Article

DCSS Protocol for Data Caching and Sharing Security in a 5G Network

Ed Kamya Kiyemba Edris [1], Mahdi Aiash [1,*] and Jonathan Loo [2]

[1] School of Science and Technology, Department of Computer Science, Middlesex University, London NW4 4BT, UK; ee351@live.mdx.ac.uk
[2] School of Computing and Engineering, University of West London, London W5 5RF, UK; jonathan.Loo@uwl.ac.uk
* Correspondence: m.aiash@mdx.ac.uk

Abstract: Fifth Generation mobile networks (5G) promise to make network services provided by various Service Providers (SP) such as Mobile Network Operators (MNOs) and third-party SPs accessible from anywhere by the end-users through their User Equipment (UE). These services will be pushed closer to the edge for quick, seamless, and secure access. After being granted access to a service, the end-user will be able to cache and share data with other users. However, security measures should be in place for SP not only to secure the provisioning and access of those services but also, should be able to restrict what the end-users can do with the accessed data in or out of coverage. This can be facilitated by federated service authorization and access control mechanisms that restrict the caching and sharing of data accessed by the UE in different security domains. In this paper, we propose a Data Caching and Sharing Security (DCSS) protocol that leverages federated authorization to provide secure caching and sharing of data from multiple SPs in multiple security domains. We formally verify the proposed DCSS protocol using ProVerif and applied pi-calculus. Furthermore, a comprehensive security analysis of the security properties of the proposed DCSS protocol is conducted.

Keywords: 5G; security protocol; network services; federated identity; data caching; data sharing; authorization; formal methods; ProVerif; applied pi calculus

1. Introduction

Fifth Generation mobile networks (5G) promise to push services to the edge; mobile subscribers will be able to access these with their User Equipment (UE) ubiquitously. After accessing services such as content data, the UE will be able to request other processes such as caching and sharing of the accessed data. 5G also promises to enable seamless connectivity, secure access to the network and services, security will be provided by authentication methods such as those specified in [1]. The initial security will be provided locally in the Home Network (HN) using primary authentication protocols to authenticate the UE to access the network [2,3]. The next security stage will be provided by the HN and external data networks (DN) using a variety of security mechanisms that are interoperable in the HN and Service Providers (SP) network via DN function of the 5G architecture [1,4,5].

With 5G supporting numerous shareholders including end-users, Mobile Network Operators (MNO), and SPs, accessing and provisioning of services will require security mechanisms that protect the shareholders and the services [4]. There is also a need for permission delegation propagation and sharing between users to allow SPs to offer specific services beyond their security domain and still have some control over the accessed data by the user. Services will be provisioned at the edge, close to the user by multiple SPs from different security domains, hence being exposed to various security threats. Therefore, this will require unified, interoperable, robust, and multi-purpose security solutions such as federated authentication and authorization mechanisms [4]. The use of Federated

Identity (FId) in multi-tenant network infrastructure with security procedures from trusted and semi-trusted third-parties can provide access to the right delegation, flexible security management, and precise tracking of critical UE data [2]. The MNO and SP will be able to transfer their security management to a third-party and allow permission delegation to end-users [6,7].

As users in 5G will be able to cache and share data supported by Information-Centric Networking (ICN) [8–10] to improve access and caching, security measures should be in place to authorize what the end-users can do with the accessed data. This can be facilitated by service authorization mechanisms that apply access controls and encryption techniques that restrict the caching and sharing of data by the UE. The use of federated authorization procedures that use the use of a single digital Identity (ID) of a user in different domains' scenarios, MNOs will be able to provide security and seamless session continuity as users move from one network to another [7]. However, there is a lack of security mechanisms that can provide efficient data caching and sharing authorization in 5G. Therefore, we propose a Data Caching and Sharing Security (DCSS) protocol to provide service authorization to the UE for secure caching, sharing of data, and access rights delegation. To the best of our knowledge, no protocol in 5G provides data caching and sharing with permission delegation, facilitated using federated authorization, and no formal analysis on data caching and sharing security protocol.

Our contributions in this paper are summarized as follows. We explore how federated identity, access controls, and permission delegation can be used to provide service authorization in 5G. We propose a DCSS protocol that enables the UE to cache and share data securely with access rights delegation in 5G. Furthermore, we use a formal method approach with ProVerif and applied pi-calculus to formally verify the proposed DCSS protocol. We also conduct a comprehensive security analysis on the protocol's security properties based on two taxonomies.

The rest of the paper is structured as follows. The related work on data security in 5G is presented in Section 2. The problem definition and concepts on federated security, access controls, delegation are discussed in Section 3. Section 4 presents the proposed DCSS protocol, architecture overview, security requirements, and assumptions of the proposed scheme, while Section 5 presents the modelling of the proposed DCSS protocol. The formal verification of the DCSS protocol is in Section 6. Section 7 analyzes the security properties of the protocol. Finally, the paper concludes in Section 8.

2. Related Work

In addressing 5G security, much attention has been given to physical layer and network security, providing the UE with secure access to the network. The related work in [11–16] explored different physical layer security techniques such as artificial noise injection to improve channel quality, anti-eavesdropping signal methods and secure beamforming to enhance the spatial distribution properties of the transmitted signal, while the related work on network security [3,17–21] analyzed the 3GPP specified protocols [1] for flaws by formally analysing the protocols' security properties and suggesting some improvements. These include enhancement of diameter protocol security, ensuring that Security Anchor Function (SEAF) should never initiate unsolicited authentication request after synchronization failure, and adding a Subscriber's Permanent Identifier (SUPI) in the message sent from Authentication Server Function (AUSF) to SEAF when confirming the session key after as successful primary authentication between the UE and HN.

Different security solutions have been proposed specifically to address security at the network level, which provides the basis of the service level security. The authors in [22] discussed mobile edge computing and its security issues and proposed a multi-server protocol that providing authentication, key establishment between UEs, privacy protection, and UE tracking by the SP. In [23], the authors explored identity protection in legacy and 5G networks. They proposed an anonymous access authentication scheme, providing the users' identities protection, analyzed and verified with formal methods and BAN logic.

The authors in [24] explored multi-server and small cells in heterogeneous networks. They considered authentication, key exchange, data storage, and transmissions. They analyzed the protocol in [25] that claimed to provide security in multi-server architecture in 5G but had some flaws. They improved the protocol by proposing some improvements. In [26], authors explored Wireless Sensors Networks (WSN) and IoT in 5G. Base on the work in [27] that provided authentication and access control scheme, they introduced a system architecture that integrated IoT with 5G and proposed a formally analyzed ECC-based security solution providing authentication, authorization, key agreement, and privacy-preservation for WSN in 5G.

Most of the related work on service security discussed caching schemes and storage techniques, not much on the service security mechanisms. The authors in [28] proposed cooperative edge caching architecture for caching and computing resources at the wireless network edge in 5G. In [29], the authors proposed a caching and sharing optimization solution to assist sensor networks in 5G for content delivery to the UE. Enabling ICN with edge computing improves communication and content distribution, the authors in [30] proposed an ICN-capable Radio Access Network (RAN) architecture for edge computing in 5G that offers caching at the edge. Also, [31] discusses CCN-based caching in 5G and proposes an edge caching scheme based on content-centric networking.

On service security in 5G, the authors in [32] discussed malicious access and destruction of data and proposed a label-based access control mechanism that provides authentication to authorized nodes to ensure authenticity and integrity of data using embedded labels, while [33] proposed a security mechanism that uses hash function technique to protect data sharing in for 5G networks. In [34], the author proposed an authentication and access control mechanism for 5G applications that can be delegated to the 3rd parties SPs at the edge, providing security at the network and service level. The authors in [4] proposed a federated authentication and authorization protocol that provides secure access to services from multiple SPs and single sign-on to users using federated identity for seamless access to services in 5G.

However, most of the related work discussed in this paper did not consider the security issues raised by integrating ICN with 5G. In addition, those who explored the security issues did not consider the multiple shareholders, data security, and the need for a unified, interoperable solution in 5G [2]. With network and service access security addressed, content caching, and sharing authorization of the accessed content needed addressing. Moreover, their proposed security solutions were not formally analyed using formal methods and no security analysis was conducted on their protocols' security properties for security guarantees. Based on the above observation, we intend to address both data caching and sharing using federated ID, access right delegation, and access controls that use user and data capabilities explained in the next section.

3. Data Caching and Data Sharing Security in 5G

This section discusses some of the concepts used in this paper. After being granted access to the service, the UE should able to access data, request further authorization to cache, and share data. It should also be able to perform its own activity such as delegating its access rights to another UE. This could be achieved with the assistance of the SP or independently depending on the service policies, network coverage conditions and the delegation permission [4]. We are more interested in the content access and retrieval process, which deals with content delivery. With content delivery and sharing between end-users in proximity of each other, caching at UE level is possible with the use of content-centric networking (CCN)-based framework to enable traffic offloading, caching at the edge and content dissemination from the SP to UE via baseband unit pool (BBU) [9,35].

