NDN-MPS: Supporting Multiparty Authentication over Named Data Networking

Zhiyi Zhang
UCLA
Los Angeles, USA
zhiyi@cs.ucla.edu

Siqi Liu
UCLA
Los Angeles, USA
siqi.liu@ucla.edu

Randy King
Operant Networks
Santa Rosa, USA
randy.king@operantnetworks.com

Lixia Zhang
UCLA
Los Angeles, USA
lixia@cs.ucla.edu

ABSTRACT

Modern digitally controlled systems require multiparty authentication to meet the security requirements. This paper describes the design and development of NDN-MPS, an automated solution to support multiparty signing and verification for NDN-enabled applications. NDN-MPS proposes three basic changes to the existing NDN security solutions. First, it introduces a new type of trust schema that supports the semantics for multiparty singing and verification. Second, it extends the NDN signing process design to accommodate multisignature schemes such as BLS signature and to ensure the data consistency across signers. Third, NDN-MPS provides options for different application scenarios to coordinate the joint signing process of multiple signers. We evaluate NDN-MPS by assessing its security properties and measuring its performance. Our results show that NDN-MPS provides resistance against different types of attacks and is practical to be deployed.

CCS CONCEPTS

• **Networks** → **Security protocols**; • **Security and privacy** → **Authentication**.

KEYWORDS

multiparty authentication, multisignaure, named data networking

ACM Reference Format:
Zhiyi Zhang, Siqi Liu, Randy King, and Lixia Zhang. 2021. NDN-MPS: Supporting Multiparty Authentication over Named Data Networking. In *8th ACM Conference on Information-Centric Networking (ICN '21), September 22–24, 2021, Paris, France.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3460417.3482971

1 INTRODUCTION

For reasons such as multiparty contractual policies and mitigating single points of failure, many real-world systems require a joint decision-making process where multiple parties are involved. For example, in grid-connected distributed energy resources (DER) systems, the conventional way of having single proprietary connections to secure grid assets will no longer be sufficient due to the business model change [17, 35]. Therefore, smart devices like solar inverters will have multiple parties, including customers, manufacturers, grid operators, who need to access and send jointly-approved remote commands, often over the public Internet.

Recently, Named Data Networking (NDN) [38] started being explored to provide secure networking support for DER systems. The diversity and number of stakeholders and DER service provider business models [17] require a multiparty trust model with expressive and flexible trust policies. While NDN has developed supporting mechanisms to secure producer-consumer communications by using cryptographic signature schemes such as RSA and ECDSA, the existing NDN security solutions [42] support single party data signing only. The security model considers only two types of parties: a consumer (*verifier*) authenticates its received data generated by a producer (*signer*) based on given policies.

Compared to the existing single party authentication solutions, supporting multiparty authentication brings up three different aspects. First, multiparty signing involves third-party signers in the system who need to sign data produced by others. Second, the verifier will need to verify the data's signatures against a given set of signers, so a new type of trust schema is needed to define the acceptable set of signers that can jointly produce a legitimate signature. Third, effective and secure coordination among multiple signers is required to ensure the correctness of the joint signing process.

Regarding the cryptographic implementation, a straightforward approach to multiparty authentication would be collecting a list of signatures from individual signers. However, this approach may suffer from high network overhead and low efficiency, because a verifier needs to retrieve and validate all individual signatures. Multiparty signature schemes like BLS [7] and its later variant [4] provide an efficient alternative, with which individual signatures can be aggregated into a single one (*i.e.* signature space complexity from *O(n)* to *O(1)*) and the aggregated signature only requires one verification operation (*i.e.* verification time complexity from *O(n)* to *O(1)*).

Contribution. This paper proposes NDN-MPS, a multisignature based multiparty authentication toolset over NDN. NDN-MPS provides applications with the support of signing, verification, and orchestration of the signing process among multiple signers. More specifically,

- NDN-MPS develops a new type of trust schema that supports the semantics for multiparty signing and verification. The schemas can also realize threshold-based policies (*e.g.*, legitimate if signed by any k out of n signers).
- NDN-MPS designs a new type of key locator scheme to accommodate multiple signers and ensure the consistency of data object to be signed in the multiparty signing process.

ICN '21, September 22–24, 2021, Paris, France

ACM ISBN 978-1-4503-8460-5/21/09.
https://doi.org/10.1145/3460417.3482971

- NDN-MPS provides two options to coordinate the joint signing among multiple signers, a Remote Procedure Call (RPC) based signature collection protocol and the use of an existing NDN sync protocol. We discuss how they may be used in different application scenarios and what are the trade-offs.

Our design is based on the latest NDN Specification v0.3. We have implemented NDN-MPS in C++ with the multisignature scheme BLS [7]. The library can be easily extended to other non-interactive multisignature schemes such as MSP multisignature [4] and the Bitcoin ECDSA threshold signature [13, 15].

NDN-MPS makes use of the network and security support of NDN in several aspects. First, the Interest and Data exchange allows the asynchronous transmission of unsigned and signed packets in the multiparty signing process. Second, NDN lets each entity have its own identity managed by the local authority and uses trust schemas to explicitly define the trust relationship. This simplifies the development and deployment of multiparty authentication solutions compared to today's network security practice, where user authentication mostly relies on the public key infrastructure which is largely managed by commercial certificate authorities (CAs) and the application layer credentials (*e.g.* username and password) that are verified by a centralized service.

Outline. In the rest of the paper, we first introduce the background of our work and an example application scenario that will be used throughout the paper in §2. We then define the system model of the NDN-based multiparty signing and verification in §3. In this section, we will also analyze the trust schemas needed by the model and describe the security objectives and technical challenges in realizing this model in NDN After that, we present the design of NDN-MPS in §4 and describe NDN-MPS's support of threshold policies in §5. We then assess the security properties of NDN-MPS in §6, evaluate NDN-MPS in §7, and discuss a number of design choices and issues in §9. Finally, we conclude our work in §10.

2 BACKGROUND

2.1 NDN Producer-Consumer Model

The producer-consumer trust model is adopted by the existing NDN security support. Under this model, each NDN data object is assembled, named, and cryptographically signed by its producer. To help consumers to verify the data, each data object carries a signature information field to keep the meta information of the signature; the most important information is the key locator, indicating which key should be used to verify the signature value. A key locator is usually the name of the verification key so that a consumer can fetch the corresponding key to verify the received data.

Verifying the signature against a trustworthy public key only ensures authenticity. To further verify the authorization of the producer, a trust schema is used to determine whether the producer is legitimate to produce the named data (Figure 1). NDN's semantic naming enables the trust schemas to use names to systematically define the relationship between the signing key and data object of each step in an acceptable certificate chain. Trust schema rules for a specific application system usually contains the following elements: (i) the format or structure of the data names that the rules apply to, (ii) the name pattern of the expected signing key name at each step in the certificate chain, (iii) the acceptable trust anchors that certificate chains end with. If a signature is generated by a party whose key name is not allowed for signing such data or the signer's identity cannot be certified by an acceptable certificate chain to a trusted anchor, the signature will be rejected.

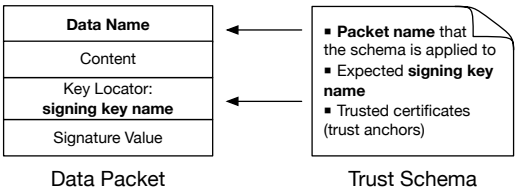


Figure 1: Trust schemas are based on names

Importantly, the producer-consumer trust model only *assumes two parties*, the producer and the consumer, both of which are directly involved with the data.

In a multiparty signing scenario, the existing trust scheme is insufficient to clearly describe the policies of the new trust model because (i) there are multiple signers, (ii) (at least some) signers are not the producer of the data to be signed; and (iii) multisignature requires a new party that coordinates the signing, which the trust schemas should clearly define. The current protocol specification and implementation of NDN libraries also need enhancement to support multiparty signing. For example, the current key locator cannot carry sufficient meta information of multiple signers, and the trust schema does not support the signing and verification of multisignatures.

