

## Article

# NDNOTA: NDN One-Time Authentication

Manar Aldaoud <sup>1,\*</sup> , Dawood Al-Abri <sup>1</sup>, Firdous Kausar <sup>1,2</sup> and Medhat Awadalla <sup>1,3</sup>

<sup>1</sup> Department of Electrical & Computer Engineering, Sultan Qaboos University, Al-Khouth, Muscat P.O. Box 33, Oman; alabrid@squ.edu.om (D.A.-A.); firdous@squ.edu.om (F.K.); medhatha@squ.edu.om (M.A.)

<sup>2</sup> Computer Science Department, Fisk University, Nashville, TN 37208, USA

<sup>3</sup> Computer and System Department, Helwan University, Cairo 12612, Egypt

\* Correspondence: s117905@student.squ.edu.om; Tel.: +968-99565201

**Abstract:** Named Data Networking (NDN) stands out as a prominent architectural framework for the future Internet, aiming to address deficiencies present in IP networks, specifically in the domain of security. Although NDN packets containing requested content are signed with the publisher's signature which establishes data provenance for content, the NDN domain still requires more holistic frameworks that address consumers' identity verification while accessing protected contents or services using producer/publisher-preapproved authentication servers. In response, this paper introduces the NDN One-Time Authentication (NDNOTA) framework, designed to authenticate NDN online services, applications, and data in real time. NDNOTA comprises three fundamental elements: the consumer, producer, and authentication server. Employing a variety of security measures such as single sign-on (SSO), token credentials, certified asymmetric keys, and signed NDN packets, NDNOTA aims to reinforce the security of NDN-based interactions. To assess the effectiveness of the proposed framework, we validate and evaluate its impact on the three core elements in terms of time performance. For example, when accessing authenticated content through the entire NDNOTA process, consumers experience an additional time overhead of 70 milliseconds, making the total process take 83 milliseconds. In contrast, accessing normal content that does not require authentication does not incur this delay. The additional NDNOTA delay is mitigated once the authentication token is generated and stored, resulting in a comparable time frame to unauthenticated content requests. Additionally, obtaining private content through the authentication process requires 10 messages, whereas acquiring public data only requires two messages.

**Keywords:** authentication server; cryptography; Named Data Networking; one-time authentication; single sign-on; tokens



**Citation:** Aldaoud, M.; Al-Abri, D.; Kausar, F.; Awadalla, M. NDNOTA: NDN One-Time Authentication. *Information* **2024**, *15*, 289. <https://doi.org/10.3390/info15050289>

Academic Editor:  
Abderrezak Rachedi

Received: 31 March 2024  
Revised: 20 April 2024  
Accepted: 22 April 2024  
Published: 20 May 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The Named Data Networking (NDN) paradigm was proposed to overcome many challenges in IP networks. One of these challenges is security. Unlike the traditional IP networks where traffic is secured by creating secure channels like Transport Layer Security (TLS), in NDN networks, the content itself is cryptographically secured and digitally signed and this in turn enables asynchronous production and consumption of data [1]. In NDN, Interest packets are used to request content and Data packets are used to deliver content. Data packets are required to be signed and therefore the publisher's identity can be verified. On the other hand, Interest packets are optionally signed, therefore, the consumer's identity cannot always be verified [2]. Many contents and services require user authentication before granting access to the user, such as emails, web accounts, cloud resources, etc. Due to these issues, the need for NDN authentication appears.

In today's Internet world, security is one of the most challenging issues. Using Internet applications and contents is increasing rapidly, and the transmitted data must be secured to avoid reading, copying, or altering it by unauthenticated parties, e.g., attackers.

Cryptography is a technique used to secure data and maintain data confidentiality, by converting data from readable data to unintelligible data for unauthenticated users. A cryptographic algorithm is mainly classified into three categories: (1) hash cryptography, which is a mathematical function that maps data of any size—called a message—into a fixed length of array bits—called a message digest, (2) symmetric key cryptography that uses the same key to encrypt and decrypt data, and (3) asymmetric key cryptography, which uses the public key to encrypt the data and the private key to decrypt the encrypted message. The appropriate type is chosen according to the application and the data sensitivity [3].

Security includes two main terms: authentication and authorization, where authentication is the process or action of verifying the identity of a user or process and authorization involves verifying the specific applications, files, and data that a user can access [4]. Nowadays, the authentication process can be described in two phases: identification and genuine authentication, where the user identity is checked by the provided evidence called “credential”. There are three credential categories to authenticate users: (1) “something the user knows” such as passwords, (2) “something the user is” such as biometric characteristics, and (3) “something the user has” such as tokens and keys. These credentials can be combined to increase security [5]. Single sign-on (SSO) is one of the authentication schemes that allows users to authenticate against a service by receiving a provisional token from an authentication server (AuthServ) based on their credentials [6]. SSO is based upon a trust relationship between the service provider and the identity provider. The trust relationship in SSO is based on identity data that takes the form of tokens. SSO enables an end user’s account information to be used by third-party services without exposing the user’s password [7,8].

The main contribution of this paper is to propose an operational framework that comprises the messaging protocol, mechanism, and workflow to allow NDN consumers to authenticate to NDN publishers via NDN authentication servers in native clean-slate NDN networks. This framework allows an NDN consumer to access private NDN content, applications, or services using a login credential. NDNOTA accomplishes this goal through three primary mechanisms: (1) leveraging single sign-on (SSO), (2) employing token credentials and certified asymmetric keys, and (3) making use of cryptographically-signed Interest and Data packets. The authentication process in NDNOTA involves four distinct phases and relies on the utilization of a proposed authentication message (AuthMsg), which is integrated into either the Interest Application Parameters element or the Data Content element. The motivation behind NDNOTA is to address the shortcomings in Peer Name Provider Servers (PNPSs) [9,10] by providing a secure authentication layer, a single sign-on mechanism, and a communication protocol to solve PNPS scalability issues and to eliminate the need to manually manage the PNPS records, i.e., the mapping of the publishers’ name to autonomous system NDN names. Later in Section 3.2, we will detail how an authorized consumer, “Bob”, will authenticate to one of the PNPSs’ preapproved authentication servers, “AuthServ”, to manage the PNPS records of the publisher, “Alice”. The subsequent sections discuss the framework design overview and particulars. For the sake of understanding, the abbreviations frequently used in this paper are found at the end of the paper.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 covers the proposed authentication framework named “NDNOTA”. Section 4 discusses the security analysis of our proposed solution, examining its resilience against various attacks. Section 5 describes the lab setup and presents the performance evaluation of the proposed framework. Finally, Section 6 concludes the paper and presents the future work.

## 2. Related Work

Several studies survey the prominent security breaches within NDN and various detection methods, as indicated by [11,12]. Nonetheless, our research introduces an authentication framework aimed at verifying user identity before accessing private data or

services. We have conducted theoretical analyses demonstrating the robustness of this proposed framework against three integrity attacks mentioned in [13]. Authors in [14–16] highlighted specific trust management and security aspects in NDN, notably the “producer signature”. Their argument supports the notion that signing NDN packets improves data integrity and authenticity by requiring content producers to cryptographically sign their data before transmission. Nonetheless, this approach brings about supplementary challenges, encompassing verification overhead, certificate retrieval, Data packet size, and producer privacy concerns. However, when weighing these challenges against the benefits of the producer’s signature, they may be deemed negligible. Moreover, Benmoussa et al. [17] survey proposed solutions aimed at addressing Interest flooding attacks, with some of these solutions utilizing signed Interest packets to mitigate such attacks.

Additionally, many solutions have been proposed in the NDN authentication area. Most of these works have focused on access control lists for NDN. Nour et al. [18] conducted an exhaustive and thorough survey of the access control mechanisms in NDN. Nonetheless, our attention will be directed towards the work most closely aligned with our own.

Chen et al. [19] proposed a probability-based access control mechanism for NDN that used both symmetric and asymmetric encryption. First, the consumers and providers exchange symmetric keys. Then, a consumer sends an Interest packet signed with its private key. After that, the provider replies with a Data packet that includes the requested content encrypted with the symmetric key. Their proposed mechanism is improved with a Bloom filter that contains authorized clients’ public keys. This Bloom filter database is used by the intermediate routers to verify the consumer authority before forwarding encrypted content. Although their access control mechanism is resilient and distributed across the network, it increases the computational burden on the intermediate routers. Also, as the number of clients rises, the size of the filters in intermediate routers increases.

