorsements. Figure 12(a) shows an example of expelling a member through endorsement revocation.

As we mentioned eariler, a user's membership can be reinforced through the membership endorsements made by multiple existing members, so that a single endorsement revocation may not completely isolate the user from the rest of the chatroom. For example, member S and G are mutually endorsed in Figure 12(b), the endorsement reovcation made by member F does not affect the S's membership. As a result, expelling a user from a chatroom may require all the endorsers of the user to take an action.

We define an endorsement revocation in terms of a data packet as shown in Figure 13. The name of a revocation data is constructed by appending a special name component `REVOKE` to the name of the revoked endorsement. Since the whole purpose of the revocation data has been expressed in its name, the content of revocation data is left empty. The endorser, by signing the


Name: /ChronoChat/fellowship/M-ENDORSE/[endorsee]/[endorser]/**REOVKE**

Content: **(empty)**

SignatureInfo:
KeyLocator: The name of the identity certificate of Frodo (the endorser)

SignatureValue: ...

Figure 13: An example of endorsement revocation

revocation data, states that the corresponding endorsement is no longer valid.

5.3.1 *Revocation Distribution*

Same as distributing membership endorsement, membership endorsement revocations are distributed through the synchronization mechanism provided in ChronoChat. Revocation data is encapsulated into the same wrapper as the one for the endorsement data. The synchronization semantic allows members to receive the revocation at the first time, thus providing stronger protection.

A membership endorsement revocation also serves as a request to expel the endorsee of the revoked endorsement from the chatroom. A user, on receiving a membership endorsement revocation, should check whether it has endorsed the user to expel. If the user has made such an endorsement before, the user needs to decide whether to revoke the endorsement or not. As we mentioned above, a user will be expelled only when none of existing members is willing to endorse its membership.

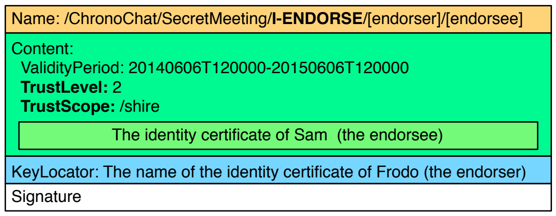
Note that a member, when inviting a new user, should also export the endorsement revocation to the invitee. As a result, every member should also keep a copy of endorsement revocations.

6. IDENTITY AUTHENTICATION

The membership management only prevents users from receiving false status updates of a chatroom. Users still need to fetch and validate chat messages separately. Since chat messages are published under each user's own namespace which is associated with the user's public key through an identity certificate, validating a chat message is equivalent to authenticating the publisher's identity. In this section, we elaborate how to authenticate a user's identity through the endorsement trust model.

Recall that a member joins a chatroom through an invitation from an existing member. An invitation implies that both inviter and invitee may have already establish trust, to some extent, on each other regarding to the identity authentication. At least, both of them should have authenticated each other's identity, i.e., trust Level-1. Such a direct trust regarding to identity authentication can be leveraged for identity endorsement.

Same as membership endorsement, the context of an


Name: /ChronoChat/SecretMeeting/**I-ENDORSE**/[endorser]/[endorsee]

Content:
ValidityPeriod: 20140606T120000-20150606T120000
TrustLevel: 2
TrustScope: /shire
The identity certificate of Sam (the endorsee)

KeyLocator: The name of the identity certificate of Frodo (the endorser)

Signature

Figure 14: An example of identity endorsement. In this example, Frodo trusts Sam to endorse any user with an identity under the namespace `/shire`, but Frodo does not trust Sam to delegate the endorsment power to others.

identity endorsement is also restricted within the chatroom. That is, an identity endorsement becomes invalid when it goes out of a particular chatroom or the endorser is not a member of the chatroom. However, unlike membership authentication in which members trust each other to endorse the membership of another. A member may trust the other members' identity endorsement in different trust scopes. For example, one may trust a member with the identity `/shire/frodo` to endorse all the members with an identity under the prefix `/shire` but does not trust the same member to endorse members with other identities. Therefore, besides the normal context (i.e., the chatroom), a user's trust settings regarding to identity authentication should include a specific trust scope for each directly trusted users. Same as the normal context, the trust scope for identity authentication also monotonically shrinks along the delegation chain.