3.1. Federated Authorization

The end-user will be able to access services from numerous SPs controlled by Identity Provider (IdP) using the same digital identification [7]. Following a successful federated

authentication between the UE and the IdP, the IdP will be able to send the FId and security context to the SP for end-user verifications and access authorization, [36]. This is supported by authentication and authorization mechanisms such as access controls, security protocols and the OAuth2 framework [37], which also supports Single Sign On (SSO) to provide a common login credentials for users across multiple systems.

The access controls facilitate authorization by granting the end-user access to the services, verified against the user's profile and a permission list [38]. Some of the access controls used are Role-based Access Control (RBAC) [39], Encryption-based Access control EBAC [40], Attribute-based access control ABAC [41], and Capabilities-based Access control CBAC [42]. These access control mechanisms use different techniques to give different access rights, attributes, capabilities, and encryption to a subject to access an object in the form of security tokens.

3.2. Permission Delegation

Permission delegation allows the assigning of access rights to an end-user by an administrative user or another end-user. It requires a user to have the ability to use the access right being delegated but not the administrative user [43]. A federated delegation mechanism can be applied to the capability generation and propagation process for authorization and capability revocation management. Using ABAC and CBAC with FId in a content-aware mobile network could efficiently address challenges in access control strategy processing. By delegating part of the authentication and authorization tasks to other users in different security domains, it supports the 5G security and services access objectives. Processing validation of capability in the HN and third-party SPs enables a flexible, elastic, context-aware, and fine-grained access control mechanism in 5G.

3.3. Capabilities and Attributes

The access control technique is based on attributes and capabilities with security context from the federated identity-based protocol [4], which are used to enable the UE to request caching and sharing permission of the data. With capabilities, the mechanism leverages CBAC and ABAC to enable the UE to obtain the data cache and share authorization from the SP. The capabilities techniques used are based on the abilities defined in [42] and labels defined in [44] as token claims. After being granted access to the service, the UE retrieves the data and then sends another request to SP for caching and sharing authorization. The SP generates a security token that defines the object and subject abilities, the data is the object and UE is the subject. The generated tokens are cache and share tokens with capabilities and attributes parameters, the tokens are similar in structure, but their roles differ. They are linked with the data object name and the UE, which is used to define objects and subjects' abilities. As in [4,44], dot-separated sequence of numbers are used for an ability in form of a label whereby an ability is a string $.i_1.i_2.i_3 \ldots i_n$ of value n where $i_1, i_2, i_3 \ldots, i_n$ are integers. An example of abilities for a UE or Network Data Object (NDO) will be .1.2.3.4, or 01.02.03.04.

During the authorization, procedure tokens are created and sent to the relevant entities for verification. Both the data and UE will be given labels as part of security tokens validation and verification. Access to a data object is granted if the data object's label is a prefix of the UE's label. Whereby, a data object with a label "2.0.0" with abilities "0.1.0" for caching and "0.0.1" for sharing could only be accessed by UE with abilities such as ".2.0.0", ".2.1.0", ".2.1.1"…etc, i.e., whenever an authenticated UE requests data, it needs to provide the right label to SS that confirms its rights to access, cache, or share the data. The SP creates the labels and links to the subject and object, the UE cannot promote themselves to access other data objects but the permission can be delegated to other UE if delegation was part of the initial request of the access token [4]. The labels are integrated into the security tokens as access rights. Any other subject to access the data will require the nonce key to decrypt that data object as the data can only be accessed with the access token or nonce key both generated by SP. The label consists of user capabilities (ucap) and data capabilities

(dcap), label = $(ucap, dcap)$. A generic security structure and attributes are described in Table 1.

Table 1. Generic security token structure.

Attribute	Description
Token ID	security token identifier
Issuer	ID of the token issuer
Issue date	timestamp when token was created
Issue Sign	Token's digital signature
Subject	UE ID granted the rights of the token
Service	SP associated with token
Audience	the target entity for token
Nonce	authentication nonce (optional)
Expiry date	time when token becomes invalid
Access rights	set of attribute and capabilities (Label)
Security Info	nonce key ID
Scope	set of conditions (type of grant, token and offline access setting)

3.4. Problem Definition

With 5G promising to offload data traffic from the backhaul to the fronthaul, it will facilitate the pushing of content to the edge closer to the user. The content dissemination will facilitated by CDN technologies such CCN. This is due to the end-users' increasing demand for quick access and downloads of multimedia content, to enable this service, users with their UEs will be allowed to cache and share content. This raises security concerns at the edge and while the UE is accessing, caching, and sharing the content. Related work has suggested new ways to access service in 5G [4] but there is still a lack of security mechanism that can address the security concern while the UE is caching and sharing data.

Our research proposes a multi-tiers security framework as shown in Figure 1 below that addresses security at three levels namely:

- Network: Deals with securing connections at the network level
- Service: addressees security challenges between devices and service providers to access and cache data.
- D2D: support secure sharing of data aiming devices with or without network operator involvement.

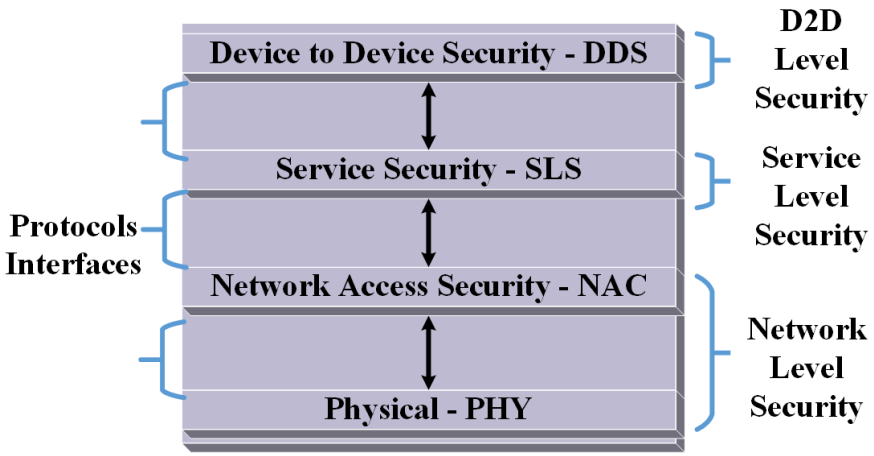

Figure 1. Security Framework.

This paper presents a potential protocol to address the security challenges at the service level because it enables mobile devices-having already been authenticated at the network level-to be authorized by service provider to cache and share data, while the work

in [45] presents protocols at the D2D level to achieve mutual authentication among devices prior to data sharing.

4. The Proposed Data Caching and Sharing Security (DCSS) Protocol

This section presents the proposed Data Caching and Sharing Security (DCSS) Protocol that leverages the [46], security [1] and Service-based Architecture (SBA) [47] of 5G with Federated Identity Management (FIdM) and enhanced access controls supporting the provision of network services and slicing [2]. This protocol relies on primary authentication protocols such as 5G-AKA [3] to authenticate the UE on the HN and federated authorization protocols such as [4,32] to authorize the UE access the SP services. The federated security process is controlled by IdP via trust and Public Key Infrastructure (PKI), after this process, the UE is assigned federated ID (FId) and SSO. Therefore, the UE will be able to use those credentials during the run of DCSS protocol.

The proposed 5G DCSS protocol has following advantages:

- It uses the security context from other service authorization protocols [4] that provides the UE with access to the data such as keys and IDs. Hence encouraging security integration as specified in [2] without compromise the security.
- It provides two options to the UE in terms of how to use the data that is caching and sharing it with other UEs.
- It can be used for caching or sharing or for both.
- It allows the UE delegate its access permission, if it was permitted by the SP.
- The data security is considered by use of subject and object labels that gives the UE and the data capabilities aligned with user's access rights.

4.1. Architecture Overview

The proposed DCSS protocol leverages FIdM model [7] in Figure 2 aligned with 5G system [46], security entities [1], and federated entities. It can be applied in both 5G core network (5GC) or third-party SP environment. It uses IdPs, third-party SP Authentication, Authorization, and Accounting (AAA) servers, and Service Servers (SS) to provide federated security. To enable FIdM, it can also be coupled with 5G systems and security entities. It enables the redefinition of the UE ID settings as well as the sharing of the security context such as keys, tokens, nonces, and IDs, both inside and outside the 5GC. The following entities are defined for DCSS protocol and may play many roles:

- UE: end-user is the entity trying to access the service.
- SMF: 5G HN Session Management Function (SMF) manages the session between the UE and HN and acts as a pass-through authenticator.
- SPAAA: The AAA servers of the SP. It verifies the ID of the user, checks the authenticity of the request, and generates authorization tokens used by the UE to request services.
- SS: service server, which hosts the service and grants access to the protected services.