Hamdane et al. [20] proposed an access control solution to encrypt contents. The proposed solution is based on a symmetric key that is associated with the content name, which is valid whether the authorized entities are known or not. The authors claimed that their proposed solution has an acceptable overhead. The solution uses a lazy revocation mechanism, which is complicated and undesirable especially when multiple files are encrypted with the same key [21].

Shang et al. [22] proposed “NDN-ACE”, an access control framework for actuators in an NDN-IoT environment where the actuator generates a symmetric key and shares it with the authorized server. After that, the authorized server derives access keys per consumer per service based on the shared key. When the consumer sends a command Interest packet, the actuator verifies the command based on the appended information: (1) a signature signed by an access key that shows that the consumer is authorized by the authorized server, and (2) signature information that the actuator uses to recompute the access key. The proposed framework shows better results than other solutions in terms of overhead because it does not need to maintain secure channels.

Misra et al. [23] proposed a broadcast-based access control framework that uses Shamir’s secret sharing and is especially appropriate for mobile devices. The content provider in the proposed framework has to: (1) encrypt each content with a symmetric key, and (2) generate and distribute small metadata that contain the symmetric keys. The consumer first sends an Interest packet with some credentials requesting content and its metadata. This Interest packet is signed with the provider’s public key. After that, the provider replies with a unicast Data packet that is not cached in any intermediate router, which is signed with the consumer’s public key. Lastly, the consumer utilizes the metadata and his private key to decrypt the requested content. The authors argue that the proposed framework has an effective user revocation that does not affect the system performance and requires metadata creation and network-wide dissemination. However, the proposed framework incurs computation overhead every time a consumer requests content, which increases when the number of consumers increases.

The authors in [24] combined both an identity-based proxy re-encryption scheme (IB-PRE) and symmetric encryption to enhance access control for content sharing in NDN, aiming to protect content confidentiality. However, these access control schemes suffer from key escrow problems.

The most related work to the proposed NDNOTA is accomplished in [25]. Li et al. [25] suggested a lightweight digital signature mechanism called “LIVE”. The content provider initially classifies authorized routers from unauthorized ones. Then, it generates three tokens per content: (1) two private tokens for authorized routers and users, and (2) one public token for unauthorized routers. These tokens are produced and allocated by the content provider. However, LIVE suffers from the storage overhead that occurs because of three tokens per content. Moreover, it requires the content provider to know the identity information of authorized routers alongside their public keys, which is against NDN architecture that is based on anonymous consumers.

Additionally, a new security aspect that appears to ICN is blockchain (BICN). Blockchain constitutes a decentralized ledger featuring expanding arrays of blocks interconnected securely through cryptographic hashes. Each block encompasses a cryptographic hash of its predecessor, alongside a timestamp and transactional details. Given that each block references the preceding one, they establish a sequential chain, wherein each new block extends the connection to the preceding blocks. Thus, alterations to data within any block post-recording necessitate modifications across all ensuing blocks, rendering blockchain transactions irreversible. In blockchain, tokens act as the adhesive binding a decentralized blockchain network [26]. BICN primarily safeguards the network against denial-of-service (DoS), hijacking, and cache pollution vulnerabilities. Nevertheless, privacy remains a concern as anonymizing transaction addresses does not assure user anonymity [26–29]. However, in our proposed NDNOTA, we aim to authenticate the identity of the requester intending to access private data. In [27], the authors attempted to employ blockchain authentication to circumvent the necessity of communication with a central authority, given its impracticality within vehicular networks. However, in our proposed NDNOTA, we seek to authenticate the identity of the requester seeking access to private data through multiple centralized authorities (authentication servers). Furthermore, one of the mechanisms we attempted to integrate was single sign-on (SSO) within the NDN network, using the proposed framework NDNOTA as in the TCP/IP protocol. Table 1 includes a summary of some other works.

**Table 1.** Summary of other works.

| Approach                                                                                                               | Objective                                                                                                                                                                                                                                   | Limitation                                                                                         |
|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| Encrypting data using probabilistic methods [19]                                                                       | Prevent unauthorized access                                                                                                                                                                                                                 | Employing Bloom filters results in higher memory usage and a greater occurrence of false positives |
| Support fine-grained access control over the objects [20]                                                              | Objects’ confidentiality                                                                                                                                                                                                                    | Unable to conceal the popularity of objects and users’ preferences from potential attackers.       |
| Secure actuators in an NDN-IoT environment [22] by using generated symmetric keys                                      | Prevent data leakage                                                                                                                                                                                                                        | Addresses the sending of commands, as demonstrated in the case of lighting control                 |
| Implementing broadcast encryption for content stored in content stores [23] by using symmetric keys to encrypt content | Data confidentiality especially for mobile objects                                                                                                                                                                                          | Does not take into consideration dynamic changes that occur in content stores                      |
| Lightweight integrity verification [25] by using three private and public tokens                                       | Distinguish between authorized and unauthorized routers                                                                                                                                                                                     | Does not take into consideration token refreshment                                                 |
| Using (IB-PRE) and symmetric encryption [24]                                                                           | Improve access control for content sharing in NDN, aiming to safeguard content confidentiality                                                                                                                                              | Key escrow problems                                                                                |
| NDNOTA is an authentication framework for NDN services, applications, and content                                      | 1. One authentication for multiple services such as SSO<br>2. Uses asymmetric cryptography to generate tokens<br>3. Uses the original NDN signature for both Interest and Data packets. Accordingly, no modifications occur to NDN packets. | Token revocation must be considered in future work                                                 |

### 3. The Proposed Authentication Framework

The suggested approach in this paper comprises three primary elements: consumer, producer, and authentication server (AuthServ); all elements were developed as NDN applications and communicate using NDN native channels, i.e., not encapsulated in Internet Protocol “IP” packets. The AuthServ serves as an identity provider (IdP) for consumers, providing them with authentication tokens. The producer functions as a service provider, integrating its identity management with the AuthServ to enable consumers to access their respective private data using the tokens issued to them. AuthServ is specifically crafted to authenticate consumers who submit a signed Interest packet to a producer, aiming to access private NDN content or service. Each entity in the network—consumer, producer, and AuthServ—possesses a pair of public and private keys, with their public keys certified by a reputable certification authority.

Additionally, the NDN trust model is adopted, which relies on signing both Interest and Data packets. This signature serves to guarantee integrity and facilitates the identification of data provenance, allowing the recipient to trust the data independently of how or where it was acquired. In NDNOTA, all Interest and Data packets are signed with an NDN signature known as SignatureSha256WithEcdsa [30], which specifies an ECDSA public key signature computed over the SHA-256 hash of the “signed portion” of an Interest or Data packet. This signature type, upon verification, provides strong guarantees concerning both the authenticity of the packet’s source (authentication/provenance) and its integrity while in transit, ensuring that the packet remains unchanged along the way.

This section presents the proposed authentication message and the four main phases of the proposed authentication framework NDN One-Time Authentication (NDNOTA).

#### 3.1. Authentication Message

The suggested authentication message (AuthMsg) adheres to a type-length-value (TLV) format. This message can serve as an Application Parameters element in the Interest packet or as a Content element in the Data packet. This authentication message has the following fields, as shown in Figure 1:

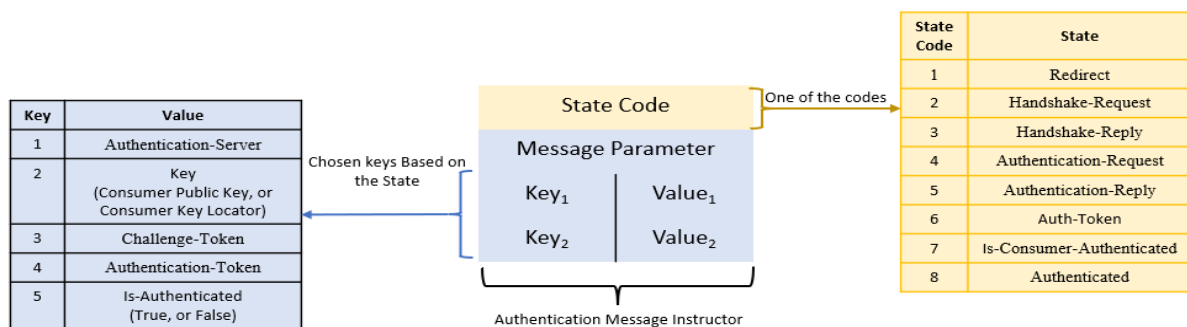


Figure 1. Authentication message.

