Secure Content Delivery Scheme Based on Yaksha System for CCMANETs

Xian Guo, Tao Feng, Jun-li Fang, JingWang, and Ye Lu
Lanzhou University of Technology, Lanzhou 730050, P. R. China
Email: iamxg@163.com

Abstract—Content-Centric Networking (CCN) is a candidate future Internet architecture that gives favorable promises in MANET environment. It is allowed that data of content producer is cached anywhere in Content Centric MANET (CCMANET). This scheme decoupling of data from the source makes traditional end-end authentication transmission unavailable in CCMANET. To ensure high availability of the cached data only to legitimate users, Secure Content Delivery scheme based on Yaksha (YSCD) for CCMANET is proposed in this paper. Yaksha server manages users and verifies user identification, and distributes “license” only to legitimate user for publishing content to network or attaining content from network. This in this paper, license is the joint signature of the legitimate user and the Yaksha server. In YSCD, it is guaranteed that only the legitimate users can publish to the network and access the content cached on the network. Finally, we prove security properties of YSCD in Protocol Composition Logic (PCL).

Index Terms—CCN/NDN, content delivery, MANET, yaksha

I. INTRODUCTION

According to the recent Cisco Visual Networking Index Global Forecast and Service Adoption for 2013 to 2018 [1]: content delivery networks will carry over half of internet traffic by 2018; Traffic from wireless and mobile devices will exceed traffic from wired devices by 2018. To satisfy these new application requirements such as content delivery and mobile application etc., the research community pays close attention to the discussion of new internet architecture [2]-[7]. In this context, Information –Centric Networking (ICN) has emerged as a promising candidate for the architecture of the future internet. CCN/NDN (Content Centric Networking [8] Named Data Network [9]) is pioneering fully-fledged ICN architecture. The CCN/NDN has rapidly gained consensus, thanks to the simple, robust and effective communication model.

The salient CCN/NDN features, i.e., location-independent named data, in-network caching and lightweight forwarding, make CCN/NDN a particularly attractive solution for wireless ad hoc environments, like Mobile Ad hoc Networks (MANETs) [10]-[12]. However, in CCN/NDN, decoupling of data from the source allows data to be stored anywhere. This in-network caching makes traditional end-end authentication transmission unavailable in CCN/NDN. It is a challenge how to ensure publishing of network data only by legitimate users and high availability of the cached data only to legitimate users [13].

In addition, CCN/NDN changes the security model from securing the path to securing the content, which is available to all ICN nodes. As a consequence, new attacks have appeared with this new security model in addition to the legacy attacks that may have an impact on ICN. Surveys of security for CCN/NDN are investigated in [14]-[16]. Attacks to CCN/NDN are classified into four categories: naming, routing, caching, and other miscellaneous related attacks in [14]. New privacy challenges are studied in [15]. They also think that the interest flooding and the content pollution are two important attacks to CCN/NDN architecture.

So, it is necessary to design a secure content delivery scheme especially in sensitive network environments, such as the tactical and emergency MANETs.

Yaksha system [17] is a variant of public-key cryptosystems RSA [18], in which the RSA private key is split in two portions. One portion becomes a user’s Yaksha private key, and the other the Yaksha server’s private key. The Yaksha security system is a security technology capable of re-using a single security infrastructure to perform various security functions such as authentication, key exchange, digital signature, and key escrow.

In this paper, we propose a Secure Content Delivery scheme based on Yaksha system. It is abbreviated to YSCD in this paper. In our new scheme, joining and leaving of network users is uniformly managed by the Yaksha server. It is particularly important that the Yaksha server is responsible for content publishing and obtaining. When the Yaksha server receives a request from a data source who wants to publish content, firstly, it verifies the source’s identity, if this verification success, it generates a key for encrypting content, and generates a joint signature (between the source and the Yaksha server) that is “license” for the source publishing content, and then it sends these information to requester. Now, the data source can publish content to the network. The Yaksha sever saves this key and the relative information for the

Manuscript received April 10, 2015; revised February 18, 2016.
This work was supported by NSFCNo. 61461027, No. 61462060; Gansu province science and technology plan project under grant No. 14RSZA078, 1308RSZA277; Specialized research fund for the doctoral program of Lanzhou University of technology; We thank the referees for helpful comments.
Corresponding author email: iamxg@163.com.
doi:10.12720/jcm.11.2.221-230

©2016 Journal of Communications

221
other user decrypting the content. Similarly, when the Yaksha server receives a request from a content consumer to obtain some interested content, if it is a legitimate user and there exists the requested content in network, Yaksha server will send a joint signature (between the content consumer and the Yaksha server) and the key decrypted the corresponding content to the requester. Now the user can access the interested content in network, when it obtains the key and the license. Such, it can be guaranteed by YSCD that only the legitimate user can publish content to network and access content from network. In addition, our scheme also can efficiently prevent the interest flooding attack and the content pollution attack because of using the joint signature.