The UE registers to the MNO prior to initial authentication to 5GC via the SEAF controlled AUSF, whereby the user's subscription data and security context are stored in Unified Data Management (UDM)/Authentication Credential Repository and Processing Function (ARPF) [1,3]. Simultaneously, the MNO registers the UE with SP based on the service agreement and access policies. The MNO and third-party SP also must agree on the security process and parameters such as PKI mechanism, access control policies, identities, authentication, and authorization levels protocols to be used [4]. In addition, the SP registers its services, access control policies, shared secret and keys, SP identity (SPID), and credentials to the IdP. To complete the trust and federated security interoperability the HN registers the UE's GPSI, user attributes, credentials, and Home Network ID (HNID). This can be managed by an internal or external entity such as the IdP to implement federated security in 5G [7].

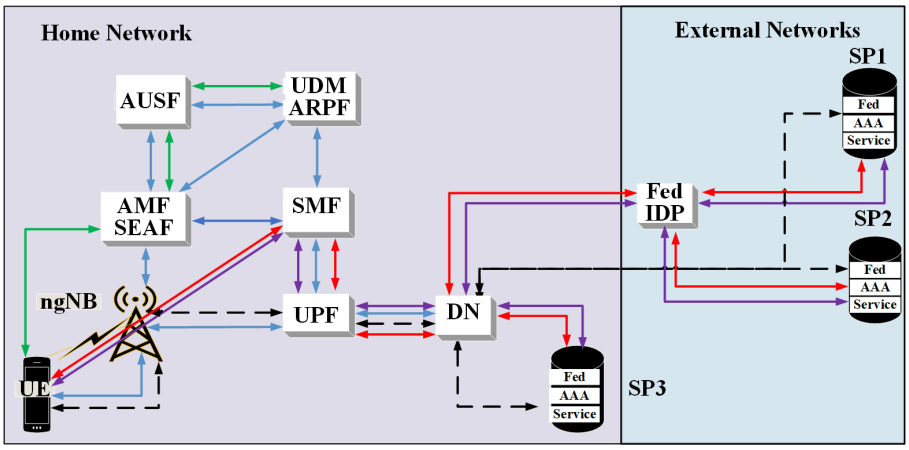

Figure 2. 5G FIdM Model.

Then the UE obtains authenticated to access the 5G network using 3GPP primary authentication protocols such as 5G authentication and key agreement (AKA) protocol [3]. After being granted access to the network, the UE requests access to a service which in this case is provided by the third-party SP, the authentication and authorization are provided by protocols such as [4,5], respectively hence the UE gains access to the service, access/fresh tokens and SSO for further requests such as caching and sharing. The fresh token can also be used to refresh its permissions if the original token is expired or to request permissions to other services.

4.2. Data Caching and Data Sharing Authorization

The DCSS protocol intends to provide authorization to the UE to cache and share the data accessed using the service authorization protocols such as NS-FId protocol [4]. After accessing the data, the UE will have the option to cache or share data or do both, hence, the need for cache and share security. The proposed protocol is divided into caching and sharing authorization phases, uses cache token ChT and share token ShT for caching and sharing requests, respectively comprising claims, labels, and access rights delegation. The token structure similar defined in [4], the tokens however have some additional claims and scope to achieve their objectives. Both cache and share tokens should include the UE and data object capabilities, delegation ability in form of label bitstring, and the security context such as nonce key ID. The nonce key could be used to extend data security with encryption and decryption options. The use of labels enable the caching and sharing of the data even when the SP involvement is minimal or direct authorization is no existent, for example, in areas with no network coverage.

The UE uses the access token and security context from federated authorization protocol [4] such as the keys to request caching authorization of the data accessed. The security token is timestamped, has an expiry date that deems the token invalid, enforcing the UE to request a new token or refresh the expired token [37]. It also includes an identifier, hence a security token will have the following tuple, similar to those in [4]: Token = ID, label, timestamp, expiry date, so the cache token is $ChT = (fid, ssid, d1, label, Ts_1, Exp_1)$. The use of timestamps and the expiry date reduces the risk of replay attacks. The security token is assigned to the respective entities $UE, SPAAA, SS$ and digitally signed by the $SPAAA$ securely to provide the integrity and authenticity of the token. The data caching security provides authorization to UE to cache the accessed data.

After Caching the data, the UE uses the cache token and security context such as ACK_1 strings to request sharing authorization of the data accessed and now cached. The data sharing security provides authorization for the UE to share the accessed data. The UE in HN or VN might want to share the cached data with another UE, to be able to do so the UE must request permission from the SP to grant the UE access to data. The Share Token is $ShT = (fid, d1, label, Ts_2, Exp_2)$.

The security token can be deemed invalid if it used after the expiry date and revoked if it is misused or the attributes do not match the requested data attributes in the SS database. This is followed with negative acknowledgement (NACK) to the UE from the SS.

4.3. Security Assumptions and Requirements

We assume that the UE would have achieved primary authentication to HN and service authorization to the SP services. The UE will be in possession of the pre-shared key K_{UE3A}, K_{UESS} and the access token AcT as the results of those procedures. The expected security properties to be achieved by the DCSS protocol are data confidentiality, integrity, authenticity, and entity authentication. Even though the DCSS is intended to provide data caching and share authorization, to harden the security in 5G, this protocol will re-authenticate the participating entities. These security properties are defined informally before being formalized, two taxonomies are adopted for precise and meticulous security analysis [4]. First, the security properties of the protocol are specified from an agent A's point of view, with defined between two agents A and B as set 1 [48];

- aliveness,
- weak agreement,
- non-injective agreement
- injective agreement

Secondly, the security protocol should meet the following security properties as set 2 [49];

- mutual entity authentication
- mutual key authentication
- mutual key confirmation
- key freshness
- unknown-key share
- key compromise impersonation resilience.

5. Modelling of DCSS Protocol

In this section, the proposed protocol is modelled using entities; UE, $SPAAA$ and SS leveraging on system architecture shown in Figure 2. We have omitted SMF as it has no significant role in the protocol execution. The protocol applies cryptographic primitives such as hash function, symmetric/asymmetric encryption, digital signature, and Message Authentication Code (MAC), access, and cache tokens to provide message authentication, confidentiality, integrity, and non-repudiation. Its security assumptions are based on the 5G specifications [1], while the Universal Subscriber Identity Module (USIM) in the UE has cryptographic capabilities such as key agreement, Key Derivation function (KDF), encryption facilitated by Elliptic Curve Integrated Encryption Scheme (ECIES) [50].

Protocol Message Exchange

We now give an overview of DCSS protocol execution and message exchange illustrating the protocol's full execution. It involves the following parties, `UE`, `SPAAA`, and `SS` and consists of cache and share authorization phase. The UE is granted access to the SP services using federated authorization protocol [4], which is when the access token AcT and K_{UESS} are generated before. The K_{UE3A} is a preshared key between the UE and $SPAAA$, the UE is in possession of the access token AcT before this proposed protocol is run. In addition, the UE and SS would also know each other's ID, public key, and shared session key from the previous message exchange of the authentication and authorization protocol that granted the UE access to the service. The use of signed digested messages is to provide message authenticity, integrity, and non-repudiation. The protocol messages between the parties are illustrated in Figure 3, with reference to notations in Table 2 and described in detail below:

Table 2. DCSS Protocol Notation and Description.

Notation	Description
SPID	SP identifier
SID	session ID
SSID	authorization server ID
DNN	service code:SPID
R1	nonce
EID	UE permanent identifier
PK_{SP}	SPAAA public key
K_{UE3A}	UE and SPAAA preshared key
K_{UESS}	UE and SS session key
Ack_1	acknowledgement
Hack_1	hash for Ack_1
Ack_2	acknowledgement
Hack_2	hash for Ack_2
Exp	expiry date
D1	dataname
Ts	timestamp
label	capabilities strings
ACT	access token
ChT	cache token
ShT	share token
h(x)	hash value (h) of message (x)
{x}{k}	message (x) encrypted with key K