2.2 Multisignature Scheme

We follow the definition of Boneh *et al.* [4, 6] and define the multisignature scheme that is used for multiple parties to sign the same message with the following algorithms:

Parameter Generation. The system generates the public system parameters $param$.

Key Generation. Each signer generates its key pair $(pk, sk) \leftarrow KeyGen(param)$.

Individual Sign. Each signer generates a signature of a given message m: $\sigma \leftarrow Sign(sk, m)$.

Signature Aggregation. Multiple signatures can be aggregated into one signature: $\sigma' \leftarrow SigAgg(\sigma_1, ..., \sigma_n)$

Verification. An aggregated signature can be verified with the signers' public keys: $VALID\ or\ INVALID \leftarrow Verify(\sigma', m, pk_1, ..., pk_n)$.

An example construction is the BLS signature scheme [6, 7]. Compared with other multisignature schemes like identity-based aggregate signatures [14] and fast multiparty threshold ECDSA signatures [12], BLS does not require additional negotiation among signers nor a trusted key generator in the key generation step. In addition, an arbitrary number of BLS public keys and signatures can be easily aggregated without knowing the private keys. For these reasons, BLS is more practical than other schemes and thus is used in the design description in the rest of this paper.

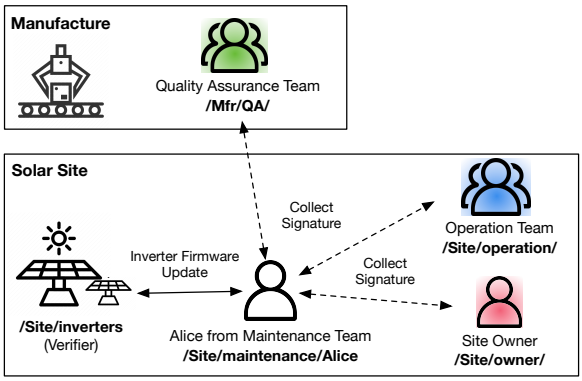


Figure 2: Solar inverter firmware update requires multiparty authentication

2.3 An Example Application Scenario

We introduce an example scenario used throughout the paper to ease the reader's comprehension of various concepts. In a grid-connected distributed solar network system, solar devices are managed by a local solar site for power production. At the same time, the manufacture of solar devices provides technical support of the device firmware update.

Consider a typical use case where the manufacturer requests the solar site to upgrade an inverter's firmware to correct a security vulnerability. To make the update, the solar site's maintenance operator, Alice "`/Site/maintenance/Alice`", will need to issue a command, an NDN data object, to be jointly signed by the following parties:

- The solar site's operation team to ensure they know the inverters will be offline for some time. Their prefix is "`/Site/operation`"
- The site's owner will permit the possible production reduction, if the process cannot be completed promptly. The owner will have a prefix "`/Site/Owner`".
- An operator from the equipment manufacturer's quality assurance department for certifying the upgrade is of release quality. This is a third party and thus its prefix "`/Mfr/QA`" is different from other parties.

The inverters will verify the signed data to ensure that it can safely proceed with the update before executing the firmware update.

Therefore, in this command issuance process, Alice is the producer, the inverter is the verifier, and the signers are "`/Site/operation`", "`/Site/Owner`", and "`/Mfr/QA`". Note that under each prefix, there can be multiple entities; for example, under "`/Mfr/QA`" there can be "`/Mfr/QA/operatorX`" and "`/Mfr/QA/operatorY`".

3 NDN MULTISIGNATURE BASED MULTIPARTY AUTHENTICATION MODEL

With multisignature, there are three basic types of parties in each multiparty authentication process:

- **Producer** The producer generates the data object that will be signed by the signers. We consider that by default, the producer will also collect signatures from the signers and then aggregate them into a multisignature to finish the data production. Nevertheless, in certain scenarios, other parties allowed by the application systems can also collect and aggregate signatures.
- **Signers**. Signers will sign the data object created by the producer individually. The signers can come from different organizations and have different trust anchors. If a signer approves the data to be signed, either through in-band operations (*e.g.*, database query) or out-of-band operations (*e.g.*, human decisions), that individual will make a signature piece for aggregating with other signers' pieces.
- **Verifier**. The verifier will verify a multisignature along with the data object covered by the signature. The verification succeeds if the multisignature is valid and if its signers satisfy the application's requirements as defined in the trust schema.

The above roles are separated based on their functional logic. In practice, one entity can serve multiple roles. Normally, a producer is expected to be one of the signers given the producer is responsible for the content and is directly involved in the signing process.

Under the system model, the multiparty authentication process can be abstracted into the following steps.

(1) The producer decides the signers based on the trust schema and publishes the data object to these signers.
(2) Each signer first obtains the data object and then authenticates the data object against the trust schema. After that, each signer will verify the data object against the application logic (*e.g.* defined by the business model). If all checks are successful, the signer will generate the signature and publish the signature value.
(3) The producer (or any authorized party allowed by the trust schema) collects and aggregates all the signature pieces into one, and appends it to the data object to finalize the data production.
(4) The verifier obtains the final data object and verifies its signature against the signers' public keys.

Trust Relationships. We first assume that all entities have gone through the standard NDN bootstrapping process [42], through which each entity will obtain identity with a certificate issued by the application system and install the system's trust anchors.

To finish the multiparty signing process, it is required that signers can authenticate the producer's identity, for example, by verifying a signature over the unsigned data object that is generated by the producer's identity private key. To verify an aggregated signature, the verifier needs to obtain the public key certificates of all the signers. Nevertheless, multiparty signing does not require that signers know each other.

These trust relationships are defined in the NDN trust schema by the application system and are securely installed to the entities.

Other Assumptions. We assume all the signers' key types should be compatible with NDN-MPS and generated with the same public parameters (*e.g.* the same pairing-friendly elliptic curve, the same type of hash function, etc.), so that their signatures can be aggregated. We also assume the producer's awareness of the signers from which the producer can select appropriate signers to generate legitimate signatures.

This model does not restrict how packets are delivered in the signing or verification process, which makes our model general

and can be used in different application scenarios. For example, the model does not specify how should a verifier obtain the final data object; in the solar inverter example, the final data object can be sent to the inverter as a signed Interest packet, or be fetched by the inverter as a Data packet, or be synchronized with an NDN sync protocol.

3.1 Trust Schema Needed by the Model

As stated, the trust relationships are defined in the trust schema. To be more specific, the following policies should be explicitly and rigorously defined.

- **Producer Trust Schema**. When verifying the data object created by the producer, the producer trust schema can decide whether the producer is legitimate to generate such a data object.
- **Multisignature Trust Schema**. When verifying the final data object, the multisignature trust schema can specify how many and which signers can generate a legitimate signature for a given named data.

Note that these trust schema rules do not only guide the verification process but also can facilitate the signing process. To be more specific, (i) a producer with multiple identities can use the producer trust schema to decide which identity should be used to deliver the data object to signers; (ii) the producer can also use the multisignature trust schema to decide the list of signers; (iii) a signer can also check the multisignature trust schema whether her signature is truly needed for such a data object.

It is noteworthy that when deciding whether to sign a data object, the signer will not only query trust schema but also the application logic. This is because the trust schema does not guarantee the correctness of the data content. In the solar inverter example, when deciding whether to sign the inverter firmware update command, the maintenance team needs to ensure that, when the inverter is offline, the remaining power supplies can still meet the demand, *e.g.*, by querying some databases and calculators. If not, the signer should reject the data object.

Producer Trust Schema. When delivering the data object from the producer to signers, a regular signature generated by the producer is sufficient for authentication. Therefore, the producer trust schema can be expressed using the existing trust schema languages or tools [25, 34]. In our solar inverter example, the trust schema defines that the signature request must originate from maintenance operators, who are under the prefix of "`/Site/maintenance`" and derived from the anchor certificate "`/Site/maintenance/KEY/123`". The pseudo-code of such a trust schema policy is shown below.

```
Signature Request Name:
  /<SignerPrefix>/MPS/request/Site/maintenance/<operator>/<>
Key Name: /Site/maintenance/<operator>/KEY/<>
Anchor: /Site/maintenance/KEY/123
```

Multisignature Trust Schema. The multisignature trust schema requires additional semantics for legitimacy verification of multiple signers for a single signature. In particular, a multisignature trust schema should contain three pieces of information.