- **State Field:** contains code, where each code represents a state: (1) Redirect state, (2) Handshake-Request state, (3) Handshake-Reply state, (4) Authentication-Request state, (5) Authentication-Reply state, (6) Auth-Token state, (7) Is-Consumer-Authenticated state, and (8) Authenticated state;
- **Message Parameter Field:** contains one or more (key, value) pairs. The key field is filled with one code from 1–5. Each key code represents a value that is described in Table 2.

**Table 2.** Authentication message parameters.

| Key Code | Value                                                                                                                                                                         |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1        | One or more AuthServ names                                                                                                                                                    |
| 2        | Either the consumer public key when state field is 2, or consumer key locator, which is used as a consumer unique identifier (consumer ID), when the state field holds code 7 |
| 3        | Challenge token (CT)                                                                                                                                                          |
| 4        | Authentication token (AT)                                                                                                                                                     |
| 5        | The value is TRUE when the consumer is authenticated, and FALSE otherwise                                                                                                     |

### 3.2. NDNOTA Authentication Phases

Consider “Bob” as the consumer and “Alice” as the producer, who acts as a service provider. Bob, Alice, and AuthServ will go through the following steps to ensure that Bob is authenticated to access the NDN content, service, or application provided by Alice.

When Bob wants content, he will send an Interest packet to Alice requesting the data. If the content is publicly accessible, meaning it does not require authentication, Alice will reply with a Data packet containing the requested information in its Content element.

If the content is private and requires authentication, Alice will send a NACK Data packet, shown in Figure 2, that holds AuthMsg in its Content element, which includes the following information: (1) state code one, and (2) a pair consisting of (one, AuthServs), where the AuthServs value includes a list of one or more suggested authentication servers for requesting authentication. This is similar to requesting authentication from various service providers in the IP-based single sign-on (SSO) process, such as requesting sign-in from platforms like Google, YouTube, Facebook, and others. In the provided example, the designated AuthServ is represented as “/example/authserv”.

```
Named Data Networking (NDN), Data, Name: /om/edu/squ/www/protected/params-sha256=b4a5be5a987ac5b8a8d6645105c0e8a43f5ef4ae
  Data, Type: 6, Length: 247, Name: /om/edu/squ/www/protected/params-sha256=b4a5be5a987ac5b8a8d6645105c0e8a43f5ef4ae66c5
    Name: /om/edu/squ/www/protected/params-sha256=b4a5be5a987ac5b8a8d6645105c0e8a43f5ef4ae66c512bdf41d3a0c8b96feab
    MetaInfo, Type: 20, Length: 3, ContentType: (3) NACK
      Content: 152ffd01710817955a99932fe873fd01720101fd01731afd01740101fd0175112f657861..., Type: 21, Length: 47
        NDNOTA Message Type: (1) REDIRECT
          NDNOTA Parameter, Type: 371, Length: 26, Para-Key: (1) AUTHENTICATION_SERVER, Para-Value: /example/authserv
            Key: (1) AUTHENTICATION_SERVER
            Value: /example/authserv
          SignatureInfo, Type: 22, Length: 52, SignatureType: (3) SignatureSha256WithEcdsa, KeyLocator: Name: /om/edu/squ/KEY
            SignatureType: (3) SignatureSha256WithEcdsa
              KeyLocator, Type: 28, Length: 47, Name: /om/edu/squ/KEY/%F9%81%27%27R%15%D0%8C/self/54=%00%00%01%8B%A4%C1%B6%8B
```

**Figure 2.** Redirect NACK Data packet: capture.

When Bob receives this NACK Data packet, it will go through the NDNOTA authentication phases, unless he already has an AT from one of the suggested AuthServs. NDNOTA authentication goes through four phases that can be summarized as follows: (1) handshake authentication request phase, (2) challenge token exchange phase, (3) authentication token exchange phase, and (4) authentication token verification phase. These four phases cover consumer registration, authentication, and verification. Henceforward, it will be mandatory for both interest creators, namely consumers and producers, to sign all Interest packets.

#### 3.2.1. Handshake Authentication Request Phase

The main goal of this phase is to initiate a handshake session between the consumer and the AuthServ to obtain an AT. When Bob receives a Redirect message from Alice, he will send a “Handshake Authentication Request” (HAR) Interest packet to one of the suggested AuthServs, e.g., “/example/authserv”. This packet is signed with Bob’s signature. The Application Parameters contain the AuthMsg that includes the following information:

(1) state code two, and (2) a pair consisting of (two, KEY value), where the KEY value includes Bob's public key, as shown in Figure 3.

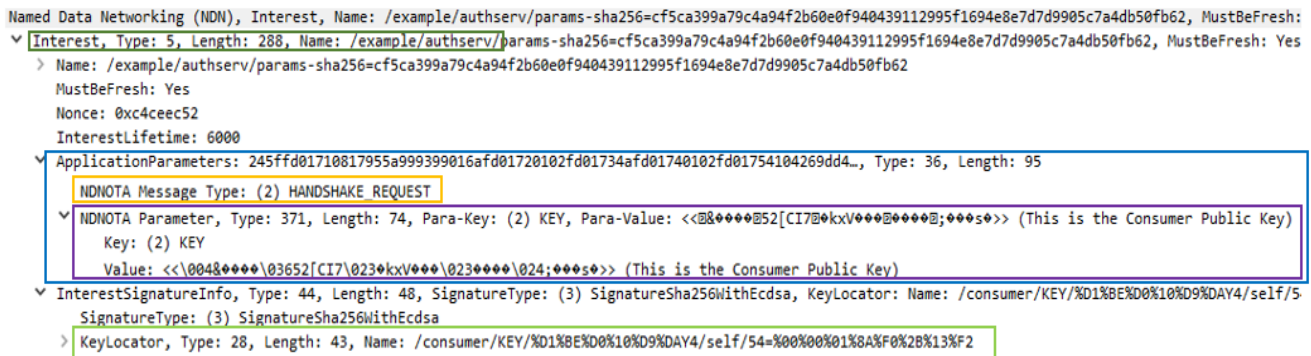


Figure 3. Handshake authentication request Interest packet: capture.

### 3.2.2. Challenge Token Exchange Phase

The main goal of this phase is to facilitate the exchange of a challenge token (CT) between the AuthServ and the consumer, thereby establishing trust in the consumer. Upon receiving the HAR Interest packet, the AuthServ extracts Bob's key locator from its signature and his public key from the AuthMsg Parameter. The key locator is used to create a session entry for Bob with the AuthServ. Afterward, it assembles an AuthMsg that includes the following information: (1) state code three, (2) a pair consisting of (three, CT value), where the CT value is a randomly generated value and encrypted using Bob's public key, and (3) a pair consisting of (two, KEY value), where the KEY value includes the AuthServ public key.

Following this, the AuthServ establishes a session entry for Bob, containing: (1) Bob's key locator, (2) CT value, and (3) the expiration time of the CT. Finally, it dispatches to Bob a Data packet named "Handshake Reply" signed with the AuthServ's signature, and its Content element encapsulates the generated AuthMsg.

Once Bob receives the Handshake Reply packet, he extracts both the encrypted CT and the public key of the AuthServ. Subsequently, Bob decrypts the received CT into plaintext-CT using his private key. Following this, Bob generates a signed Interest packet named "Authentication Request" (AR) to (1) send back the CT, and (2) request an authentication token (AT) from the AuthServ. Within the AR's Application Parameters element, there is an AuthMsg that includes the following information: (1) state code four, and (2) a pair consisting of (three, CT value), where the CT value is the plaintext-CT encrypted with the AuthServ public key. At last, the AR is dispatched to the AuthServ. This concludes this phase, signifying the successful exchange of the CT between the consumer and the AuthServ.