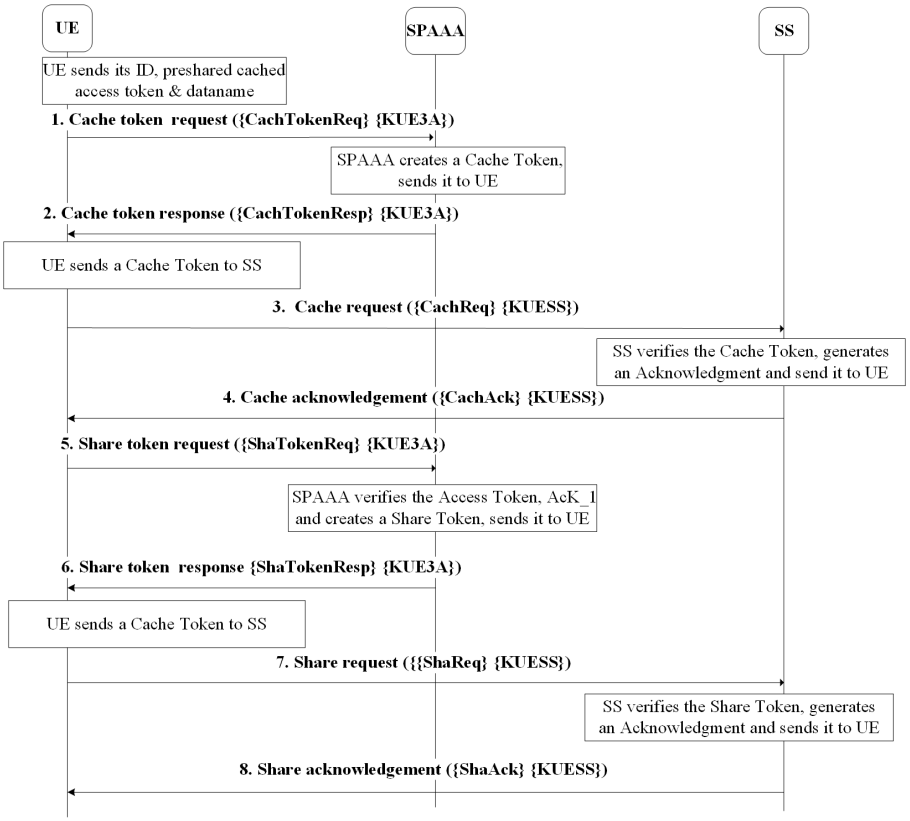

Figure 3. DCSS Protocol Message Exchange Flow.

Caching Authorization

`Msg1.UE→SPAAA: ({CachTokenReq},{KUE3A})`

After the UE is granted access to the data, it sends a cache request *CachTokenReq* for a

cache token. It includes the UE's IDs (EID/FID), data name $DataName$, access token AcT, the hash of the access token $hact$ signed by the $SPAAA$ private key SK_{AAA} and key K_{UESS} to $SPAAA$ encrypted with preshared key K_{UE3A}. The Act, $hAcT$, SK_{AAA}, K_{UE3A} and K_{UESS} are generated during the authentication and authorization procedure between UE and $SPAAA$ to grant the UE access to the service. Key K_{UESS} is included if it was not sent during the previous authentication and authorization procedure, while the access token AcT supports verification of the user's ID and subscription since it includes the UE subscription details.

`Msg2.SPAAA→UE: ({CachTokenResp},{KUE3A})`

When the $SPAAA$ receives message 1, the authorization server verifies the AcT. If the subscription policy includes caching rights, $SPAAA$ creates cache token ChT and sends it with a hash of the cache token $hChT$ signed with its private key SK_{AAA} in cache response $CachResp$ message encrypted with session key K_{UE3A}.

`Msg3.UE→SS: ({CachReq},{KUESS})`

After receiving the cache token ChT from $SPAAA$ in message 2, the UE sends a cache request message $CachReq$ encrypted with preshared key K_{UESS} to the service server SS requesting authorization to cache the data in its possession. The message includes a ChT and a signed hash of the cache token $hChT$ with $SPAAA$ private key SK_{AAA}.

`Msg4.SS→UE:({CachAck},{KUESS})`

When the SS receives $CachReq$ in message 3, it verifies the cache token ChT, if it is valid then it acknowledges the UE's request to cache the data by sending an acknowledgement Ack_1 in $CachAck$ message encrypted with K_{UESS}, which also includes the hash of the Acknowledgement $hAcK_1$ signed with the SS private key SK_{SS}.

Sharing Authorization

`Msg5:UE→SPAAA:({ShaTokenReq},{KUE3A})`

Now that the UE is authorized to cache the data, it sends a share token request message $ShaTokenReq$ to $SPAAA$ to obtain sharing authorization of the cached data. The message includes access token AcT and cache acknowledgement Ack_1, for validation and to notify the authorization server that it has permission to cache the data that it wishes to share.

`Msg6:SPAAA→UE:({ShaTokenResp},{KUE3A})`

When $SPAAA$ receives message 5, it verifies the access token, checks if the subscription policy includes sharing rights then sends a share response message $ShaTokenResp$ to the UE. The message includes a share token ShT, hash of the share token $hShT$ signed with SK_{AAA} and encrypted with the K_{UE3A} that authorizes the UE to share the content.

`Msg7:UE→SS: ({ShaReq},{KUESS})`

When the UE receives message 6, it sends a share request message $ShaReq$ to SS encrypted with K_{UESS} that includes a share token ShT and its hash $hShT$ that were sent to UE in message 6 by $SPAAA$.

`Msg8:SS→UE:({ShaAck},{KUESS})`

The SS verifies the received share token parameters in message 7 and sends an acknowledgement Ack_2 with its hash $hAck_2$ in a share acknowledgement message $ShaAck$ to UE acknowledging the request to share the data.

6. Verifying of DCSS Protocol

This section formally models and verifies the proposed protocols using formal methods, ProVerif, and security properties are formalized with ProVerif results.

6.1. Formal Method Approach

Formal method approach with automated protocol verifier have been used in the verification of mobile network authorization protocols to for security properties guarantees [4,51]. However, most verification techniques and tools find these properties very challenging and hard to deal with, due to the cryptographic primitive's algebraic properties used, some tools and manual proof checks are not suitable for symbolic reasoning. For protocol formal analysis, there are a variety of automated verification methods available, such

as Automated Validation of Internet Security Protocols and Applications (AVISPA) [52], Tamarin [53] and ProVerif [54].

ProVerif [54] is an automatic tool for analyzing security protocols, with Dolev-Yao [55] as the adversary, and it supports equational theories which are defined by users and as well as enabling the verification of security properties. It supports the underlying theory of abstraction, but it may also lead to false attacks. Applied pi-calculus [56] is used as a formal language to describe and model security protocols. The syntax is paired with formal semantics to enable reasoning about protocols. With a variety of cryptographic primitives modelled by equations and rewrite rules. In addition, the security properties with observational equivalence properties are proved as inputs in form of secrecy and authentication. The information is translated into an internal representation of the protocol which makes some abstraction crucial to an unbounded number of sessions. Functions are used to model cryptographic primitives, while terms built over an infinite set of names such as (a, b, c, ...) with an infinite set of variables such as (x, y, z, ...) and a finite set of function symbols such as $f1, \ldots, fn$ represent the messages. A set of reduction rules describes how the application of function symbols to terms is affected. Table 3 shows the syntax and grammar of ProVerif process language and more details can be found in [54]. As a result, we believe ProVerif an appropriate tool for this analysis. This tool has been used in the formal verification of security properties of 5G authentication [3], authorization [4] and federated [57] protocols for security guarantees.

Table 3. Core Language: Term and Process Grammar.

Term	Grammar
a, b, c, k, s	name
x, y, z	variable
M,N ::=	terms
h(D1, ...,Dn)	function application
f(M1,...,Mn)	constructor application
D ::=	expressions
fail	failure
P,Q ::=	processes
out(N,M); P	output
in(N, x : T); P	input
!P	!P replication
0	nil
P | Q	parallel composition
new a : T; P	restriction
let x : T = D in P else Q	expression evaluation
if M then P else Q	conditional

6.2. Formal Verification Using ProVerif

In ProVerif, the protocol modelling is composed of declaration, process macros, and main processes. With queries used in the rectification of the correctness and secrecy of a protocol. Using declarations of types, functions, queries, and events, the ProVerif code is used to effectively specify the protocol. Free names are free variables that are known to the public, globally known whereas bound names are locally known by the process such as the public channel for communication, [private] excludes names from the attacker [54].

Specification includes the following:

- Functions: `fun sign(bitstring, sskey): bitstring,`
- Key: `type key.`
- Private and public names: `free fid:id [private]. free kuess:key [private] free pubChannel: channel.`
- Queries: Queries on secrecy, reachability, and authentication. A query of the attacker's knowledge `attacker(M)` is used to specify a secrecy property. The attacker

may have knowledge of `M` if the fact `attacker(M)` can be derived from the horn clauses. There is no way for the attacker to learn about `M` if the fact `attacker(M)` cannot be derived from the clauses. With reachability, the query `query attacker(K)` is also used to debug the model of the protocol to check a particular branch is reachable or not. `query k: bitstring; event(endServer(k))`. The correspondence assertions are used to specify authentication properties as `event(e1(M)) event(e2(M))`. If all clauses that conclude `event e1` include `event e2` in their hypotheses, then `event e1` can only be derived when `event e2` is true, proving the correspondence assertion [54]. In the case of the DCSS protocol, the following is queried: `query attacker (eid)`. `query attacker (kue3a)` are used to test the secrecy of the message, *FID* and key K_{UESS}, respectively, while `query U:host, SS:host, K:key; event (endSS (U,SS,K)) ==> event(beginUE (U,SS,K))` is used to test events relationships (authentication).