Moreover, a member may trust different members at different levels. Unlike membership endorsements, we allow an identity endorsement to be a Level-1 or Level-2 endorsement. For example, one may trust only one member to endorse the identity of the others. In this case, in order to prevent the directly trusted member from delegating the power of endorsement to others, the member's trust level should be set to Level-2. Therefore, when one converts its trust settings into identity endorsements, both trust scope and trust level should be explicitly included in each endorsement.

Figure 14 shows an example of identity endorsement. The name of an identity endorsement resembles the name of a membership endorsement. The first two name components provide the normal context of the identity endorsement. The special name component `I-ENDORSE` is used to distinguish identity endorsements from membership endorsements (which use `M-ENDORSE`). Note that the content of identity endorsement data includes two more properties, `TrustLevel` and `TrustScope`, to reflect the endorser's trust settings.

Same as membership endorsement, identity endorsements are distributed and revoked through the synchronization in a chatroom.

6.1 Customized Trust

A user's trust settings for identity authentication could be highly configurable. It is possible that a member's identity can be authenticated by some users but fails to be authenticated by others. Such a failure, however, is expected because it faithfully reflects the user's own trust preference. In order to increase the chance of authenticating the identity of every member in a chatroom, a user may tune its own trust settings or adopt some other key management systems (e.g., a hierarchical key management system [1]) as complements.

We should also point out that an identity authentication failure has a limited impact on affected users. Since membership authentication is separated from identity authentication, the affected users can still synchronize the knowledge about the chat message set with others. The only consequence is that the affected users may not see the unvalidated messages or see them marked "unvalidated".

7. ENDORSEMENT vs. HIERARCHICAL

We have implemented the endorsement-based key management system in ChronoChat.[3] In this section, we perform a comparative analysis of the endorsement based key management system by comparing it against a naming hierarchy based key management system which was adopted by NDNS (NDN Domain Name System) [1].

If the hierarchical system is used to secure ChronoChat, each member in a chatroom gets two certificates: one certificate, which is used to validate sync replies, associates the member's public key with the chatroom name; the other certificate, which is used to validate chat messages, associate the member's key with the member's own identity. The two certificates are issued respectively by the one who manages a parent namespace of the certificate name.

The two systems are compared on four aspects: 1) third party dependency, 2) robustness, 3) key revocation, and 4) complexity. On the first aspect, a user of the hierarchical system is required to unconditionally trust at least one third party, the root key, while a user of the endorsement system trusts those who are known privately and the trust is contextual.

For robustness, we compare the consequence of a single point failure. In the hierachical system, an invalid parent certificate in the hierarchy causes all the keys below it unverifiable. The consequence of a single point failure depends on the location of the failure. In contrast, the endorsement system supports multi-path authentication, so that the failure point may be bypassed. However, we should admit that the robustness of the endorsement system may vary from one user to another, depending on a user's own trust settings and the collected endorsements. A user with very strict trust settings and a limited collection of endorsements may still suffer from a single point failure.

[3]For source code, refer to `https://github.com/bruinfish/ChronoChat-V2`

For complexity, we compare the overhead of authentication. In the hierarchical system, keys are authenticated on demand and a user only needs to store authenticated keys. In contrast, a user of the endorsement system has to update the set of authenticated keys whenever an endorsement is received and needs extra storage for endorsements.

For key revocation, we compare the attack window caused by a revocation, i.e., the gap between the moment when a key is revoked and the moment when a user is notfied of the revocation. In the hierachical system, each key has a freshness period, a user does not query the revocation status of a key before the key becomes stale. Therefore, the attack window depends on the freshness of a key and could be more than a day. In contrast, in the endorsement system, all users are immediately notified of a revocation through the synchronization and the attack window is much smaller. However, we should point out that it is easy to revocke a single endorsement but revoking a user's membership may require all members in a chatroom to reach an agreement.

In summary, the endorsement based key management system provides contextual trust and may be more robust than the hierarchial key management system, but at the cost of more authentication overhead. Moreover, the system provides more efficient endorsement revocation through synchronization, but it may take extra steps to revoke a user's membership of a chatroom.