### 3.2.3. Authentication Token Generation Phase

The main goal of this phase is to instantiate an authentication token for Bob, enabling him to use it when requesting a private content. Upon receiving the AR Interest packet, the AuthServ extracts the encrypted CT and Bob's key locator from the Application Parameters. Subsequently, it uses its private key to decrypt the received CT and verifies it against the initially generated CT. Accordingly, both the AuthServ and consumer identities are validated and confirmed from both sides.

In case of no match, the AuthServ dispatches a NACK Data packet, otherwise, AuthServ dynamically generates another random token called "Authentication Token" (AT) which will be encrypted using the AuthServ key. After that, it assembles an AuthMsg comprising the following information: (1) state code five, and (2) a pair consisting of (four, encrypted AT value). Following this, the AuthServ updates Bob's session entry, as shown in Figure 4:

- Replace the CT value with the AT value;
- Replace the expiration time of the CT with the expiration time of the AT.

|                 |      |             |
|-----------------|------|-------------|
| Bob key-locator | ⊕ AT | ⊕ AT Expiry |
|-----------------|------|-------------|

**Figure 4.** Session entry.

Finally, it dispatches a Data packet named “Authentication Reply” to Bob with the generated AuthMsg encapsulated in its Content element.

Upon receiving the Authentication Reply Data packet, Bob extracts and stores his authentication token (AT) alongside the AuthServ name for future utilization when accessing secured online data, applications, and services. Accordingly, in our proposed framework, each consumer has one saved authentication token (AT) in each AuthServ database that can be used to access private content. On the contrary, LIVE has a storage overhead problem since it saves three tokens per content.

### 3.2.4. Consumer Verification Phase

The main goal of this phase is for the producer to verify the consumer’s authentication token (AT) by querying the AuthServ, to enable the consumer to access the requested private content. Now, when Bob wants private data from Alice, he will assemble an AuthMsg comprising the following information: (1) state code six, (2) a pair consisting of (one, AuthServ), where the AuthServ value includes the name of the AuthServ that issued the AT and has it in its database, e.g., “/example/authserv”, and (3) a pair consisting of (four, AT value), where the AT value is what has been received from the AuthServ and stored. Then, he sends Alice a signed Interest packet named “Private Content Request” whose Application Parameters element encapsulates the generated AuthMsg.

When Alice receives Bob’s Private Content Request Interest packet, she extracts the AT and Bob’s key locator from the AuthMsg in the Application Parameters element. Then, she verifies whether this key locator is cached and stored in her database. Upon discovery, she assesses the authentication token (AT) to confirm a match, and if successful, a Data packet that holds the private content is sent to Bob.

On the other hand, if the key locator is not cached, Alice constructs an AuthMsg that contains the following: (1) state code seven, (2) a pair consisting of (two, KEY value), where the KEY value includes Bob’s key locator, and a pair consisting of (four, AT value). Then, she sends an Interest packet named “Is-Consumer-Authenticated”, that is signed with her signature and includes the generated AuthMsg in its Application Parameters element, to the AuthServ in the AuthMsg, e.g., “/example/authserv”.

Once the AuthServ receives Alice’s Interest packet, it obtains the AT and the consumer’s key locator. Then, it compares the received AT with the one saved in the consumer’s session entry. After that, the AuthServ sends a signed Data packet to Alice that contains an AuthMsg in its Content element which includes the following information: (1) state code eight, and (2) a pair consisting of (five, Is-Consumer-Authenticated value), where the value will be either ‘True’ if Bob is authenticated or ‘False’ if he is not authenticated to access the private data.

Depending on the Is-Consumer-Authenticated value in the received message, Alice will take the corresponding action. If Alice receives a ‘False’ value, a NACK Data packet is sent to Bob. On the other hand, if Alice receives a ‘True’ value, she will dispatch a Data packet containing the private content to Bob. As can be noticed, when the requested content is public, the producer is unaware of the consumer’s identity. However, in the case of private content, the consumer is required to send a request in the form of a signed Interest packet, which discloses their identity and key locator, similar to LIVE [25].

A multi-step flow diagram in Figure 5 represents the complex interaction between Bob, Alice, and the AuthServ in the NDNOTA authentication process.

The algorithm of the NDNOTA authentication process in the three main elements, consumer, producer, and authentication server is as follows: Algorithm 1 shows the proposed algorithm from the consumer side, Algorithm 2 shows the proposed algorithm from the producer side, and Algorithm 3 shows the proposed algorithm from the authentication

server side.

---

**Algorithm 1** NDNOTA Consumer Side

---

```

1: Output 1: Handshake Authentication Request Interest (HAR) to AuthServ
2: Output 2: Authentication Request Interest (AR) to AuthServ
3: Output 3: Protected-Content Request Interest to Producer


---


4: Content Request Interest to Producer
5: Data Packet  $\leftarrow$  Producer
6: if Data Packet type == 0 (Blob) then
7:   do nothing
8: else if Data Packet type == 3 (NACK) && AuthMsg.Type = 1 then
9:   Extract AuthServer list
10:  if consumer has no AT of one of the AuthServ list then
11:    AuthMsg  $\leftarrow$  2 (Handshake-Request State), (2, Consumer public key)
12:    ApplicationParameters  $\leftarrow$  AuthMsg
13:    HAR Interest  $\leftarrow$  (AuthServer Name + ApplicationParameters) & Sign Interest with
    Consumer Signature
14:    Issue HAR to AuthServ  $\rightarrow$  Output 1
15:  else
16:    go to 26
17:  end if
18: end if
19: Data Packet  $\leftarrow$  AuthServ
20: switch Received Data Packet do
21:  case Handshake-Reply (3) State
22:    Extract CT, AuthServ public key
23:    D-CT  $\leftarrow$  Decrypt CT with Consumer private key
24:    CT  $\leftarrow$  Encrypt D-CT with AuthServ public key
25:    AuthMsg  $\leftarrow$  4 (Authentication-Request State), (3, CT)
26:    ApplicationParameters  $\leftarrow$  AuthMsg
27:    AR Interest  $\leftarrow$  (AuthServer Name + ApplicationParameters) & Sign with Consumer
    Signature
28:    Issue AR to AuthServ  $\rightarrow$  Output 2
29:  case Authentication-Reply (5) State
30:    Extract & save AT
31:    AuthMsg  $\leftarrow$  6 (AuthToken State), (1, AuthServ), (4, AT)
32:    ApplicationParameters  $\leftarrow$  AuthMsg
33:    Content Request Interest  $\leftarrow$  (Producer Name + ApplicationParameters) & Sign with Consumer
    Signature
34:    Issue Content Request Interest to Producer  $\rightarrow$  Output 3