- Events: Querying events using correspondence assertion to test the relationship between events. (i) Event correspondence uses syntax to query a basic correspondence assertion, `query x1 :t1, . . . , xn :tn ; event (e (M1, . . . ,Mj)) ==> event(e'(N1, . . . , Nk) )`. Where terms `M1, . . . ,Mj,N1, . . . ,Nk` are built by the constructors applied to the variables `x1 , . . . , xn` of types `t1, . . . , tn` and `e, e'` are events. (ii) While the injective correspondence assertions are used to capture one-to-one relationship and denoted as `query x1:t1, . . , xn:tn ; inj-event(e(M1, . . . ,Mj)) ==> inj-event (e'(N1, . . . , Nk) )`. The correspondence asserts that there is a distinct earlier occurrence of the `event e'(N1, . . . ,Nk)` for each occurrence of the `event e(M1, . . . ,Mj)`. [54].
- Process: The protocol encoded using the main process and the process macros for the participating entities to allow sub-process being defined; `((!procUE(hostU))` for the UE `(!procAAA(hostA))` for the SPAAA and `(!procSS (hostSS))` for SS. The main process also starts off with several copies of the system entities *UE*, *SPAAA*, *SS* using the required parameters denoting several roles sessions as explained in the message exchange.

6.3. DCSS Protocol Formal Analysis

The protocol was modelled and simulated using secure, insecure channels and processes representing entities *UE*, *SPAAA* and *SS*. ProVerif found an attack on the protocol as shown in Figure 4, on a public channel (`UE <===> SPAAA <==> SS`). We were concerned with federated ID *FID*, the cache token *ChT*, share token *ShT*, and access token *AcT* properties. There were attacks on the protocol as a result of a formal analysis based on the adversary vector. The ProVerif results indicate that the secrecy of *ChT*, *ShT*, *AcT*, *FID*, *Ack*_1 and *Ack*_2 holds. The authentication event is queried as non-injective and injective agreements between UE and SS, which does not hold, SS may end the protocol assuming it is talking to UE, even though the UE never runs the protocol with SS.

```
ededris@ededris-VirtualBox:~/proverif2.00$ ./proverif protocols/DCSS.pv | grep RES
RESULT not attacker(Ack_1[]) is true.
RESULT not attacker(Ack_2[]) is true.
RESULT not attacker(kue3a[]) is true.
RESULT not attacker(kuess[]) is true.
RESULT not attacker(fid[]) is true.
RESULT not attacker(d1[]) is true.
RESULT not attacker(ChT[]) is true.
RESULT not attacker(ShT[]) is true.
RESULT event(endSS(U,SS,K)) ==> event(beginUE(U,SS,K)) is true.
RESULT event(endUE(U_112,SS_113,K_114)) ==> event(beginSS(U_112,SS_113,K_114)) is
false.
RESULT inj-event(endSS(U_115,SS_116,K_117)) ==> inj-event(beginUE(U_115,SS_116,K_1
17)) is true.
RESULT inj-event(endUE(U_118,SS_119,K_120)) ==> inj-event(beginSS(U_118,SS_119,K_1
20)) is false.
RESULT (even event(endUE(U_23118,SS_23119,K_23120)) ==> event(beginSS(U_23118,SS_2
3119,K_23120)) is false.)
```

Figure 4. DCSS Protocol Attack Results.

The `event beginSS` means that the protocol was completed by SS, UE received message 8 with Ack_2 after sending $Sht, hShT$ in message 7, `event beginUE` means that the SS received message 3. The protocol parameters are taken as arguments by the events: K_{UESS} and cache token ChT and its claims, SS which must verify the cache token and digital signature. If the arguments are true, then acknowledgement is sent otherwise it ends sends an invalid token. We would like to prove the correspondences `event (endSS(U, SS, K)) ==> event (beginUE(U, SS, K)). inj-event (endSS(U, SS, K)) ==> inj-event (beginUE(U,SS, K)).`

In ProVerif the direct proof of this correspondence does not hold because message 7 and message 8 sent after query `query []; event (endSS()) event(beginUE())` fail to hold due the attacker's knowledge of `A'_15173` and `kue3a_15172`, `attacker(A'_15173), attacker(kue3a_15172).` The attacker may obtain `(hostU[],hostU[],kue3a_15172). attacker((hostU [],hostU[],kue3a_15172))` as explained in attacker trace below. We further attempt to prove and conclude the desired correspondence by noting that an event with the argument *KUESS* cannot be executed before Ack_2 has been sent, i.e., before the Ack_2 is generated by executing the share request message *ShaTokenReq* with the share response message *ShaTokenResp*. One part of correspondence holds with true (basic) while the other does not hold in ProVerif with false (one to one).

6.3.1. The Attack on DCSS Protocol

Based on the attack derivation (abstracts) or attack trace (semantics), the attack on a protocol can be outlined. The derivation in ProVerif illustrates how an attacker can breach the protocol's security properties using abbreviations for the internal representation of names or terms. This is in form of a numerical list of steps, each of which corresponds to a process or an action taken by the attacker. The attack trace, on the other hand, is an executable trace of the considered process that depicts the real attack. The trace is a sequence of public channel inputs and outputs, as well as events related to the process. The input, output, or event is followed by their position in the process at n, which refers to the program point at the process's start. When ProVerif is given `query attacker (M)`, where M is the message transmitted on the channel c, the intention is trying to prove a state in which a property is unreachable by showing `not attacker (M)`, in that the `RESULT not attacker (M) is true`, the attacker ends up without `term (M)` meaning that the attack does not have `(M)`, while `query x1 : t1, . . . , xn :tn ; event (e(M1, . . . ,Mj)) ==> event(e'(N1,. . . ,Nk))`, tests the events' relationship. It is asserted that for each occurrence of the event, there must be a well defined earlier occurrence of another event. The man-in-the-middle as the first attack, the attacker I starts by eavesdropping on the communication between entities, leading to two further attacks found by ProVerif. The attacker impersonates UE continuing the protocol with SS, which completes the protocol with the attacker instead of the UE.

```
UE -> I_SPAAA: {CachTokenReq}, {KUE3A}
I_UE -> SPAAA: {CachTokenReq}, {KUE3A_15172}
SPAAA -> I_UE: {CachTokenResp}, {KUE3A}
I_SPAAA -> UE: {CachTokenResp}, {KUE3A_15172}
UE -> I_SS: {CachReq}, {KUESS}
I_UE -> SS: {CachReq}, {KUESS}
SS -> I_UE: {CachAck}, {KUESS}
I_SS -> UE: {CachAck}, {KUESS}
UE -> I_SPAAA: {ShaTokenReq}, {KUE3A}
I_UE -> SPAAA: {ShaTokenReq}, {KUE3A_15172}
SPAAA -> I_UE: {ShaTokenResp}, {KUE3A}
I_SPAAA -> UE: {ShaTokenResp}, {KUE3A_15172}
UE -> I_SS: {ShaReq}, {KUESS}
I_UE -> SS: {ShaReq}, {KUESS}
SS -> I_UE: {ShaAck}, {KUESS}
```

```
I_SS -> UE: {ShaAck}, {KUESS}
```

6.3.2. Attack Derivation and Trace

TRACE 1

```
1 The attacker has some term A'_15173. attacker(A'_15173).
2 The attacker has some term kue3a_15172. attacker(kue3a_15172).
3 The attacker initially knows hostU[]. attacker(hostU[])
4 Using the function 3-tuple the attacker may obtain (hostU[],
hostU[],kue3a_15172). attacker((hostU[],hostU[], kue3a_15172)).
5 new skue: skey creating skue_15181 at {1}
6 new skss: sskey creating skss_15182 at {2}
7 new skaaa: sskey creating skaaa_15183 at {3}
8 out(c, ~M) with ~M = pk(skue_15181) at {6}
9 out(c, ~M_15346) with ~M_15346 = spk(skss_15182) at {8}
10 out(c, ~M_15427) with ~M_15427 = spk(skaaa_15183) at {10}
11 in(c, a_15179) at {12} in copy a_15180
12 in(c, (hostU,hostU,a)) at {124} in copy a_15178
13 get keys(hostU,hostU,a) at {57} in copy a_15180
14 new skue_37: skey creating skue_15552 at {13} in copy a_15180
15 new fid_39: id creating fid_15554 at {15} in copy a_15180
16 new ucap: bitstring creating ucap_15560 at {21} in copy a_15180
17 get keys(hostU,hostSS,kuess) at {56} in copy a_15180
18 event endUE(hostU,hostSS,kuess) at {25} in copy a_15180 (goal)
19 The event endUE(hostU,hostSS,kuess) is executed.
20 A trace has been found.
21 RESULT event(endUE(U_110,SS_111,K_112)) ==>
event(beginSS(U_110,SS_111,K_112)) is false.
```

TRACE 2

```
22 event endUE(hostU,hostSS,kuess) at {25} in copy a_19431 (goal)
23 The event endUE(hostU,hostSS,kuess) is executed in session a_19431.
24 A trace has been found.
25 RESULT inj-event(endUE(U_116,SS_117,K_118)) ==>
inj-event(beginSS(U_116,SS_117,K_118)) is false.
```

The attacker's actions are explained in derivations and trace steps concisely as shown in Figure 5, with some text omitted for simplicity. The two attack traces follow similar steps but have varying inputs, outputs, and events as follows.