- **Data Profile**. A profile specifies which data object the policy should be applied to. In NDN, the data name or the name pattern with the packet type (*i.e.*, Interest or Data packet) will suffice.
- **Legitimate Signer List**. A legitimate signer list specifies the signers required to make a legitimate signature for the given data. This list can be effectively expressed by a list of the signers' names or name patterns.
- **Known Signers**. The known signers are the complete set of all potential signers in the system, a subset of which can make a legitimate signer list for a given signature. The known signers are similar to the trust anchor field in Yu *et al.*'s schematized trust design [37] with the difference that in a multisignature trust schema, each known signer's identity (*i.e.*, name and public key) can be either directly listed as a trust anchor or indirectly included if it can be authenticated by a known trust anchor with an acceptable certificate chain.

In the solar inverter example, the data profile will match the name of the firmware update command Data packet, the legitimate signers are the prefixes of three required parties, and the known signers are all the candidate signers from three required parties.

In some application scenarios, the legitimate signer list is not specified; instead, a threshold of the number of signer is given. To be more specific, a threshold-based policy can express the legitimate signers as any k out of n known signers. We will discuss how to extend the syntax of multisignature trust schemas to support threshold-based later.

3.2 Security Objectives

The security objectives of a multiparty authentication system can be summarized as follow.

- Authentication of multiple signers, which is the basic goal of the system. When illegal signers are present or the signers do not satisfy the trust schema, the authentication must be rejected.
- Integrity and authenticity of the multiparty signing process. When coordinating multiple signers, the integrity and authenticity of (i) the data object and (ii) signatures from individual signers should be ensured.
- Confidentiality of the multiparty signing process. In a traditional producer-consumer model, the content of a data object is not available before the data is published by the producer. In a multiparty signing process, the same level of confidentiality should be achieved, considering the adversaries may increase their advantages by obtaining useful information before the signed data is finalized.
- Basic resistance to denial-of-service and record-and-replay attacks in the multiparty signing and verification process. For example, a recorded request asking for a signature from a signer should not result in a new signature, and a recorded aggregated signature should not be used in a new data object.

3.3 Technical Challenges

Signature Coverage in Multiparty Signing. To be aggregated correctly, each signer's signature should cover identical data for the compactness of the finalized data. However, by the latest specification (*i.e.* NDN Packet Format Specification version 0.3) of the signing process, the signer will customize the signature information file of the data packet by writing the signing key name into the key locator field [32]. A key locator contains the signing key

information to help the verifier to check against the trust schema and use the correct public key for verification. Consequently, each signer will sign the data with customized signature information, thus leading to inconsistent data signed by different signers and a broken final signature after an aggregation.

A strawman solution is to let the producer pre-provision the signature information, and each signer signs without modifying it. However, it is insufficient because the signer list cannot be finalized before contacting the signers, considering a signer may be offline or reject to sign the data. For example, to meet the requirement of a signer under the prefix "`/Mfr/QA`", assume the producer Alice decides to contact "`/Mfr/QA/operatorX`" and thus put this name into the key locator field. However, operator X may be offline, and thus Alice needs to find an alternative signer, say "`/Mfr/QA/operatorY`". As such, Alice needs to change the signature information halfway through the signature collection and lead to the inconsistency of the signed data.

In this work, we address the issue with a late-binding key locator name as described in §4.3.

Threshold Signature. Threshold signature scheme [16] is a type of multiparty signature, which supports a flexible access structure: instead of explicitly defining the signer combinations for a legal signature, k arbitrary signers out of a signer set of size n can jointly generate a legal signature. A common approach to support the threshold scheme is to directly utilize threshold signature schemes. For example, a typical construction of threshold signature schemes is to combine traditional signature schemes with Linear Secret Sharing [28]. However, this solution suffers from a few disadvantages. First, to bootstrap the system, either (i) a centralized trusted dealer is usually needed for key distribution or (ii) the signers need to exchange a large number of messages for setting up their keys in a distributed manner. In addition, the key maintenance is complicated because when a key is compromised, or when there is a new signer, it may require a new bootstrapping process where all signers must be online and update their keys. Lastly, the key aggregation is relatively slow, which takes $O(k^2)$ traditionally and $O(k \log k)$ [36] to perform the interpolation to construct the secret.

As described in §5, NDN-MPS takes a system approach with the multisignature trust schema, bypassing the aforementioned issues.

4 DESIGN OF NDN-MPS

We describe the design of NDN-MPS by first giving an overview and then introducing each main component of the NDN-MPS.

4.1 An Overview

NDN-MPS is designed as a security support toolset for applications. NDN-MPS consists of three main components (Figure 3). First, a new type of trust schema, multisignature trust schema, that allows applications to express the signing and verification policies for multiparty authentication. Second, an extension of the key locator in NDN data objects to keep the to-be-signed data consistent across signers and allow complex signature information. Third, a signature collection protocol, which is based on RPC or NDN sync, for the producer (or any other authorized party) to collect signatures from multiple signers.

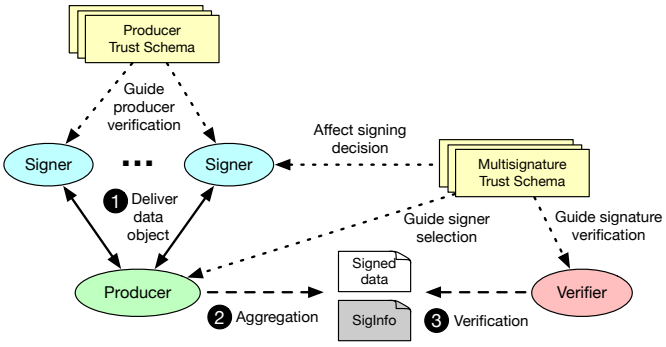


Figure 3: An Overview of NDN-MPS

These three components are used in the process of multisignature signing and verification. Note that while it may not be the producer who collects signatures from signers, but for simplicity, in the rest of this section, *we assume the collection is done by the producer.*

Multiparty Signing. The producer starts a signing process with the unsigned data object and the multisignature trust schema provided by the application. From the trust schema, the producer identifies a list of signers and then starts the coordination process. In a nutshell, it consists of (i) the delivery of the data object to be signed and (ii) collection of signature value from signers. NDN-MPS allows applications to select from an RPC based approach or an NDN sync based approach.

- When using the RPC based approach, the producer will issue a signed request to each signer to start an RPC and presents the unsigned data as the request parameter. Once the signature is generated, the producer can fetch it as the RPC result.
- When using an NDN sync protocol, it is required that the producer and all the signers are already in the same sync group. The producer publishes the data object to be signed and then each signer publishes the signature value. The producer or any signer can synchronize to the latest state and collect all the signatures.

There are tradeoffs between the two options: The RPC based approach puts more reliance on the producer; if the producer goes offline, the signing process will fail. Nevertheless, this approach is simple and the overhead is low. In contrast, the sync based approach requires a sync group, which indicates additional overhead for maintaining the sync group, *e.g.*, group membership management, security of the group messages, periodic message exchange for synchronization, and so on. However, the sync based approach allows any group member to get all the signatures and thus prevents the producer to be a single point of failure.

To ensure the consistency of the data to be signed among signers, the producer will fill the key locator field with a placeholder name called late-binding key locator. Such a placeholder allows the data to be signed deterministic and consistent across multiple signers. When the signers are finalized, for example, after all the individual signatures are collected, the producer can produce a Data packet whose name is the placeholder name and whose content is the signer information.