```

---

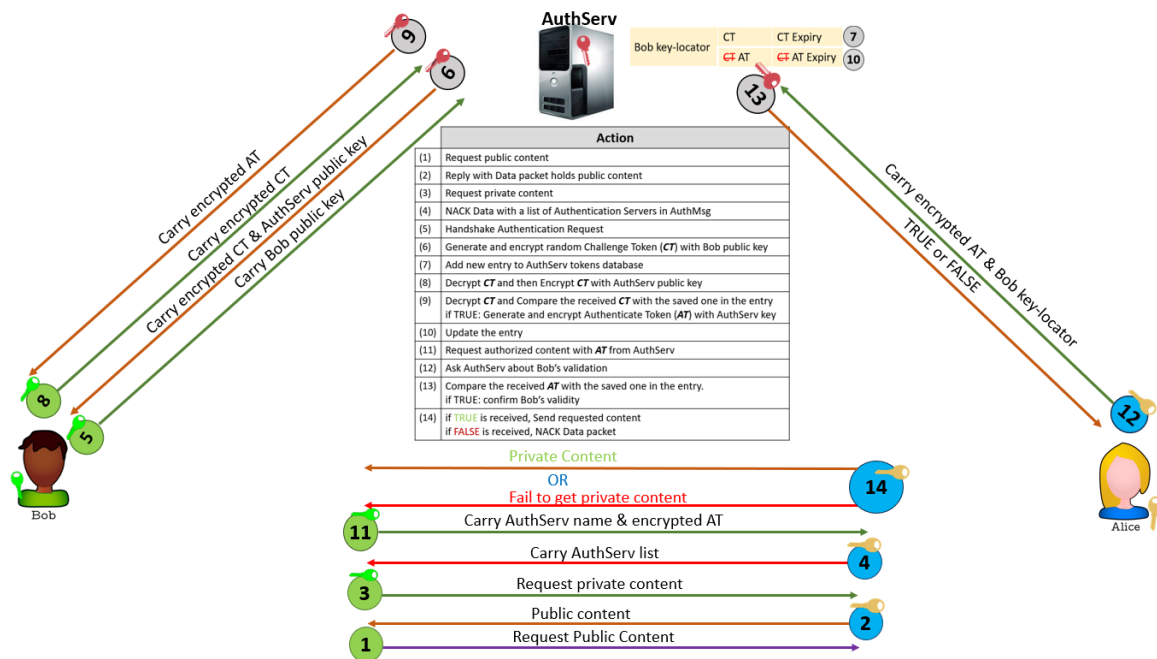
**Algorithm 2** NDNOTA Producer Side

```

1: Input 1: Content Request Interest from Consumer
2: Output 1: Is-Consumer-Authenticated Interest to AuthServ
3: Output 2: Data Packet to Consumer

4: if ApplicationParameters = Empty && requested content != protected then
5:   Issue Data packet that carries requested content
6: else
7:   if ApplicationParameters = Empty && requested content = protected then
8:     AuthMsg  $\leftarrow$  1 (Redirect State), (1, list of AuthServs)
9:     Content  $\leftarrow$  AuthMsg
10:    NACK Data Packet  $\leftarrow$  (Content) & Sign with Producer Signature
11:  else if AuthMsg.Type = 6 && AuthMsg.Parameter[1].key = 4 then
12:    Extract AT, consumer key-locator
13:    if consumer key-locator is cached then
14:      if received AT = cached AT && NOT expired then
15:        go to 27
16:      else
17:        AuthMsg  $\leftarrow$  7 (Is-Consumer-Authenticated State), (2, consumer key-locator), (4, AT)
18:        ApplicationParameters  $\leftarrow$  AuthMsg
19:        Is-Consumer-Authenticated Interest  $\leftarrow$  (AuthServer Name + ApplicationParameters) &
        Sign with Producer Signature
20:        Issue Is-Consumer-Authenticated Interest to AuthServ  $\rightarrow$  Output 1
21:      end if
22:    end if
23:  end if
24:  Data Packet  $\leftarrow$  AuthServ
25:  if AuthMsg.Type = 8 && AuthMsg.Parameter[0].value = True then
26:    Store consumer key-locator & AT
27:    Data Packet  $\leftarrow$  (Protected Content) & Sign with Producer Signature
28:    Issue Data Packet with protected content to Consumer  $\rightarrow$  Output 2
29:  else
30:    Issue NACK Data Packet to Consumer  $\rightarrow$  Output 2
31:  end if

```

**Figure 5.** NDNOTA authentication process.

**Algorithm 3** NDNOTA AuthServ Side

---

```

1: Input 1: Handshake Authentication Request Interest (HAR) from Consumer
2: Output 1: Handshake Reply (HR) Data Packet to Consumer
3: Output 2: Authentication Reply (AR) Data Packet to Consumer
4: Output 3: Authenticate Verification Data Packet to Producer

```

---

```

5: if AuthMsg.Type = 2 then
6:   Extract consumer public key, key-locator
7:   CT ← Generate & encrypt a random token with consumer public key
8:   entry ← Consumer key-locator, Plaintext CT, CT Expiry
9:   AuthMsg ← 3 (Handshake-Reply State), (3, CT)
10:  Content ← AuthMsg
11:  HR Data Packet ← (Content) & Sign with AuthServ Signature
12:  Issue HR Data Packet to Consumer → Output 1
13: end if
14: switch Received Interest Packet do
15:   case Authentication-Request (4) State
16:    Extract CT, consumer key-locator
17:    D-CT ← Decrypt CT with AuthServ private key
18:    if D-CT = entry (CT) then
19:      AT ← Generate & encrypt a random token with AuthServ key
20:      entry ← Consumer key-locator, AT, AT Expiry
21:      AuthMsg ← 5 (Authentication-Reply State), (4, AT)
22:      Content ← AuthMsg
23:      AR Data Packet ← (Content) & Sign with AuthServ Signature
24:      Issue AR Data Packet to Consumer → Output 2
25:    else
26:      Issue NACK Data Packet to Consumer
27:    end if
28:   case Is-Consumer-Authenticated (7) State
29:    Extract AT, consumer key-locator
30:    if AT = entry (AT) then
31:      AuthMsg ← 8 (Authenticated State), (5, True)
32:      Authenticate Confirmation Data Packet ← (Content) & Sign with AuthServ Signature
33:      Issue Authenticate Confirmation Data Packet to Producer → Output 3
34:    else
35:      AuthMsg ← 8 (Authenticated State), (5, False)
36:    end if
37:    Content ← AuthMsg
38:    Authenticate Verification Data Packet ← (Content) & Sign with AuthServ Signature
39:    Issue Authenticate Confirmation Data Packet to Producer → Output 3

```

---

**4. Security Analysis**

In this section, we examined the resilience of our proposed solution, NDNOTA, against various attacks prevalent in NDN, including those affecting content integrity such as man-in-the-middle-attacks (MITM), impersonation, and replay attacks as delineated in a recent relevant survey [13]. For each attack category, we elucidated how NDNOTA, the proposed solution, counters these threats. It is worth mentioning that NDNOTA inherits all the security features of NDN since it is built vertically on top of NDN-CXX [31].

- **Man-in-the-middle (MITM) Attack:** The unauthorized interception can take place either on the in-path routers or by a malicious man-in-the-middle attacker. To protect the exchanged messages between the consumer, the producer, and the authentication server our scheme uses asymmetric encryption by using Elliptic Curve Cryptography (ECC) to encrypt the exchanged tokens with the destinations' ECC public key. Additionally, to protect exchanged messages from alteration, our scheme requires all messages (Interest/Data packets) to use ECDSA digital signatures over a SHA-256 digest to enforce integrity and provenance verification [30];

- **Impersonation Attack:** Instead of examining this attack through the lens of the consumer, the producer, and the AuthServ, impersonation could occur through the transmission of either an Interest or a Data packet from an unauthorized sender;
- **Interest Packets Impersonation:** In our solution, both the AuthServ and the producer define a consumer by either an attached public key or by the key location in the first Interest packet they receive from the user. A trusted key location will have higher priority over an attached public key to distinguish between a publicly known signature and another signature with the same NDN name. The public key—whether downloaded from the key location or the attached one—will be stored and will be used to validate any future Interest packets received from the same consumer. Since Interest packets are signed using a private key, an attacker cannot create a new Interest packet without having the private key. However, an attacker can passively store an Interest packet—for example, obtained through a MITM attack—and replay it as we will see in the replay attack;
- **Data Packets Impersonation:** Our scheme mandates setting MustBeFresh to true for Interest packets and setting FreshnessPeriod to zero for Data packets. This way the Interest packets cannot be satisfied from a content store and the Data packets are marked “non-fresh” after satisfying the Interest. This shall reduce the vulnerability to poisoned cached content. Moreover, the Data packet is verified at its final destination for the correctness and authenticity of its digital signature;
- **Replay Attack:** Since all Interest packets in our proposed solution are digitally signed, both the AuthServ and the producer will use the NDN built-in mechanism to verify and validate the signed Interest packets [32]. The mechanism mandates the use of a digest component and a published and verifiable signature. If the mechanism fails, the Interest packet will be treated as invalid. In-path routers can be optionally configured to verify signatures and drop invalid packets, and content providers will always drop invalid packets.

One more aspect to cover in our security analysis is privacy. Our proposed solution leverages the following techniques to achieve privacy and confidentiality of the exchanged NDN packets both in-transit and at-rest: (1) protecting packets’ payloads via ECC with AES [hybrid AES-ECC model for the security of data over cloud storage] and (2) disable caching of authentication packets using Interest’s MustBeFresh and Data’s FreshnessPeriod. Nonetheless, other studies like [33] have proposed a scheme aimed at securing dynamic content within NDN networks. Integrating our work with initiatives such as this could enhance content privacy for cached data.