- In trace 1, Line 1–4 are steps taken by the attacker, indicating that attacker may know UE, therefore may also know key `kue3a` using the 3-tuple function in line 4. The attacker may have received message A'_15173 by eavesdropping on the public channel at input {12} uses this knowledge to obtain *kue3a*_15172. Line 5–7 at input {1}–{4} corresponds to creation and insertion of keys on public channel. Line 9–11 at output {6}–{10} correspondence with the attacker saving the keys in new variables `~M = pk(skue_15181)`, `~M_15346 = spk(skss_ 15182)` and `~M_15427 = spk (skaaa_15183)` for reuse later. Line 11 at {12} the attacker has session `copy a_15180` with *UE* pretending to be *SPAAA* as *I_SPAAA* sending `a_15179` to *UE* after eavesdropping on the channel. On line 12 the attacker obtains message `hostU,hostU,a` that includes the key for session between *UE* and *SS* in session `copy a_15178`. The attacker eavesdrops on insertion of keys K_{UESS} at input {56} and *KUE3A* input {57}, respectively, in session `copy a_15180`. On line 14 *UE* creates primitives at {13}–{21}, sends them to *SPAAA* start the protocol run. On line 18 the attacker was able to achieve is his goal in session `copy a_15180 (goal)` with *UE*, when the `event endUE(hostU,hostSS,kuess)` is executed. With *UE* ending the protocol

thinking it was talking to SS while UE never run the protocol with SS. The attack is on non-injective agreement on line 21.

- In trace 2, the same steps, inputs, and outputs but in different sessions. In this trace the attacker achieves his goal in session `copy a_19431 (goal)` with UE, when the `event endUE(hostU, hostSS, kuess)` is executed. The attack is against injective agreement a one-to-one relationship correspondence in session `copy a_19431` on line 25.

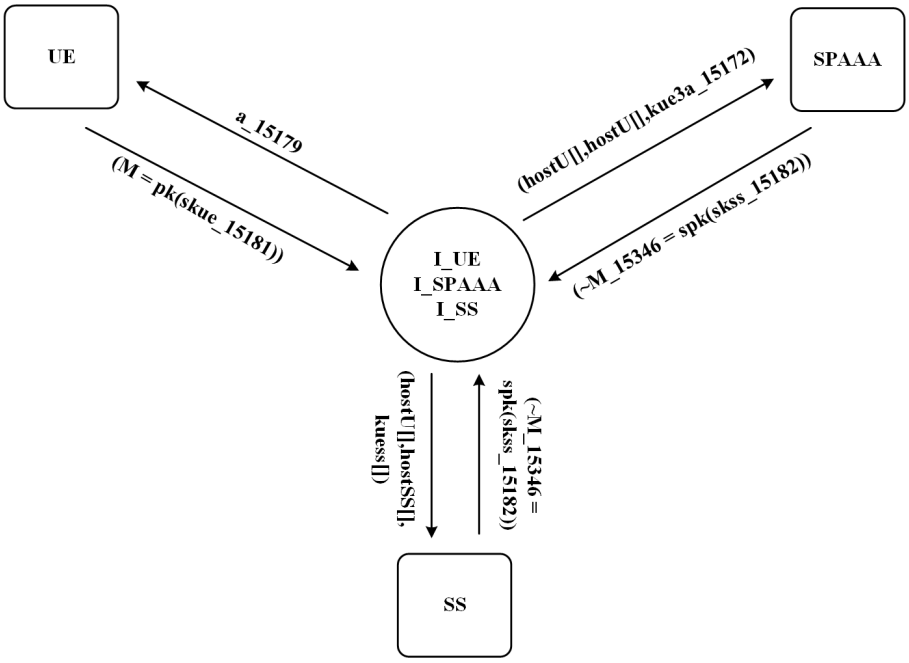

Figure 5. DCSS Protocol Attack Trace.

6.4. Improved Version of DCSS Protocol

Some changes were made in the modelling of the protocol to address the security threats discovered in the previous version and an improved version of the DCSS protocol is presented. To improve this protocol the UE send ID (FID), hash $hFID$ and to $SPAAA$ in message 1 and to SS in message 3, $SPAAA$ sends ID $(SPID)$ and hash $hSPID$ to UE in message 2 and SS sends ID $(SSID)$ and its hash $hSSID$ in message 4. This prevents the attacker from using the information it has on UE and impersonating the UE when trying to start sessions with other entities.

After changes were made, the protocol was modelled and run in ProVerif again, there was no attack found as shown in Figure 6, hence the protocol is secure. The ProVerif results indicate that the secrecy of `Ack_1, Ack_2, Fid, d1, ChT, ShT` holds as shown in Table 4. The authentication events in form of non-injective and injective agreements hold that is `query []; event (endSS()) event(beginUE())` is true.

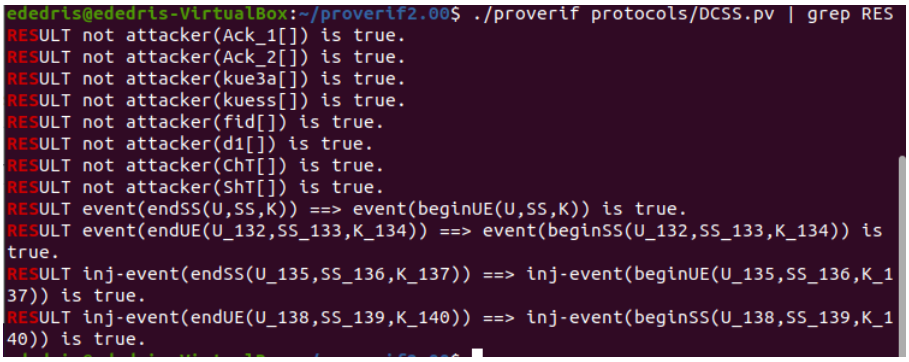

Figure 6. DCSS Protocol Safe Results.

Table 4. Proverif Query Checks.

Properties	Query	Expected Output	Proverif Output
ChT	Secrecy	True	True
ShT	Secrecy	True	True
FID	Secrecy	True	True
AcK_1	Secrecy	True	True
Ack_2	Secrecy	True	True
UE-SS	Non-injective Agreement	True	True
SS-UE	Injective agreement	True	True

7. Security Analysis

The security analysis of the DCSS Protocol security properties based on two taxonomies and discusses the security consideration of the protocol.

7.1. Protocol Security Analysis

In our threat model, a Dolev Yao model [55] is used as the adversary which controls the network communication channel. It is capable of reading, intercepting, modifying, and sending messages. Eavesdropping, manipulation, interception, impersonation, and message injection are some of the attacks it can initiate. Hashing, encryption, and sign-on values that are known to the attacker can also be used by the attacker. The analysis is based on a symbolic model, which assumes perfect cryptography, but the computational strengths of the primitives are ignored. The protocol, however, must meet specific security properties and the analysis of the protocol is based on security requirements sets 1 and 2 [48] and 2 [49], respectively.

7.1.1. The Analysis Based on Set 1

- Secrecy: The FID and $D1$ are never revealed to the attacker, hence secrecy is achieved. Also property covers the data confidentiality and privacy.
- Aliveness: The $SPAAA$ obtains the aliveness of UE when the UE sends a cache authorization request to $SPAAA$ with an access token, while SS does when the UE sends cache or share the message. The SS responds with acknowledgement strings, (Ack_1 and Ack_2) for tokens (ChT and ShT), respectively.
- Weak Agreement: When $SPAAA$ achieves weak agreement in form of non-injective agreement on access token with UE and SS achieves this when Acknowledgement strings is received by UE.
- Non-injective Agreement: This is achieved when the UE obtains a non-injective agreement on access token with the $SPAAA$. The SS achieves this on ChT, ShT with

the UE, when it is generated by $SPAAA$ and sent to UE. The tokens ChT/ShT includes labels, therefore SS obtains the assurance on ChT/ShT from the $SPAAA$.

- Injective Agreement: The tokens between the $SPAAA$ and SS are fundamental to the protocol's security goal. The injective agreement on labels with the SP assures the UE that SS is known and trusted. The UE obtains the injective agreement on ChT, ShT and AcT with the $SPAAA$ and SS respectively to assure that the sessions with SS and were authorized the $SPAAA$. Simultaneously, SS is assured over tokens used that its session with UE was authorized by $SPAAA$.