Upon receiving the request, each signer will first verify the producer's identity against the producer trust schema. After that, the

signer can decide on whether to sign the data or not. In this process, besides checking against known multisignature trust schema policies, the signer may also need to go through some application-specific procedures. If the application agrees to sign the data, it will generate the signature over the data, without modifying the key locator or any other fields.

In an unstable network or in an attack, signers may be forced offline, unable to be reached by the producer. In this case, the producer can try to reach an alternative signer permitted by the trust schema, *e.g.*, the producer Alice can try "`/Mfr/QA/operatorY`" when "`/Mfr/QA/operatorX`" is unavailable. This design provides a certain degree of resiliency that even if some signer fails, the signing can continue. Importantly, the change of signers will not affect the value of the late-binding key locator name as it is a placeholder, so the consistency of the data being signed among signers is preserved.

Signature Aggregation. After collecting sufficient signature pieces from signers, the producer aggregates them to generate a single signature and attach it to the original unsigned data (Figure 4). In addition, the producer will generate another Data packet, called a SigInfo packet, to carry the signer information, *i.e.*, signers' key names. Importantly, SigInfo is named with the placeholder key locator name. As such, a SigInfo packet can be linked from the signed data's key locator, overcoming the limitation of the existing key locator design, which does not support multiple signers.

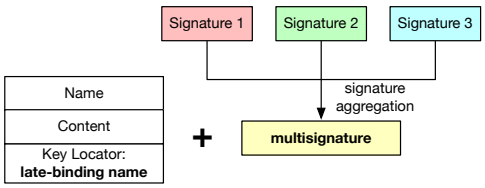


Figure 4: Aggregate signatures and finalize the data object

Signature Verification. To verify a piece of multiparty signed data, the verifier first needs to obtain both the signed data and the SigInfo packet. The signed data retrieval process is defined by the specific application and the SigInfo packet can be fetched with an Interest carrying the key locator name specified in the signed data. After that, the verifier first checks the signer list from the SigInfo packet against the multisignature trust schema to ensure the signers are legitimate to sign such a piece of data. Finally, the verifier validates the multisignature using the public keys of the signers.

4.2 Multisignature Trust Schema

As the baseline, the multisignature trust schema should allow applications to specify multiple required signers; that is, a signature is valid only when required signers R_1 to R_n are present. To support this, multisignature trust schema rule has an attribute called "all-of" to hold the list of the key name of R_1 to R_n. In a verification process, if any key listed in the "all-of" attribute is missing, the signature is considered invalid.

In many scenarios, applications do not need to or cannot require specific signers; instead, they only need a certain type of signers. For example, in the solar network system, any operator from the manufacture's quality assurance department can represent that party. To support such flexibility, NDN-MPS's multisignature trust schema allows the use of name patterns. To be more specific, when specifying a required signer, the application can use wildcard character $*$, which can match any name component. Using the same example, the wildcard character in "`/Mfr/QA/*/KEY/*`" allows any operator under the equipment manufacturer's quality assurance team to participate in the joint signing.

The pseudo-code of a multisignature trust schema for the example use case is shown below.

```
Data profile: /Site/inverters/firmware/update
All-of {    /Mfr/QA*/KEY/*
            /Site/operation/*/KEY/*
            /Site/Owner/*/KEY/*    }
Known-signer {
   ...
}
```

The rule applies to the command to update the solar inverter firmware. The known signers should be the registered operators under each responsible organization's prefixes, which are abbreviated for simplicity.

4.3 Late-binding Key Locator Name

As stated, since the signers may not be finalized during the multiparty signing process, it is infeasible for the producer to include the signer list in the original data object. At the same time, the signature information of the data object, including the key locator, needs to be deterministic and consistent among all signers so that signature pieces can be aggregated. To address the issue, NDN-MPS lets the producer put a placeholder Data packet name as the key locator. Such a placeholder is called a late-binding key locator name.

This name is late-binding because there is no Data packet bound to this name at the time of the signature collection phase. After completing the signing process, the SigInfo packet containing the signature information will be published under the key locator name. This packet typically contains the key name list of the signers; information that can potentially facilitate the verification process, such as the trust schema name or the aggregated public key, can also be included. The SigInfo packet is also signed by the producer for data integrity. Note that an adversary cannot cheat with a forged SigInfo packet because due to the underlying multisignature primitive, only correct signers' public keys can lead to a successful verification.

4.4 Remote Procedure Call based Coordination

The first option to coordinate multiple signers is to apply RPC between the producer and each signer, where the producer is the caller and the signer is the executor. A straightforward solution is to directly apply existing NDN-based RPC protocols between the producer and each signer, for example, RICE [18], the RPC used in DNMP [25], and Named Service Call (NSC) [20]. However, while RICE provides sufficient security and privacy protection, it requires modification of the underlying NDN forwarding pipeline. DNMP RPC requires an NDN synchronization protocol which is excessive. In comparison, NSC is lightweight and requires no modification of the NDN forwarding, but it does not provide payload encryption

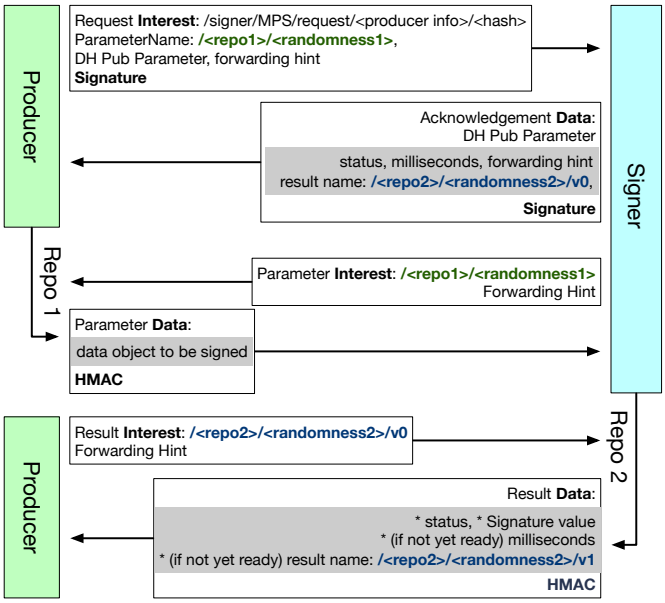


Figure 5: Remote procedure call in NDN-MPS (The gray area refers to the encrypted payload)

and result confidentiality as needed by NDN-MPS. Therefore, in this work, we extend the design of NSC to fit our needs.

We design the NDN-MPS RPC for signature collection based on NSC but provides stronger security and privacy. NDN-MPS has the following features that are different from the original NSC.

- Besides signed Data packets, Interest packets are also signed for authenticity and integrity. Producer trust schema will be applied when a signer verifies the first RPC request from the producer.
- Diffie-Hellman key exchange protocol [11] is applied to negotiate a per-RPC shared secret for packet payload encryption. As such, all the sensitive information exchanged in the RPC is encrypted for privacy protection.
- The RPC parameter (*i.e.*, the data object to be signed) and the result (*i.e.*, individual signature value) will be wrapped into Data packets whose name is meaningless and content is encrypted for stronger confidentiality.

Through the NDN-MPS RPC, attackers cannot know what content is signed in the signature collection and have no idea about the result of each RPC. The design of NDN-MPS RPC is visualized in Figure 5.

RPC Request and Acknowledgment. To start the signing process, the producer first expresses a signed request Interest to the signer. This Interest carries a name, called $ParaName$, pointing to a parameter Data packet, which will encapsulate the unsigned data object. Such a name comprises randomly generated components, revealing no information about the unsigned data object. In addition, a forwarding hint to the parameter packet should be carried as well. To provide resistance to traffic analysis, the parameter Data packet can be published to a third party repository, and accordingly, the forwarding hint should point to the repository. In addition, the Interest will carry the public parameters used for the Diffie-Hellman key exchange.