## 5. Experimental Setup and Result

Multiple experiments were conducted to verify the validation and performance of NDNOTA in the communication link between the AuthServ and the resolver [10]. This is crucial as the content provider must be authenticated to make modifications and edits to the mapping table of the PNPS. In this scenario, Bob is the content provider, and Alice is the PNPS. Both are connected to an AuthServ, which was implemented as an NDN application. This application registers a route and a face in the NFD FIB table.

The experiments were measured on a workstation with a 4 cores Intel(R) Xeon(R) E5-1620 v2 CPU running at 3.7 GHz with 32 GB RAM that runs Ubuntu.22.04 [34]. The three tries were implemented in Python 3.10.

Two experiments were conducted. The first one was conducted to assess the influence of the proposed framework on the three elements—AuthServ, consumer, and producer—in terms of processing time and delay. The second one was conducted to evaluate the delays occurring from operations within NDNOTA, including token generation, encryption, and decryption processes.

### 5.1. Authentication Impact on NDNOTA Element

In this experiment, the aim was to examine how the authentication process conducted by the proposed NDNOTA framework influences its key elements, namely the AuthServ, consumer, and producer. This experiment was repeated 10 times, and the results were averaged out.

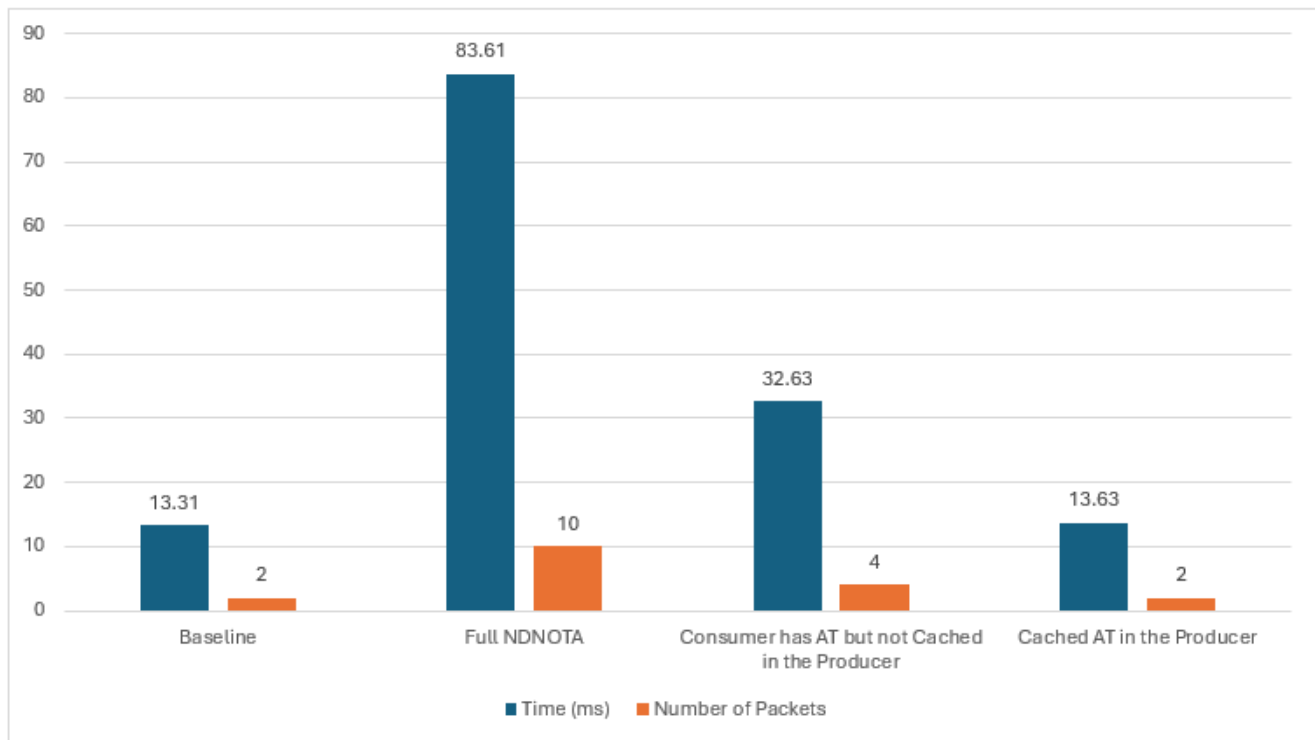
Table 3 shows the duration needed to generate Data packets by the AuthServ in response to consumer and producer registration, authentication, and verification requests during the NDNOTA phases. The first column presents the time taken for parsing Application Parameters and Interest signature elements in the received NDN Interest packet, i.e., HAR, AR, and Is-Consumer-Authenticated Interest packets. The second column indicates the processing time necessary for the following tasks: (1) handling parsed information from the aforementioned Interest packets, such as generating random tokens, comparing tokens, and encrypting/decrypting tokens, (2) establishing and modifying the consumer session entry, and (3) generating the AuthMsg and incorporating it into three Data packets according to the NDNOTA phase and the received Interest. The third column includes an explanation of the processing steps involved.

**Table 3.** Average parsing and processing duration ( $\mu$ s) in the AuthServ.

| Interest Message Type                  | Parsing Duration ( $\mu$ s) | Processing Duration ( $\mu$ s) | Processing Duration Description                                                                                                                                                                                                                                                                                         |
|----------------------------------------|-----------------------------|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Handshake Authentication Request (HAR) | 45.406                      | 4738.321                       | <ol style="list-style-type: none"> <li>1. Extract consumer's public key and key locator</li> <li>2. Generate and encrypt CT</li> <li>3. Establish a session entry</li> <li>4. Prepare AuthMsg state 3, CT</li> <li>5. Prepare Data packet with the prepared AuthMsg</li> </ol>                                          |
| Authentication Request (AR)            | 38.720                      | 4960.469                       | <ol style="list-style-type: none"> <li>1. Extract CT and key locator</li> <li>2. Decrypt CT</li> <li>3. Compare CT with stored CT</li> <li>4. Generate and encrypt AT</li> <li>5. Modify the session entry</li> <li>6. Prepare AuthMsg state 5, AT</li> <li>7. Prepare Data packet with the prepared AuthMsg</li> </ol> |
| Is-Consumer-Authenticated              | 51.885                      | 1827.104                       | <ol style="list-style-type: none"> <li>1. Extract AT and key locator</li> <li>2. Compare AT with stored AT</li> <li>3. Prepare AuthMsg state 8, True</li> <li>4. Prepare Data packet with the prepared AuthMsg</li> </ol>                                                                                               |

As indicated in Table 3, the parsing process duration, which includes extracting the AuthMsg and signature pointers, is almost identical for all three Interest packets. Additionally, the processing duration is the longest for the authentication request Interest packet due to its higher processing requirements. Nevertheless, the NDNOTA authentication process takes approximately 11,661.905  $\mu$ s from the AuthServ side, with the majority of this time allocated to processing received Interest packets.

Figure 6 shows the time and packet requirements for delivering the content in various scenarios. Table 4 provides further elaboration on the details presented in Figure 6. The first scenario considered the baseline, which involves delivering public content. The second scenario involves the full NDNOTA authentication process when the consumer requests private content for the first time. In the third scenario, the consumer possesses an AT but has not requested private content from this producer, meaning the consumer's key locator and AT are not cached in the producer database. The final scenario demonstrates the time and packet count needed to provide the consumer with private content when the producer has cached the consumer's key locator and AT in its database.