7.1.2. The Analysis Based on Set 2

- Mutual Entity Authentication: Since the UE already authenticated with $SPAAA$ and uses SSO for further authorization, it uses the access token to acquire both cache/share tokens from $SPAAA$, resulting in implicit authentication between SS and UE. In addition, reverifies the access token, the SS verifies the ChT/ShT and in return, it agrees to UE caching/sharing request by sending Acknowledgements. The FID and $Ack1/Ack2$ proved to hold, implying that this requirement is also enforced.
- Mutual Key Authentication: This property is not required as the involved parties are in possession of session keys K_{UE3A} and K_{UESS}.
- Mutual Key Confirmation: Despite that the keys are preshared, the requirement met by the successful run of the protocol between the UE, $SPAAA$, and UE, SS.
- Key Freshness: Although there is no function in ProVerif to validate key freshness, the $SPAAA$ verifies if the AcT expiry date and timestamp supplied in the message enforce the session's freshness. However, because the keys' secrecy has not been compromised, it is assumed that they are new.
- Unknown-Key Share: ProVerif uses the reachability property to check aliveness in a protocol. This attack is prevented by the entities' IDs and key binding. This requirement is demonstrated by the inclusion of IDs (FID, $SPID$), and access token AcT parameters; $fid, ssid, label$ in the authorization process.
- Key Compromise Impersonation Resilience: The UE, $SPAAA$ and SS are in possession of K_{UE3A} and K_{UESS} when the protocol starts which are pre-shared keys in this protocol. Since the secrecy property of all data exchanged holds hence, they enforce this requirement. Further enforcement of this requirement is achieved when digital signatures are used in message exchange this protocol.

The UE will be able to refresh the access token and request new tokens for other services as defined by the SP via access control claims and security parameters. The use of FID and security tokens enable the UE and SP to start new authentication and process that might require new parameters. With SSO the UE can access granted service and request other services which reduce exposure to adversaries. The protocol is proved to be secure after formally analysing it using ProVerif. With DCSS the UE will be able to request authorization from SP to cache the data as an option. This protocol provides authorization, implicit authentication to the UE and SP. The implementation of the DCSS protocol will enable the UE to cache the data and security tokens for further uses such as sharing with other UEs leveraging on SSO, which provides secure communication, improving user experience, and use of network resources.

After the successful run of the protocol, the UE should achieve data cache and sharing authorization for the UE to cache and share the data accessed from SS. It will also achieve entity authentication and message authentication. This protocol is an optional protocol that might be used if the UE needs to cache or share data. It also depends on if data have caching abilities and if the UE has been given the ability to cache or share the data. In the case where neither ability of the subject or the object match then the caching/sharing authorization will be denied. This is achieved using labels and delegation of access permissions. The messages between entities are encrypted and signed by the sender in terms of the tokens they are signed by the $SPAAA$).

With 5G enabling seamless access to edge services and promising to improve user experience, it is paramount to address the legacy mobile system's security issues. Hence, DCSS protocol will address security issues at the service level, for example when the UE is granted access to a service and permission to cache the data but not to share it. The UE will include its request if intends to cache and share the data. Whereby during the generation of security tokens, the right access permissions and labels can be included in tokens and the data security.

8. Conclusions

5G promises to push services to the edge, closer to the mobile end-users, and they will be able to use their UEs to access services from multiple SPs seamlessly. After accessing the services, the UE will then be able to request other services such as caching and sharing of the accessed data and delegate permission to other users. However, it requires a robust federated service authorization mechanism to apply access controls and delegation that restrict the caching and sharing of data access by the UE. In this paper, we explored authorization of data caching and sharing of accessed services using FId, access control, and delegations in 5G. We proposed a federated protocol DCSS that can provide authorization of data caching and sharing in 5G and SP networks. We used formal methods and the automated proof verifier ProVerif to model and formally analyze the DCSS protocol. We also looked at the protocol and its security properties using two different security property taxonomies. End-users could use this protocol to request other operations with their data securely in heterogeneous network, such as 5G. Future work will be about data caching and sharing in communication applications such as D2D communications.

Author Contributions: Conceptualization, E.K.K.E., M.A. and J.L.; methodology, E.K.K.E. and M.A.; validation, E.K.K.E. and M.A.; formal analysis, E.K.K.E.; investigation, E.K.K.E.; writing—original draft preparation, E.K.K.E.; writing—review and editing, E.K.K.E. and M.A.; supervision, M.A. and J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. 3GPP. Security architecture; procedures for 5G system. In *Technical Specification (TS) 3GPP TS 33.501 V17.0.0 (2020-12)*; Third Generation Partnership Project: Sophia Antipolis, France, 2020.
2. 5GPPP. *Deliverable D2.7 Security Architecture (Final)*; Technical Report for 5G ENSURE; 5G ENSURE: Brussels, Belgium, 2017.
3. Edris, E.K.K.; Aiash, M.; Loo, J. Formal Verification and Analysis of Primary Authentication based on 5G-AKA Protocol. In Proceedings of the 2020 Seventh International Conference on Software Defined Systems (SDS), Paris, France, 20–23 April 2020.
4. Edris, E.K.K.; Aiash, M.; Loo, J. Network Service Federated Identity (NS-FId) Protocol for Service Authorization in 5G Network. In Proceedings of the Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 20–23 April 2020.
5. Edris, E.K.K.; Aiash, M.; Loo, J.; Alhakeem, M.S. Formal Verification of Secondary Authentication Protocol for 5G Secondary Authentication. *Int. J. Secur. Networks* **2021**, In Press.
6. 5GPPP. *5G PPP White Paper: Phase 1 Security Landscape*; Technical Report for 5GPPP; 5GPPP: Brussels, Belgium, 2017
7. Edris, E.K.K.; Aiash, M.; Loo, J. The Case for Federated Identity Management in 5G Communications. In Proceedings of the Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 20–23 April 2020.
8. Chandrasekaran, G.; Wang, N.; Hassanpour, M.; Xu, M.; Tafazolli, R. Mobility as a Service (MaaS): A D2D-Based Information Centric Network Architecture for Edge-Controlled Content Distribution. *IEEE Access* **2018**, *6*, 2110–2129. [CrossRef]
9. Ravindran, R.; Suthar, P.; Trossen, D.; Wang, C; White, G. Enabling ICN in 3GPP's 5G NextGen Core Architecture. In *IETF (The Internet Engineering Task Force) Request for Comments*; IETF: Fremont, CA, USA, 2018.
10. Edris, E.K.K.; Aiash, M.; Loo, J. Investigating Network Services Abstraction in 5G enabled Device-to-Device (D2D) Communications. In Proceedings of the IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Leicester, UK, 19–23 August 2019; pp. 1660–1665. [CrossRef]
11. Sun, L.; Du, Q. Physical layer security with its applications in 5G networks: A review. *Commun. China* **2017**, *14*, 1–14. [CrossRef]
12. Wu, Y.; Khisti, A.; Xiao, C.; Caire, G.; Wong, K.K.; Gao, X. A Survey of physical Layer security techniques for 5G wireless networks and challenges ahead. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 679–695. [CrossRef]