On receiving the request Interest, the signer will first verify the producer's signature against the producer trust schema. Then, the signer will also generate a pair of public and private keys for the Diffie-Hellman key exchange, and derive from the negotiated secret a message authentication code (MAC) key K_{mac} and a symmetric encryption key K_{enc}. A key derivation function is usually used together with the Diffie-Hellman key exchange for better security. After that, the signer will generate an acknowledgment Data packet containing the following information. (i) A status code indicating whether the request is accepted. Since the signer has not seen the data to be signed, the status mainly relies on the producer identity verification result and the signer's availability. (ii) The estimated time for the RPC result to be available. (iii) The name of the result Data packet called $ResultName$. This name is supposed to be randomly generated. Similar to a parameter packet, to prevent information leakage by traffic analysis, the result Data can be published to a third-party repository, and a forwarding hint to that repository will be added to the acknowledgment Data as well. Importantly, all the above parameters are encrypted with the K_{enc}. In addition, the acknowledgment Data also contains the signer's public parameters for Diffie-Hellman key exchange in plain text. The acknowledgment packet will be signed by the signer's private key and replied to the RPC request the Interest packet.

If the signer rejects the RPC request, the status code in the acknowledgment will indicate the reason. To prevent an adversary from learning the rejection, the signer can pad the Data packet content to the size similar to a success response, and send an Interest to retrieve the parameter packet.

Parameter Retrieval. After replying with the acknowledgment, the signer will issue an Interest to fetch the parameter.

On the producer side, upon receiving the acknowledgment packet, the producer first verifies the signature, calculates the same K_{mac} and K_{enc} from the shared secret established in the Diffie-Hellman key exchange, and decrypts the payload carried in the acknowledgment Data content. Then, the producer will finalize the parameter Data packet by (i) naming it as $ParaName$, (ii) encrypting the unsigned data object with K_{enc} into the content field of the Data packet, and (iii) signing it with K_{mac}. The producer will make the parameter Data packet available, *e.g.*, by publishing it to an NDN repo, and it will be fetched by the signer.

Result Retrieval. After knowing the $ResultName$ and time for fetching the result from the acknowledgment packet, the producer can schedule a data fetch accordingly.

After fetching the parameters, if the signer agrees to sign the data object, a signature value will be generated. Note that the signer will not modify any field of the data object, including the signature information field. Then the signer will (i) encrypt the signature value with K_{enc}, (ii) wrap the it into a Data packet named as $ReaultName$, and (iii) signed with K_{mac}. The result Data packet will then be published, *e.g.*, on to an NDN repo, and fetched by the producer.

4.5 NDN Sync based Coordination

The second option to coordinate the multiparty signing in NDN-MPS is to utilize NDN sync. Moll *et al.*'s work [23] provides a good survey of existing NDN sync protocols. In general, any NDN sync protocol with group membership management and group-level

packet encryption is sufficient. Some of the sync protocols already utilize packet encapsulation where the published Data packets are wrapped into sync Data packets. Sync Data packets are named with a different naming convention and do not reveal information of the encapsulated Data packet. With packet encryption, the adversaries cannot learn the signing process by eavesdropping on the packet exchange in the sync group.

In this section, we use State Vector Sync (SVS) [22] as an example to illustrate the process. SVS uses Data packets named with the sequence number to encapsulate Data packets created by group members. In addition, symmetric key encryption schemes like AES can be easily applied to the group messages.

Prerequisites. The producer and all signers should already be in the same sync group. The group manager should ensure that no unauthorized parties can join this group. An encryption key should be negotiated within the group to encrypt all the group messages.

Data Object Publication. The producer publishes the data object with a list of signer names to the sync group. As mentioned, the data object should contain the late-binding key locator. In addition, both of them should be signed by the producer. After receiving the data object and the list, if a signer is listed, the signer should publish an acknowledgment indicating whether she agrees to sign it or not. Through this process, the producer can update the signer list to add or remove signers accordingly.

Signature Result Publication. A signer who agrees to sign the data object will publish the signature value when it is ready. Through the NDN sync protocol, the producer will be notified when a new Data packet is generated by the signer. The producer can then synchronize and fetch the signature. After collecting all individual signatures, the producer can move on to aggregate the signatures and finalize the data object.

5 SUPPORT OF THRESHOLD POLICY

NDN-MPS provides support of threshold policy by a system means. To be more specific, the multisignature trust schema allows the application to specify the threshold used for the signature verification process. Specifically, in NDN-MPS multisignature trust schema, applications are allowed to specify the threshold k in an attribute called "at-least-num" and the information of n signers in the attribute "from". In a verification process, only when k out of n signers listed in the "from" participant in the signing, the signature is considered valid.

Consider a typical application scenario in a solar network system: to minimize the human-made faults in critical operations, certain commands like power plant shutdown require at least two managers' approvals before it can be operated. In this case, we have $k = 2$ and the identities of all the team managers in the system will form the signer list in the "from" attribute. The multisignature trust schema is shown below (known signers omitted for compactness).

```
Data profile: /site/operation/command/shutdown/*
At-least-num 2
From {     /Site/operation/manager1/KEY/*
           /Site/operation/manager2/KEY/*
           /Site/operation/manager3/KEY/*
           /Site/operation/manager4/KEY/*  }
```

More flexibility can be supported by combining the regular multisignature policy and threshold policy. To be more specific, applications can specify that a signature needs m required signers and k out of n optional signers. Using the same solar shutdown command example, besides any two manager's approvals, the command may also require the owner's awareness. For this purpose, the policy can be written as follows (known signers omitted for compactness).

```
Data profile: /site/operation/command/shutdown/*
All-of {   /Site/Owner/*/KEY/* }
At-least-num 2
From {     /Site/operation/manager1/KEY/*
           /Site/operation/manager2/KEY/*
           /Site/operation/manager3/KEY/*
           /Site/operation/manager4/KEY/*  }
```

Compared with the solutions that rely on additional cryptographic schemes like secret sharing or dedicated threshold signatures, NDN-MPS's use of multisignature trust schema enjoys following benefits:

- It does not rely on the underlying signature scheme. Thus, there is no need for applications to support a dedicated threshold signature scheme specially for threshold scenarios, reducing the overall key management complexity.
- There is no centralized trusted dealer for secret sharing. Each signer's key can be managed individually. For example, a signer's key can be renewed or revoked without bothering other signers.
- It supports more flexible policies as described, in which both certain signers and k out of n are required. In contrast, using a threshold signature scheme to support such policies is nontrivial.

6 SECURITY ASSESSMENT

We start the security analysis of NDN-MPS against the desired properties as stated in §4.4 Then, we consider several attacks potentially conducted by attackers.

Multiparty Authentication. The verifier can authenticate the signers of the aggregate signature and judge whether the signature is legitimate by the features of the underlying multisignature scheme and the multisignature trust schema. To be more specific, the signature verification will guarantee the following properties:

- The signers are qualified to generate a legitimate signature required by the multisignature trust schema.
- The signature value can be verified by public keys of the claimed signers. This is ensured by the underlying multisignature scheme in which the signature can only be verified with the known signer keys defined in Section 3.1.
- The data object covered by the signature has not been altered. If data has changed after the signature pieces are created, the signature value cannot be verified successfully.

If signature verification succeeds, it means all the involved signers are authenticated successfully.

Integrity and Authenticity of the Signing Process. In the signer coordination process, regardless of using RPC or NDN sync, a signer can verify the producer's identity and ensure the integrity of the data object by the signature. Similarly, the producer can also verify each signer's identity and the signature value. Therefore, any alteration to the exchanged data object or signature value will be

identified and an unauthorized party cannot impersonate a producer or a signer because he does not have the right secret keys.

Confidentiality of the Signing Process. When using NDN sync protocol, confidentiality is ensured by encrypting all the messages in the sync group. Here, we mainly analyze the confidentiality of the RPC protocol used in NDN-MPS. Specifically, we assume the adversary can eavesdrop on the Interest and Data packets exchanged in the RPC between the producer and signers. The first possible failure of confidentiality can come from the plain-text information of the NDN packets, including packet names, key locator names, and the payload of the RPC request Interest and acknowledgment Data packet. We iterate the revealed information from these sources.