**Figure 6.** Authentication process time and number of packets.

**Table 4.** Average time ( $\mu$ s) and number of NDN packets required to serve consumer with requested content.

| Scenario                                                     | Time ( $\mu$ s) | Number of Interest Packets | Interest Packets                                                                             | Number of Data Packets | Data Packets                                                                                | Total Packets |
|--------------------------------------------------------------|-----------------|----------------------------|----------------------------------------------------------------------------------------------|------------------------|---------------------------------------------------------------------------------------------|---------------|
| Baseline                                                     | 13,313.211      | 1                          | Normal content request                                                                       | 1                      | Normal content                                                                              | 2             |
| Full NDNOTA                                                  | 83,609.248      | 5                          | Private content request, HAR, AR, private content request with AT, Is-Consumer-Authenticated | 5                      | Redirect, Handshake Reply, Authentication Reply, Is-Consumer-Authenticated, private content | 10            |
| Consumer has <i>AT</i> but <i>not cached</i> in the producer | 32,627.304      | 2                          | Private content request with AT, Is-Consumer-Authenticated                                   | 2                      | Is-Consumer-Authenticated, private content                                                  | 4             |
| <i>Cached AT</i> in the producer                             | 13,631.168      | 1                          | Private content request with AT                                                              | 1                      | Private content                                                                             | 2             |

As evident from Table 4, when the entire process is executed, indicating that Bob lacks an authentication token, the procedure takes approximately 70 ms longer than requesting public content, which serves as the baseline. Moreover, the total number of traffic messages increases to 10 instead of two in the scenario where the complete process is carried out. However, if the consumer possesses an authentication message and seeks to request content from a producer other than Alice, the process will take an additional 19 ms compared to the baseline, accompanied by an increase of two packets. Conversely, when the consumer already has an authentication token stored and cached at the producer, the authentication process remains unaffected, maintaining the same number of packets.

Table 5 presents the delay and the number of NDN packets involved in responding to the consumer's request from the producer's perspective across different scenarios. In the baseline situation, the consumer requested normal public content. In the second scenario, when the consumer requested private content, the producer dispatched a Redirect Data packet. Subsequently, consumer registration and authentication occurred via the AuthServ. The producer then spent time verifying the consumer's AT in the AuthServ, which is

average around 18,996 ( $\mu$ s), and time issuing the private content in the Content element of a Data packet, which is the second scenario. Notably, the third scenario took nearly the same amount of time as the first scenario because the consumer's AT and key locator were cached in the producer's database, obviating the need for communication with the AuthServ.

**Table 5.** Average producer responding delay ( $\mu$ s).

| Scenario                          | Response Delay ( $\mu$ s) | Number of Packets                                                                           |
|-----------------------------------|---------------------------|---------------------------------------------------------------------------------------------|
| Baseline                          | 5070.941                  | 1<br>(Public content in Data packet)                                                        |
| Private content for first time    | 24,524.307                | 2<br>(1 'Is-Consumer-Authenticated'<br>Interest packet<br>1 private content in Data packet) |
| Private content when AT is cached | 5386.789                  | 1<br>(private content in Data packet)                                                       |

As indicated in Table 5, if the consumer ID is not cached in the producer database, the producer response experiences a delay of approximately 19,453  $\mu$ s compared to responding with normal public content in order to authenticate the consumer. However, when the consumer ID and its authentication token (AT) are cached, the response time aligns with that of responding with normal content.

## 5.2. NDNOTA Operations Latency

In this experiment, a Python method named '*Crypto.Random.get\_random\_bytes(N)*' was utilized to generate both tokens. This method produces a random byte string with a length of  $N$ , which in our experiment was set to 16 bytes [35]. Additionally, a Python class named '*Cryptodome.Cipher*' was used [36] to combine asymmetric cryptography with the Advanced Encryption Standards–Galois Counter Mode (AES–GCM) method, a block cipher mode of AES, in order to secure the encryption and decryption processes for generated tokens. AES–GCM was chosen for its ability to deliver efficient processing and low-latency performance in both encryption and decryption processes. The AES–GCM method utilizes the generated token and the ECC public key, with a prime size of 256 bits, as an encryption key to generate a cipher message. This process also generates a random nonce and an authentication tag, which functions as a Message Authentication Code (MAC). Furthermore, the components including the cipher message, private key, nonce, and tag are employed in the AES–GCM decryption process.

Table 6 displays the latency and delay resulting from the token's generation, encryption, and decryption. The experiment was conducted 1000 times, and the averages were taken.

**Table 6.** NDNOTA operation average latency ( $\mu$ s).

| Token Generation Duration ( $\mu$ s) | Token Encryption Duration ( $\mu$ s) | Token Decryption Duration ( $\mu$ s) |
|--------------------------------------|--------------------------------------|--------------------------------------|
| 2.315                                | 2603.133                             | 1720.810                             |

As evidenced by previous experimental results, NDNOTA offers successful single authentication for various NDN services with minimal time and message overhead. Essentially, when a consumer authenticates for a service for the first time, it takes approximately 80 milliseconds with an additional 10 NDN messages for the authentication process. However, if the consumer already possesses an authentication token, equivalent to a username and password in today's networks, the process takes nearly 30 milliseconds with an extra four messages. Furthermore, if the consumer's identity and token are cached, the NDNOTA process takes just about 10 milliseconds with only two additional exchanged messages, which is negligible considering the network time.

## 6. Conclusions and Future Work

This paper introduces a framework called NDN One-Time Authentication (NDNOTA) for authenticating NDN over-Internet services online. This framework enables an NDN consumer to gain access to private NDN content, applications, or services using a login credential. NDNOTA achieves this objective through three key mechanisms: (1) leveraging single sign-on (SSO), (2) employing token credentials and certified asymmetric keys, and (3) utilizing signed Interest and Data packets. The authentication process in NDNOTA encompasses four phases and necessitates the use of a proposed authentication message (AuthMsg), which is incorporated as part of the Interest Application Parameters element or Data Content element. The theoretical security analysis demonstrates that NDNOTA provides confidentiality and privacy to NDN packets and is resilient to integrity attacks like MITM, impersonation, and replay attacks. Moreover, the findings from practical tests indicate that the initial NDNOTA full authentication process imposes a delay of approximately 70 ms on the consumer. However, once the consumer possesses an authentication token stored in both its database and the producer's database, the authentication process becomes instantaneous, and retrieving private data occurs in the same timeframe as requesting public data. Moreover, it took 19.454 milliseconds for the producer to respond to the consumer's request for private content for the first time.

Enhancing the NDNOTA framework could involve exploring a different library or programming language, beyond Python. Moreover, NDNOTA token revocation is considered a primary constraint encountered by the proposed NDNOTA that must be considered as a topic for future exploration. Finally, the resilience of NDNOTA against identity attacks will be assessed by employing various real attack tools.

For the convenience of the reader, the abbreviations used in this paper are listed in the Abbreviations.

**Author Contributions:** Conceptualization, M.A. (Manar Aldaoud), D.A.-A., F.K., and M.A. (Medhat Awadalla); methodology, M.A. (Manar Aldaoud); validation, M.A. (Manar Aldaoud); formal analysis, M.A. (Manar Aldaoud); investigation, M.A. (Manar Aldaoud); resources, M.A. (Manar Aldaoud); data curation, M.A. (Manar Aldaoud); writing—original draft preparation, M.A. (Manar Aldaoud); writing—review and editing, M.A. (Manar Aldaoud), D.A.-A., F.K., and M.A. (Medhat Awadalla); visualization, M.A. (Manar Aldaoud); project administration, M.A. (Manar Aldaoud) and D.A.-A.; funding acquisition, M.A. (Manar Aldaoud). All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data sharing does not apply to this article as no new data were created or analyzed in this study.

**Acknowledgments:** The authors would like to thank Sultan Qaboos University for financially supporting this research and publication.