8. RELATED WORK

NDN security has been studied in many research efforts, ranging from building automation systems [4] to distributed storage systems [1]. Some works investigated the network layer security, such as interest flooding [2] and DDoS attack [7]. In this paper, we focus on the application layer. There are many efforts in application level security [4, 14, 19]. These works either assumed the existence of a trustworthy key management system [19] or adopted a hierarchical trust model [1, 4]. In contrast, the endorsement-based key management in this paper implements the concept of the Web-of-Trust.

Key management systems have been intensively studied in IP network. Some systems, such as Kerberos [12], provided symmetric key based authentication. In this paper, we aim at public key based authentication. Among the deployed public key management systems, most are based on a hierarchical trust model, such as CA [8] and DNSSEC [3]. Some systems, such as Vantages [13], PERSPECTIVE [16] and Convergence [11], explored the concept of "security through publicity". The key management system proposed in this paper share the spirit of the third category of key management systems which is based on the Web-of-Trust. Some existing Web-of-trust based key management systems follow the standard defined in OpenPGP [5]. Web-of-Trust were also proposed to manage the trust in peer-to-peer systems over the IP network [6].

9. CONCLUSION

In this paper we explored the concept of Web-of-Trust to secure ChronoChat, a serverless group chat application over NDN. In order to develop a systematic WoT-based solution with clearly defined trust relations, we proposed an endorsement-based trust model. The endorsement-based trust model provides explicitly specified endorsement semantic (such as context, trust level, and trust scope). Within this model, users can clearly state their trust settings and accurately interpret the endorsements made by others, so that trust becomes transitive among users in a chatroom.

Based on the endorsement-based trust model, we designed a key management system for ChronoChat. Since the endorsement-based trust model enables the transitive trust, the key management system allows users in a chatroom to collaboratively manage the chatroom membership and to authenticate the identity of other members within the chatroom, thus eliminating any dependency on any external public key infrastructure.

The endorsement-based trust model is one of our initial steps to explore a variety of trust models in NDN. We hope that the endorsement-based trust model could become an complement to existing trust models for NDN applications.

10. REFERENCES

[1] A. Afanasyev. *Addressing Operational Challenges in Named Data Networking Through NDNS Distributed Database*. PhD thesis, UCLA, 2013.

[2] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang. Interest flooding attack and countermeasures in named data networking. In *IFIP Networking Conference*, 2013.

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Dns security introduction and requirements. RFC 4033, March 2005.

[4] J. Burke, P. Gasti, N. Nathan, and G. Tsudik. Securing instrumented environments over content-centric networking: the case of lighting control. In *Proceedings of IEEE INFOCOMM 2013 NOMEN Workshop*, 2013.

[5] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. Openpgp message format. RFC 4880, November 2007.

[6] P.-A. Chirita, W. Nejdl, M. T. Schlosser, and O. Scurtu. Personalized reputation management in p2p networks. In *ISWC Workshop on Trust, Security, and Reputation on the Semantic Web*, 2004.

[7] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. Dos and ddos in named data networking. In *ICCCN*, 2013.

[8] R. Housley, W. Ford, W. Polk, and D. Solo. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, May 2008.

[9] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *CoNext*. ACM, 2009.

[10] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW*. ACM, 2003.

[11] M. Marlinspike. Convergence. http://convergence.io/.

[12] B. C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *Communications Magazine, IEEE*, 1994.

[13] E. Osterweil, D. Massey, B. Tsendjav, B. Zhang, and L. Zhang. Security through publicity. In *HOTSEC*, 2006.

[14] V. Perez, M. T. Garip, S. Lam, and L. Zhang. Security evaluation of a control system using named data networking. In *NPSec*, 2013.

[15] D. Smetters and V. Jacobson. Securing network content. Technical report, PARC, 2009.

[16] D. Wendlandt, D. G. Andersen, and A. Perrig. Perspectives: Improving ssh-style host authentication with multi-path probing. In *ATC*. USENIX, 2008.

[17] Z. Zhu and A. Afanasyev. Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking. In *ICNP*. IEEE, 2013.

[18] Z. Zhu, A. Afanasyev, C. Bian, V. Jacobson, and L. Zhang. Chronos: Serverless multi-user chat over ndn. Technical report, UCLA, 2012.

[19] Z. Zhu, J. Burke, L. Zhang, P. Gasti, Y. Lu, and V. Jacobson. A new approach to securing audio conference tools. In *Asia Workshop on Future Internet Technologies*, 2011.