13. Gao, Y.; Hu, S.; Tang, W.; Li, Y.; Sun, Y.; Huang, D.; Cheng, S.; Li, X. Physical Layer Security in 5G Based Large Scale Social Networks: Opportunities and Challenges. *IEEE Access* **2018**, *6*, 26350–26357. [CrossRef]
14. Noura, H.N.; Melki, R.; Chehab, A. Efficient data confidentiality scheme for 5g wireless NOMA communications. *J. Inf. Secur. Appl.* **2021**, *58*, 102781.
15. Nandan, N.; Majhi, S.; Wu, H.C. Beamforming and Power Optimization for Physical Layer Security of MIMO-NOMA Based CRN Over Imperfect CSI. *IEEE Trans. Veh. Technol.* **2021**. [CrossRef]
16. Wang, J.; Wang, X.; Gao, R.; Lei, C.; Feng, W.; Ge, N.; Jin, S.; Quek, T.Q. Physical Layer Security for UAV Communications in 5G and Beyond Networks. *arXiv* **2021**, arXiv:2105.11332.
17. Arkko, J.; Eronen, P.; Lehtovirta, V.; Torvinen, V. *Improved Extensible Authentication Protocol Method for 3GPP Mobile Network Authentication and Key Agreement (EAP-AKA)*; Rfc 5448, IETF: Fremont, CA, USA, 2020.
18. Arkko, J.; Norrman, K.; Näslund, M.; Sahlin, B. A USIM Compatible 5G AKA Protocol with Perfect Forward Secrecy. In Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 20–22 August 2015; Volume 1, pp. 1205–1209 [CrossRef]
19. Basin, D.; Dreier, J.; Hirschi, L.; Radomirović, S.; Sasse, R.; Stettler, V. A Formal Analysis of 5G Authentication. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1383–1396. [CrossRef]
20. Fang, D.; Qian, Y.; Hu, R.Q. Security for 5G Mobile Wireless Networks. *IEEE Access* **2018**, *6*, 4850–4874. [CrossRef]
21. Zhang, J.; Yang, L.; Cao, W.; Wang, Q. Formal Analysis of 5G EAP-TLS Authentication Protocol Using ProVerif. *IEEE Access* **2020**. [CrossRef]
22. Lee, J.; Kim, D.; Park, J.; Park, H. A Multi-Server Authentication Protocol Achieving Privacy Protection and Traceability for 5G Mobile Edge Computing. In Proceedings of the 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 10–12 January 2021; pp. 1–4. [CrossRef]
23. Ma, D.; Lyu, X.; Zou, R. A Novel Variable K-Pseudonym Scheme Applied to 5G Anonymous Access Authentication. *arXiv* **2021**, arXiv:2106.07158.
24. Wu, T.Y.; Lee, Z.; Obaidat, M.S.; Kumari, S.; Kumar, S.; Chen, C.M. An authenticated key exchange protocol for multi-server architecture in 5G networks. *IEEE Access* **2020**, *8*, 28096–28108. [CrossRef]
25. Wu, F.; Li, X.; Xu, L.; Sangaiah, A.K.; Rodrigues, J.J. Authentication protocol for distributed cloud computing: An explanation of the security situations for Internet-of-Things-enabled devices. *IEEE Consum. Electron. Mag.* **2018**, *7*, 38–44. [CrossRef]
26. Shin, S.; Kwon, T. A privacy-preserving authentication, authorization, and key agreement scheme for wireless sensor networks in 5G-integrated Internet of Things. *IEEE Access* **2020**, *8*, 67555–67571. [CrossRef]
27. Adavoudi-Jolfaei, A.; Ashouri-Talouki, M.; Aghili, S.F. Lightweight and anonymous three-factor authentication and access control scheme for real-time applications in wireless sensor networks. *Peer-Peer Netw. Appl.* **2019**, *12*, 43–59. [CrossRef]
28. Zhang, K.; Leng, S.; He, Y.; Maharjan, S.; Zhang, Y. Cooperative Content Caching in 5G Networks with Mobile Edge Computing. *IEEE Wirel. Commun.* **2018**, *25*, 80–87. [CrossRef]
29. Vo, N.S.; Duong, T.Q.; Guizani, M.; Kortun, A. 5G Optimized Caching and Downlink Resource Sharing for Smart Cities. *IEEE Access* **2018**, *6*, 31457–31468. [CrossRef]
30. Ullah, R.; Rehman, M.A.U.; Naeem, M.A.; Kim, B.; Mastorakis, S. ICN with edge for 5G: Exploiting in-network caching in ICN-based edge computing for 5G networks. *Future Gener. Comput. Syst.* **2020**, *111*, 159–174. [CrossRef]
31. Wang, T.X.; Chen, A.M.; Taleb, V.; Ksentini, V.; Leung, V. Cache in the air: Exploiting content caching and delivery techniques for 5G systems. *IEEE Commun. Mag.* **2014**, *52*, 131–139. [CrossRef]
32. Wang, Q.; Chen, D.; Zhang, N.; Qin, Z.; Qin, Z. LACS: A Lightweight Label-Based Access Control Scheme in IoT-Based 5G Caching Context. *IEEE Access* **2017**, *5*, 4018–4027. [CrossRef]
33. El-Latif, A.A.A.; Abd-El-Atty, B.; Venegas-Andraca, S.E.; Mazurczyk, W. Efficient quantum-based security protocols for information sharing and data protection in 5G networks. *Future Gener. Comput. Syst.* **2019**, *100*, 893–906. [CrossRef]
34. Behrad, S.; Bertin, E.; Tuffin, S.; Crespi, N. A new scalable authentication and access control mechanism for 5G-based IoT. *Future Gener. Comput. Syst.* **2020**, *108*, 46–61. [CrossRef]
35. Zhang, T.; Fang, X.; Liu, Y.; Nallanathan, A. Content-centric mobile edge caching. *IEEE Access* **2019**, *8*, 11722–11731. [CrossRef]
36. Bertino, E.; Takahashi, K. *Identity Management: Concepts, Technologies, and Systems*; Artech House: London, UK, 2010.
37. Dick, H. *The OAuth 2.0 Authorization Framework*; Rfc 6749, IETF: Fremont, CA, USA, 2012
38. Sandhu, R.S.; Samarati, P. Access control: Principle and practice. *IEEE Commun. Mag.* **1994**, *32*, 40–48. [CrossRef]
39. Ferraiolo, D.; Kuhn, D.R.; Chandramouli, R. *Role-Based Access Control*; Artech House: London, UK, 2003.
40. Damgård, I.; Haagh, H.; Orlandi, C. Access control encryption: Enforcing information flow with cryptography. In *Theory of Cryptography Conference*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 547–576.
41. Hu, V.C.; Ferraiolo, D.; Kuhn, R.; Friedman, A.R.; Lang, A.J.; Cogdell, M.M.; Schnitzer, A.; Sandlin, K.; Miller, R.; Scarfone, K. Guide to attribute based access control (ABAC) definition and considerations (draft). *NIST Spec. Publ.* **2013**, *800*, 1–54.
42. Dennis, J.B.; Horn, E.C.V. Programming semantics for multiprogrammed computations. *Commun. ACM* **1983**, *26*, 29–35. [CrossRef]
43. Crampton, J.; Khambhammettu, H. Delegation in role-based access control. *Int. J. Inf. Secur.* **2008**, *7*, 123–136. [CrossRef]

44. Aiash, M.; Loo, J. A formally verified access control mechanism for information centric networks. In Proceedings of the 12th International Joint Conference on e-Business and Telecommunications (ICETE), Colmar, France, 20–22 July 2015; Volume 4, pp. 377–383.
45. Edris, E.K.K.; Aiash, M.; Loo, J. Formal Verification of Authentication and Service Authorization Protocols in 5G enabled Device-to-Device Communications using ProVerif. *Electronics* **2021**, Accepted for Publication. [CrossRef]
46. 3GPP. System Architecture for the 5G System. In *Technical Specification (TS) 3GPP TS 23.501 V16.7.0 (2020-12)*; Third Generation Partnership Project: Sophia Antipolis, France, 2020
47. 3GPP. 5G System; Technical Realization of Service Based Architecture. In *Technical Specification (TS) 3GPP TS 29.500 V17.1.0 (2020-12)*; Third Generation Partnership Project: Sophia Antipolis, France, 2020
48. Lowe, G. A hierarchy of authentication specifications. In Proceedings of the 10th Computer Security Foundations Workshop, Rockport, MA, USA, 10–12 June 1997; IEEE: Piscataway, NJ, USA, 1997; pp. 31–43. [CrossRef]
49. Menezes, A.J.; Oorschot, P.C.V.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 2018.
50. SECG. SEC 1: Recommended Elliptic Curve Cryptography, 2009. Available online: https://www.secg.org/sec1-v2.pdf (accessed on 26 May 2021)
51. Armando, A.; Carbone, R.; Compagna, L.; Cuellar, J.; Tobarra, L. Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps. In Proceedings of the 6th ACM workshop on Formal Methods in Security Engineering, Alexandria, Virginia, 27–31 October 2008; ACM: New York, NY, USA, 2008; pp. 1–10. [CrossRef]
52. Armando, A.; Basin, D.A.; Boichut, Y.; Chevalier, Y.; Compagna, L.; Cuellar, J.R.; Drielsma, P.H.; Heam, P.C.; Kouchnarenko, O.; Mantovani, J.; et al. The AVISPA tool for the automated validation of Internet security protocols and applications. *Comput. Aided Verif. Proc.* **2005**, *3576*, 281–285.
53. Meier, S.; Schmidt, B.; Cremers, C.; Basin, D. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification*; Sharygina, N., Veith, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8044, pp. 696–701. [CrossRef]
54. Blanchet, B.; Smyth, B.; Cheval, V.; Sylvestre, M. ProVerif 2.01: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial, 2020. Available online: https://opam.ocaml.org/packages/proverif/ (accessed on 2 July 2021).
55. Dolev, D.; Yao, A.C.C. On the Security of Public Key Protocols. *IEEE Trans. Inf. Theory* **1983**, *30*, 198–208. [CrossRef]
56. Ryan, M.D.; Smyth, B. Applied pi calculus. In *Formal Models and Techniques for Analyzing Security Protocols*; IOS Press: Amsterdam, The Netherlands, 2011; Volume 5, pp. 112–142.
57. Bhargavan, K.; Fournet, C.; Gordon, A.D.; Swamy, N. Verified implementations of the information card federated identity-management protocol. In Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, Tokyo, Japan, 18–20 March 2008; pp. 123–135. [CrossRef]