**Conflicts of Interest:** The authors declare that they have no competing interests.

## Abbreviations

|          |                                  |
|----------|----------------------------------|
| AR       | Authentication request           |
| AT       | Authentication token             |
| AuthMsg  | Authentication message           |
| AuthServ | Authentication server            |
| CT       | Challenge token                  |
| HAR      | Handshake authentication request |
| NDN      | Named Data Networking            |

NDNOTA    NDN One-Time Authentication  
SSO        Single sign-on

## References

1. Singh, V.P.; Ujjwal, R.L. A walkthrough of name data networking: Architecture, functionalities, operations and open issues. *Sustain. Comput. Inform. Syst.* **2020**, *28*, 100419. [CrossRef]
2. Tao, Y.; Zhu, Y. An interest-based access control scheme via edge verification in Named Data Networking. *Int. J. Commun. Syst.* **2022**, *35*, e5169. [CrossRef]
3. Kumar, S.; Wollinger, T. Fundamentals of Symmetric Cryptography. In *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*; Lemke, K., Paar, C., Wolf, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 125–143. [CrossRef]
4. Ahmad, J.; Mohammad, C.W.; Sadiq, M. Generation of One-Time Password for the Authentication of Software Requirements Using Secure Hash Algorithms. In *Proceedings of the International Conference on Recent Trends in Computing*; Springer Nature: Singapore, 2022; pp. 627–640.
5. Zhang, Z.; Liu, S.; King, R.; Zhang, L. NDN-MPS: Supporting multiparty authentication over named data networking. In *Proceedings of the 8th ACM Conference on Information-Centric Networking*, Paris, France, 22–24 September 2021; pp. 83–94.
6. Trnka, M.; Cerny, T.; Stickney, N. Survey of Authentication and Authorization for the Internet of Things. *Secur. Commun. Netw.* **2018**, *2018*, 4351603. [CrossRef]
7. Beltran, V.; Bertin, E.; Crespi, N. User Identity for WebRTC Services: A Matter of Trust. *IEEE Internet Comput.* **2014**, *18*, 18–25. [CrossRef]
8. FORCEPOINT. End User Single Sign-On for Forcepoint Web Security Cloud. Available online: [https://www.websense.com/content/support/library/web/hosted/sso\\_guide/how\\_sso\\_works.aspx](https://www.websense.com/content/support/library/web/hosted/sso_guide/how_sso_works.aspx) (accessed on 24 October 2023).
9. Aldaoud, M.; Al-Abri, D.; Awadalla, M.; Kausar, F. Data Structure and Management Protocol to Enhance Name Resolving in Named Data Networking. *Future Internet* **2024**, *16*, 118. [CrossRef]
10. Aldaoud, M.; Al-Abri, D.; Awadalla, M.; Kausar, F. Towards a Scalable Named Data Border Gateway Protocol. In *Proceedings of the 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, Maldives, 16–18 November 2022; pp. 1–5.
11. Kumar, N.; Singh, A.K.; Aleem, A.; Srivastava, S. Security Attacks in Named Data Networking: A Review and Research Directions. *J. Comput. Sci. Technol.* **2019**, *34*, 1319–1350. [CrossRef]
12. Chatterjee, T.; Ruj, S.; Bit, S.D. Security Issues in Named Data Networks. *Computer* **2018**, *51*, 66–75. [CrossRef]
13. Shah, M.S.M.; Leau, Y.B.; Anbar, M.; Bin-Salem, A.A. Security and Integrity Attacks in Named Data Networking: A Survey. *IEEE Access* **2023**, *11*, 7984–8004. [CrossRef]
14. Zhang, Z.; Yu, Y.; Zhang, H.; Newberry, E.; Mastorakis, S.; Li, Y.; Afanasyev, A.; Zhang, L. An Overview of Security Support in Named Data Networking. *IEEE Commun. Mag.* **2018**, *56*, 62–68. [CrossRef]
15. Anjum, A.; Agbaje, P.; Mitra, A.; Oseghale, E.; Nwafor, E.; Olufowobi, H. Towards named data networking technology: Emerging applications, use cases, and challenges for secure data communication. *Future Gener. Comput. Syst.* **2024**, *151*, 12–31. [CrossRef]
16. Kapetanidou, I.A.; Sarros, C.-A.; Tsaoussidis, V. Reputation-Based Trust Approaches in Named Data Networking. *Future Internet* **2019**, *11*, 241. [CrossRef]
17. Benmoussa, A.; Kerrache, C.A.; Lagraa, N.; Mastorakis, S.; Lakas, A.; Tahari, A.E.K. Interest Flooding Attacks in Named Data Networking: Survey of Existing Solutions, Open Issues, Requirements, and Future Directions. *ACM Comput. Surv.* **2022**, *55*, 139. [CrossRef]
18. Nour, B.; Khelifi, H.; Hussain, R.; Mastorakis, S.; Mounsla, H. Access Control Mechanisms in Named Data Networks: A Comprehensive Survey. *ACM Comput. Surv.* **2021**, *54*, 61. [CrossRef]
19. Chen, T.; Lei, K.; Xu, K. An encryption and probability based access control model for named data networking. In *Proceedings of the 2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, Austin, TX, USA, 5–7 December 2014; pp. 1–8.
20. Hamdane, B.; Boussada, R.; Elhdhili, M.E.; Fatmi, S.G.E. Towards a Secure Access to Content in Named Data Networking. In *Proceedings of the 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, Poznan, Poland, 21–23 June 2017; pp. 250–255.
21. Backes, M.; Cachin, C.; Oprea, A. Lazy revocation in cryptographic file systems. In *Proceedings of the Third IEEE International Security in Storage Workshop (SISW'05)*, San Francisco, CA, USA, 13–13 December 2005; p. 11.
22. Shang, W.; Yu, Y.; Liang, T.; Zhang, B.; Zhang, L. NDN-ACE: Access Control for Constrained Environments over Named Data Networking. *NDN Technical Report NDN-0036*. 2015. Available online: <https://named-data.net/wp-content/uploads/2015/12/ndn-0036-1-ndn-ace.pdf> (accessed on 24 October 2023).
23. Misra, S.; Tourani, R.; Natividad, F.; Mick, T.; Majd, N.E.; Huang, H. AccConF: An Access Control Framework for Leveraging In-Network Cached Data in the ICN-Enabled Wireless Edge. *IEEE Trans. Dependable Secur. Comput.* **2019**, *16*, 5–17. [CrossRef]
24. Htet Hlaing, H.; Funamoto, Y.; Mambo, M. Secure Content Distribution with Access Control Enforcement in Named Data Networking. *Sensors* **2021**, *21*, 4477. [CrossRef] [PubMed]

25. Li, Q.; Zhang, X.; Zheng, Q.; Sandhu, R.; Fu, X. LIVE: Lightweight Integrity Verification and Content Access Control for Named Data Networking. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 308–320. [[CrossRef](#)]
26. Abdellah, A.; Saif, S.M.; ElDeeb, H.E.; Abd-Elrahman, E.; Taher, M. A Survey of Using Blockchain Aspects in Information Centric Networks. In *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2020*; Springer International Publishing: Cham, Switzerland, 2021; pp. 292–301.
27. Benmoussa, A.; Kerrache, C.A.; Calafate, C.T.; Lagraa, N. NDN-BDA: A Blockchain-Based Decentralized Data Authentication Mechanism for Vehicular Named Data Networking. *Future Internet* **2023**, *15*, 167. [[CrossRef](#)]
28. Shi, J.; Zeng, X.; Han, R. A Blockchain-Based Decentralized Public Key Infrastructure for Information-Centric Networks. *Information* **2022**, *13*, 264. [[CrossRef](#)]
29. Abdellah, A.S.; Saif, S.; ElDeeb, H.E.; Abd-Elrahman, E.; Taher, M. A Secured Blockchain-based Information-Centric Network. *J. Comput. Sci.* **2022**, *18*, 266280. [[CrossRef](#)]
30. NDN-Team. Signature. Available online: <https://docs.named-data.net/NDN-packet-spec/current/signature.html#> (accessed on 30 October 2023).
31. NDN-Team. Named-Data/Ndn-Cxx. Available online: <https://github.com/named-data/ndn-cxx/tree/master> (accessed on 10 February 2024).
32. NDN-Team. Signed Interest. Available online: <https://docs.named-data.net/NDN-packet-spec/current/signed-interest.html> (accessed on 10 February 2024).
33. Zhang, J.; Li, S.; Wang, C. A Secure Dynamic Content Delivery Scheme in Named Data Networking. *Secur. Commun. Netw.* **2022**, *2022*, 6304927. [[CrossRef](#)]
34. Hevey, D. Network analysis: A brief overview and tutorial. *Health Psychol. Behav. Med.* **2018**, *6*, 301–328. [[CrossRef](#)] [[PubMed](#)]
35. Crypto.Random Package: v3.19.0. Available online: <https://pycryptodome.readthedocs.io/en/v3.19.0/src/random/random.html> (accessed on 6 November 2023).
36. Cryptodome.Cipher. Available online: <https://pycryptodome.readthedocs.io/en/latest/src/api.html> (accessed on 13 November 2023).

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.