- Packet names contain the prefixes of producer and signers, thus revealing their identities. Since the prefix is used for forwarding, it is inevitable unless additional infrastructures are used, *e.g.* NDN-based VPN or onion routing. The rest of the name components are either statically defined by the protocol or generated from the hash function of the payload, revealing no further information than the payload itself.
- Among the plain text payload, most of the information reveals no useful application information, *e.g.*, the public parameters for Diffie-Hellman key exchange and the salt for the key derivation function. The rest of the available information is the parameter packet name and the forwarding hint. However, since the parameter packet will be encrypted and the packet name is generated irrelevantly to the data object. It will not break the confidentiality of the signing process.
- In the exchange of the RPC request, key locators can reveal the identity of the signer and the producer, but as stated, they are already revealed by the names for forwarding purposes. Therefore, the key locators do not reveal more than the public information.

Another potential means for an adversary to learn the signing process is via traffic analysis. In NDN-MPS's RPC, the parameter packets and result packets are named randomly and can be published to a third-party repository. When there exists a sufficient anonymity set[1], it is not easy for an adversary to infer the RPC result by the traffic pattern.

Therefore, by observing the public information carried in the RPC packets, an adversary cannot learn the data object nor track the result of the signing process.

Resistance to Denial-of-service Attack. An attacker may try to deny the service of a signer by flooding RPC request Interests. In general, the request Interest is signed and carries public parameters for Diffie-Hellman key exchange, which consumes a certain amount of computation at the sender side. In addition, these Interests cannot be copied from a recorded valid request, as stated before. These facts can discourage the attack.

However, the attacker may forge a large number of request Interests that are correct in the format but contain incorrect signature values or Diffie-Hellman key exchange parameters. Note that this is not a specific problem for NDN-MPS but a general issue for many protocols, including TLS. In NDN, such attacks can be mitigated by the DDoS defense mechanisms like FITT [41].

Resistance to Record-and-replay Attack. We consider a threat where an eavesdropper may record the signed RPC request Interest sent by the producer and attack a signer by replaying this Interest. Such an attack can be effectively prevented by adding a timestamp and nonce into the Interest packet signature. At the signer side, the signer can verify the timestamp of each incoming RPC request Interest against a predefined grace period. In addition, the signer can keep a small state, say a bloom filter [2], of the nonce values seen from the recent legal requests. In this way, the attacker cannot replay because there is no Interest packet whose timestamp is before the current time by the grace period and whose nonce has not been seen by the signer.

7 IMPLEMENTATION AND EVALUATION

We have implemented NDN-MPS into an open-source prototype library [40] in C++. Our library depends on ndn-cxx [31] and a BLS library [29] that has been fully examined and is compatible with the BLS API specification defined by Ethereum 2.0 [26]. The library has also been used to implement a multisignature based identity verification challenge module in NDNCERT [39]. The multisignature trust schema is implemented in the library as a trust schema configuration file parser. To support more complicated use cases where multiple legitimate signer sets are valid, one can simply create multiple trust schema files applying to the same data prefix. In the signature verification process, the signature is accepted if there is at least one trust schema rule that is satisfied. The known signers are implemented with a separate configuration file that contains a list of NDN certificates.

In the rest of the section, we report the evaluation of our NDN-MPS prototype implementation. Our evaluation result shows that

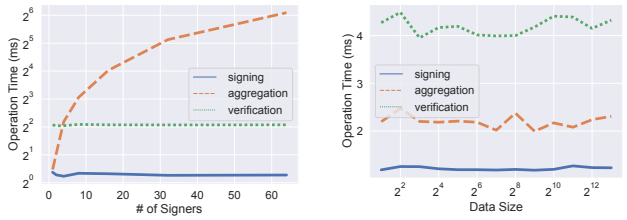
- NDN-MPS requires low-memory and low-network overhead to support multiparty authentication over NDN.
- The cryptographic operations of multisignature over NDN packets (*e.g.*, packet encoding and decoding) are computationally efficient. Thus, NDN-MPS is practical to be deployed.
- NDN-MPS provides resilience to unavailable signers.

Bandwidth and Network Overhead. A BLS signature piece from a single signer and an aggregate signature are of the same size when transmitted in the network. With BLS12-381 elliptic curve [27] (providing about 128 bits of security), the BLS signature size is 96 bytes, which is slightly larger than an ECDSA signature (about 72 bytes) and much smaller than an RSA signature (256 bytes) while providing similar bits of security. Regarding the size of public key certificates, a BLS public key over BLS12-381 is only 48 bytes, which is much smaller than public keys in ECDSA and RSA that provide a similar level of security.

The NDN-MPS RPC requires at least two round trip times (RTTs) when all signers are available. If a signer is unavailable, the producer will notice it after the first Interest's timeout. When using an NDN sync protocol, the latency is to synchronize at least two rounds of messages and the RTTs depend on the specific protocol design.

NDN Packet Signing/Verification Overhead. We compare the NDN Data packet signing and verification performance of BLS

[1] A insufficient anonymity set results in easy compromise of privacy. For example, if the producer only has contacted one signer in a given time period, it is easy to link a later Interest to the result fetch Interest. We acknowledge this assumption is less quantitative, but this is a general open issue for many privacy-enhancement systems.



(a) Different # of Signers **(b) Different data Size**

Figure 6: Scalability to Signer Number and data Size

multisignature with conventional public key signatures, including ECDSA (on elliptic curve secp256r1) and RSA (with 2048 bits key). ECDSA and RSA signatures are realized by ndn-cxx's keychain with OpenSSL [31]. The data object is 16 bytes of random data.

Operation	BLS	ECDSA	RSA
Signing an NDN packet	1.38ms	0.15ms	1.58ms
Verifying an NDN packet	4.32ms	0.43ms	0.14ms

As shown, although slower than ECDSA and RSA (OpenSSL has specialized hardware and software optimization), the performance of BLS signature in NDN is still acceptable for practical use.

Scalability to Signer Number and Data Size. We also evaluate the scalability of NDN-MPS by increasing the number of signers needed for the signature generation and the size of data to be signed. As shown in Figure 6a, only aggregation time is linearly affected by the signer number while the signing and verification time stays almost constant. As reported in Figure 6b, when increasing the data size with two signers, the time of signing, aggregation, and verification stay mostly constant.

8 RELATED WORKS

Our proposed architecture relates to two main branches of research efforts: multiparty authentication systems in today's Internet and in Information-Centric Networking (ICN).

Multiparty Authentication in Today's Internet. While the concept of multiparty signing [10] and different multisignature schemes [3, 7, 16, 30] have existed for years, most of the existing works focus on using them for compactness rather than multiparty authentication. A shred od evidence is that in most of these proposed systems, there is a lack of multiparty trust schema in the verification process. Therefore, the joint signature can be viewed as a compressed collection of unrelated signatures instead of a means to authenticate the signers for a certain data object. For example, Zhao *et al.* suggests aggregating signatures to reduce the message length for signing connection path messages in Secure BGP [43]. Maxwell *et al.* proposes the use of BLS signature in bitcoin or in general blockchain systems to save space in a block [19]. Multisignature has also been used to optimize the permissioned distributed ledger by combining multiple endorsements from different peers into one [9].

Multisignatures have also been recently used in the digital wallet design in blockchain systems, such as Bitcoin [24] and Ethereum [8], to increase security by avoiding single points of failure. The main idea is to split an account secret to storing in multiple entities (*e.g.* multiple devices owned by users). Any transaction from the wallet must be jointly signed by a certain number of secret holders [13, 15]. However, the trust policies in these systems are largely static (*e.g.*, 1-out-of-2 account always assumes a 1-out-of-2 policy), and the signature collection process is usually application-specific and not automated (*e.g.*, copy and paste signature pieces from other devices).

Multiparty Authentication in ICN. Asami *et al.* has proposed a moderator-controlled information sharing system for ICN [1]. In their system, the application uses Identity-Based Aggregate Signature to combine signatures from (i) the message producer and (ii) the message's moderator who will update the message and forward it to subscribers. However, their multiparty authentication approach is tightly coupled to their application use case, *i.e.*, a message is signed sequentially by the producer and the moderator with content modification, and is nontrivial to be used by other applications systems. In addition, due to the specific use case, their authentication does not allow applications to customize the trust schema. Also, since the moderator will modify the message, their approach will change the default NDN encoding and verification pipeline so that the aggregate signature can cover different portions of the data content. In contrast, NDN-MPS is general enough to support different applications scenarios and allows to define fine-grained multisignature trust schema.

9 DISCUSSION

9.1 Features of NDN used in NDN-MPS

We discuss how NDN-MPS benefits from the unique features of NDN. In the meanwhile, it is noteworthy that the need for multiparty authentication is required by the application logic, independent of using NDN or not.

First, the multiparty coordination can benefit from NDN's support of asynchronous communication. For example, in NDN-MPS RPC, the parameter packets or the result packets can be generated and transmitted asynchronously to tolerate long latency or intermittent connectivity and to allow the use of third party NDN repositories for high availability and privacy. Compared with TCP/IP based RFC protocols, while they can also support asynchronous result fetching at the application layer, the underlying network implementation using TCP and TLS is much more complicated (*e.g.* TCP connection state, keep-alive signals, etc.) than using connectionless Interest-Data exchange. When using NDN Sync, as already described in existing literature [23], NDN can facilitate the group communication compared with the TCP/IP architecture that discourage multicast.

Second, NDN lets each entity have its own identity and uses trust schema to explicitly define the trust relationship. This makes the development and deployment of multiparty authentication easier because the identity and trust schema can directly be used and easily adjusted for multiparty use cases. In contrast, in today's Internet, security is largely implemented in a centralized service (*e.g.*, cloud server) at the application layer, and the deployed public key infrastructure (PKI) usually relies on the commercial certificate authorities (CAs) and not all entities have identities (*e.g.*, most HTTPS clients have no identity keys and certificates). Thus, to support multiparty authentication, one either needs to rely on the

centralized application layer services, which defeats the purpose of having multiple parties, or to make considerable changes to today's PKI, which is impractical.

9.2 Selection of the Multisignature Scheme

Many multisignature schemes have been proposed. We discuss several common types of multisignature schemes and the differences when being applied with NDN-MPS.

One type of signatures requires an interactive negotiation process among signers in the key generation phase. To be more specific, the signers broadcast certain public parameters for their key pair generation so that their signatures can later be aggregated. A typical negotiation process can be based on Shamir's secret sharing scheme [28]. For example, fast multiparty threshold signature proposed by Gennaro *et al.* requires a broadcast channel for secret sharing among signers [12]. When using such a multisignature scheme, the key pairs of signers are generated together and thus when a new signer joins or a signer is revoked, the keys will be regenerated. Therefore, when used with NDN-MPS, an additional key negotiation process and key management service are needed.

Another type of multisignature schemes rely on a trusted key generator. Instead of generating one's own key pair, the trust key generator will secretly deliver the secret key to each signer. This is a common requirement when a centralized party is used in Shamir's secret sharing or the multisignature schemes are based on identity-based encryption (IBE) [5], *e.g.*, the identity-based aggregate signature scheme [14] proposed by Gentry *et al.*. When using such a multisignature scheme in NDN-MPS, an additional key generator implementation is required.

In comparison, some multisignature schemes neither require an interactive negotiation process nor any trusted key generator, *e.g.*, the BLS signature used in the prototype implementation of NDN-MPS. Thus, keys can be generated and managed individually. Note there is a known attack called rogue key attack to BLS signature but this can be addressed when a piece of additional information (called a proof of possession) is carried with each signer's public key certificate [6]. A later version of BLS can resist the rogue attack without any additional information [4]. When using such multisignature schemes in NDN-MPS, no additional components are needed.

9.3 Data Object in Multiparty Signing

NDN-MPS helps to sign a data object which can mainly be in two different forms. If the data object can fit into a single NDN Data packet, it can be directly embedded in the RPC parameter packet. A data object can also be larger than the NDN maximum packet size. In this case, the object cannot be directly embedded into one RPC parameter packet, *e.g.*, a large file or stream data. Instead, it needs to be segmented into multiple packets. To avoid multiple signers from signing each segment, we can introduce a manifest Data packet [21], which contains the hash values of all segments in the form of a Merkle Tree and their names. Then, only the manifest Data needs to be embedded in the RPC parameter packet and signed by multiple parties.

When a Interest packet needs to be signed, a data object can be an Interest packet as well, in which the multiparty signing result will be a signed Interest [33].

9.4 Design Choices in NDN-MPS RPC

We discuss some of the design choices in NDN-MPS RPC and provide the rationale behind.

Which packet should carry the parameters? In NDN-MPS, the initial request Interest to the signer contains the name of the unsigned data instead of the data itself for the following reasons. First, in the initial request Interest packet, the producer has not finished the Diffie-Hellman key exchange and thus does not know how to encrypt the unsigned data. Second, directly carrying the unsigned data does not scale well when the data is large.

How to fetch the result? NDN-MPS uses separate Interest packets for the request and the result. An alternative design would be to let the signer directly reply with the RPC result to the request Interest (*i.e.*, the first Interest packet sent by the producer in the RPC). However, this does not work when the signer's processing time is longer than the Interest timeout. Given that the producer needs to determine the signer's availability quickly, the producer cannot set the timeout of the request Interest to be too long.

How to publish the result? In NDN-MPS, the result signature and status code are encapsulated in a result Data packet. An alternative method would be to directly publish the signed data as a Data packet for fetching; however, this design may result in several issues. For one, this approach may result in cache poisoning because the result Data packet shares the same name as the Data packet carrying the aggregate signature. Additionally, directly publishing the result data can reveal the signed content and disclose that the signer has signed the data, which can break the confidentiality of the signing process. Even though the data content can be encrypted, the data name still cannot be hidden for the data retrieval purpose.

10 CONCLUSION

NDN-MPS is designed as an application-independent multisignature toolset to support and automate multiparty signature signing and verification. Our design shows how NDN-MPS leverages existing NDN features to support more complex trust models beyond the conventional producer-consumer model.

The security support of today's TCP/IP network, *e.g.*, the TLS, is limited to the point-to-point secure communication. Thus, new security models or cryptographic tools such as multisignatures can only be implemented at the application layer. In contrast, NDN's data-centric security, which is based on application layer semantic naming, enables a coherent overall system security framework, which poses no limitations to new trust models. It provides a simple yet graceful platform to explore new security models and tools with coherent network application support.

ACKNOWLEDGMENTS

This work is partially supported by the National Science Foundation under award CNS-1719403. We would like to thank our shepherd, Ken Calvert, for his valuable suggestions for improving this paper.

REFERENCES

[1] Tohru Asami, Byambajav Namsraijav, Yoshihiko Kawahara, Kohei Sugiyama, Atsushi Tagami, Tomohiko Yagyu, Kenichi Nakamura, and Toru Hasegawa. 2015. Moderator-Controlled Information Sharing by Identity-Based Aggregate Signatures for Information Centric Networking. In *Proceedings of the 2nd ACM Conference on Information-Centric Networking* (San Francisco, California, USA) *(ACM-ICN '15)*. Association for Computing Machinery, New York, NY, USA, 157–166. https://doi.org/10.1145/2810156.2810163

[2] Burton H. Bloom. 1970. Space/Time Trade-Offs in Hash Coding with Allowable Errors. 13, 7 (1970). https://doi.org/10.1145/362686.362692

[3] Alexandra Boldyreva. 2003. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In *International Workshop on Public Key Cryptography*. Springer, 31–46.

[4] Dan Boneh, Manu Drijvers, and Gregory Neven. 2018. Compact multi-signatures for smaller blockchains. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 435–464.

[5] Dan Boneh and Matt Franklin. 2001. Identity-based encryption from the Weil pairing. In *Annual international cryptology conference*. Springer, 213–229.

[6] Dan Boneh, Sergey Gorbunov, Riad Wahby, Hoeteck Wee, and Zhenfei Zhang. 2020. *BLS Signatures*. Internet-Draft draft-irtf-cfrg-bls-signature-04. IETF Secretariat. https://tools.ietf.org/html/draft-irtf-cfrg-bls-signature-04 https://tools.ietf.org/html/draft-irtf-cfrg-bls-signature-04.

[7] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short Signatures from the Weil Pairing. In *Advances in Cryptology — ASIACRYPT 2001*, Colin Boyd (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 514–532.

[8] Vitalik Buterin. 2013. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. https://ethereum.org/en/whitepaper/

[9] C. Cachin C. Stathakopoulou. 2017. *Research Report: Threshold Signatures for Blockchain Systems*. Technical Report ZUR1704-014. IBM Research – Zurich. https://dominoweb.draco.res.ibm.com/reports/rz3910.pdf

[10] Yvo Desmedt. 1987. Society and group oriented cryptography: A new concept. In *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 120–127.

[11] Whitfield Diffie and Martin Hellman. 1976. New directions in cryptography. *IEEE transactions on Information Theory* 22, 6 (1976), 644–654.

[12] Rosario Gennaro and Steven Goldfeder. 2018. Fast multiparty threshold ECDSA with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1179–1194.

[13] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. 2016. Threshold-Optimal DSA/ECDSA Signatures and an Application to Bitcoin Wallet Security. In *Applied Cryptography and Network Security*, Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider (Eds.). Springer International Publishing, Cham, 156–174.

[14] Craig Gentry and Zulfikar Ramzan. 2006. Identity-based aggregate signatures. In *International workshop on public key cryptography*. Springer, 257–273.

[15] Steven Goldfeder, Rosario Gennaro, Harry Kalodner, Joseph Bonneau, Joshua Kroll, Edward Felten, and Arvind Narayanan. 2015. Securing Bitcoin wallets via a new DSA/ECDSA threshold signature scheme. http://stevengoldfeder.com/papers/threshold_sigs.pdf

[16] Lein Harn. 1994. Group-oriented (t, n) threshold digital signature scheme and digital multisignature. *IEE Proceedings-Computers and Digital Techniques* 141, 5 (1994), 307–313.

[17] Jay Tillay Johnson. 2021. *Recommendations for Distributed Energy Resource Access Control*. Technical Report. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

[18] Michał Król, Karim Habak, David Oran, Dirk Kutscher, and Ioannis Psaras. 2018. Rice: Remote method invocation in icn. In *Proceedings of the 5th ACM Conference on Information-Centric Networking*. 1–11.

[19] Gregory Maxwell, Yannick Seurin, and Pieter Wuille. 2019. Simple Schnorr multi-signatures with applications to Bitcoin. , 2139-2164 pages. https://doi.org/10.1007/s10623-019-00608-x

[20] Daniel Meirovitch and Lixia Zhang. 2021. *NSC – Named Service Calls, or a Remote Procedure Call for NDN*. Technical Report NDN-0074, Revision 1. NDN.

[21] Ilya Moiseenko. 2014. *Fetching content in Named Data Networking with embedded manifests*. Technical Report NDN-0025, Revision 1. NDN.

[22] Philipp Moll, Varun Patil, Nishant Sabharwal, and Lixia Zhang. 2021. A Brief Introduction to State Vector Sync. *NDN, Technical Report NDN-0073, Revision 2* (2021).

[23] Philipp Moll, Wentao Shang, Yingdi Yu, Lijing Wang, Alexander Afanasyev, and Lixia Zhang. 2021. A survey of distributed dataset synchronization in Named Data Networking. *NDN, Technical Report NDN-0053, Revision 2* (2021).

[24] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf

[25] Kathleen Nichols. 2019. Lessons learned building a secure network measurement framework using basic NDN. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*. 112–122.

[26] The Ethereum Project. 2021. Ethereum 2.0 Specifications. https://github.com/ethereum/eth2.0-specs.

[27] Yumi Sakemi, Tetsutaro Kobayashi, Tsunekazu Saito, and Riad Wahby. 2020. *Pairing-Friendly Curves*. Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-09. IETF Secretariat. http://www.ietf.org/internet-drafts/draft-irtf-cfrg-pairing-friendly-curves-09.txt http://www.ietf.org/internet-drafts/draft-irtf-cfrg-pairing-friendly-curves-09.txt.

[28] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11 (Nov. 1979), 612–613. https://doi.org/10.1145/359168.359176

[29] Mitsunari Shigeo. 2021. BLS threshold signature. https://github.com/herumi/bls.

[30] Victor Shoup. 2000. Practical threshold signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 207–220.

[31] The NDN Project Team. 2021. ndn-cxx: NDN C++ library with eXperimental eXtensions. https://github.com/named-data/ndn-cxx.

[32] The NDN Project Team. 2021. NDN Packet Format Specification version 0.3: Signature. https://named-data.net/doc/NDN-packet-spec/0.3/signature.html Accessed: 2021-04-20.

[33] The NDN Project Team. 2021. NDN Packet Format Specification version 0.3: Signed Interest. https://named-data.net/doc/NDN-packet-spec/0.3/signed-interest.html Accessed: 2021-04-20.

[34] The NDN Project Team. 2021. Validator Configuration File Format. https://named-data.net/doc/ndn-cxx/current/tutorials/security-validator-config.html Accessed: 2021-04-20.

[35] The Smart Grid Interoperability Panel – Smart Grid Cybersecurity Committee. 2014. *NISTIR 7628 Rev. 1: Guidelines for Smart Grid Cybersecurity*. Technical Report. NIST.

[36] Alin Tomescu, Robert Chen, Yiming Zheng, Ittai Abraham, Benny Pinkas, Guy Golan Gueta, and Srinivas Devadas. 2020. Towards Scalable Threshold Cryptosystems. In *2020 IEEE Symposium on Security and Privacy (SP)*. 877–893. https://doi.org/10.1109/SP40000.2020.00059

[37] Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. 2015. Schematizing Trust in Named Data Networking. In *Proc. of ACM ICN*.

[38] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, Patrick Crowley, Christos Papadopoulos, Lan Wang, Beichuan Zhang, et al. 2014. Named data networking. *ACM SIGCOMM Comp. Comm. Review* (2014).

[39] Zhiyi Zhang, Alexander Afanasyev, and Lixia Zhang. 2017. NDNCERT: Universal Usable Trust Management for NDN. In *Proceedings of the 4th ACM Conference on Information-Centric Networking* (Berlin, Germany) *(ICN '17)*. ACM, New York, NY, USA, 178–179. https://doi.org/10.1145/3125719.3132090

[40] Zhiyi Zhang and Siqi Liu. 2021. NDN-MPS Source Code Repository on GitHub. https://github.com/UCLA-IRL/ndn-multi-party-signature. Accessed: 2020-06-01.

[41] Zhiyi Zhang, Vishrant Vasavada, Siva Kesava Reddy K., Eric Osterweil, and Lixia Zhang. 2019. Expect More from the Networking: DDoS Mitigation by FITT in Named Data Networking. *CoRR* abs/1902.09033 (2019). arXiv:1902.09033 http://arxiv.org/abs/1902.09033

[42] Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry, Spyridon Mastorakis, Yanbiao Li, Alexander Afanasyev, and Lixia Zhang. 2018. An Overview of Security Support in Named Data Networking. *IEEE Communications Magazine* 56, 11 (November 2018), 62–68. https://doi.org/10.1109/MCOM.2018.1701147

[43] Meiyuan Zhao, Sean W. Smith, and David M. Nicol. 2005. Aggregated Path Authentication for Efficient BGP Security. In *Proceedings of the 12th ACM Conference on Computer and Communications Security* (Alexandria, VA, USA) *(CCS '05)*. Association for Computing Machinery, New York, NY, USA, 128–138. https://doi.org/10.1145/1102120.1102139