Formal security analysis based on logic is an important approach to verify security properties of a network and cryptographic protocol. Protocol Composition Logic (PCL) [19]-[22] is a formal logic for stating and proving security properties of network protocols. The logic was originally proposed in 2001 and has evolved over time through case studies of widely-used protocol such as SSL/TLS, IEEE 802.11i, and variants of Kerberos. In [23], based on observational equivalence theory, we extend PCL to be Anonymity PCL (APCL) to satisfy the special needs of anonymous analysis. In this paper, we model and demonstrate secrecy and authentication properties of YSCD in PCL.

The rest of this paper is organized as follows. Some background on YSCD is given in Section II, followed by YSCD protocol in Section III. Security analysis of YSCD is discussed in Section IV. And conclusion is in Section V.

II. BACKGROUND
A. Overview of CCN/NDN
Information retrieval in CCN/NDN is driven by the consumer, which uses interest packets to request content “by name”. Intermediate nodes in network select their outgoing network interface(s) for interest forwarding. Finally, upon receiving an interest, a provider, which is the content source or any other network node that temporarily stores the requested content, replies with a named data packet that piggybacks authentication and data-integrity information. The content packet is a self-identify and self-authenticating unit, so in-network data caching is enabled. CCN/NDN packet types are illustrated in Fig. 1. The data packet follows the “bread crumbs” left by the interest at intermediate nodes back to the consumer. The process of forwarding the interest packet and the content packet is shown in Fig. 2. CCN/NDN node is composed of three main components, namely the Pending Interest Table (PIT), and the Content Store (CS). The PIT table stores domains (name prefixes) and their corresponding outgoing face. The PIT table keeps track of outstanding interests; when an intermediate router forwards an interest packet through one of its outgoing faces, a corresponding “pending interest” entry is created in the PIT. The last component, CS, is essentially a content repository. For a more complete discussion of CCN/NDN, please see [8], [9].

![Fig. 1. CCN/NDN packet types [8], [9]](image)

B. Overview of Yaksha System
The Yaksha system is a variant of the RSA public-key cryptosystem. The system works as follows: as in the RSA system, user Alice has her public key pair \((e_A, n_A)\). Unlike the traditional RSA system, however, the Yaksha system uses two distinct private keys—Alice’s private Key, denoted by \(d_{A_A}\), and the Yaksha server’s corresponding key for Alice, denoted by \(d_{A_Y}\). These two new private keys are related to the original RSA private key \(d_A\) by the mathematical relation \(d_{A_A} \times d_{A_Y} = d_A \mod n_A\).

In the Yaksha system, each user \(X\) has his or her own private key \(d_{XX}\), and the Yaksha server maintains a corresponding server private key \(d_{AY}\). The system can be used to perform several security functions such as joint digital signatures, key exchange, key escrow etc.

The joint digital signature is an important technology provided by Yaksha system. Now, we show how the user can interact with the server to sign a message \(M\).

1. Alice calculates \(s = M^{d_{AX}} \mod n_A\) and sends \(s_1\) to the Yaksha server.
2. The Yaksha server uses \(d_{AY}\) to calculates the joint signature

\[
s = s_1^{d_{AY}} \mod n_A = (M^{d_{AX}} \mod n_A)^{d_{AY}} \mod n_A = M^{d_{AX} \times d_{AY}} \mod n_A
\]
Now $s$ is Alice’s signature on message $M$ and is indistinguishable from a regular RSA signature. The other user can verify the signature using Alice’s RSA public key $e_A$.

C. Protocol Composition Logic (PCL)

1) Syntax of the PCL

Protocol Composition Logic (PCL) is developed in [19]-[22]. A simple protocol programming language is used to represent a protocol by a set of roles, such as "initiator" or "responder," each specifying a sequence of actions to be executed by an honest participant. Protocol actions include nonce generation, encryption, decryption and communication steps (sending and receiving). Every principal can be executing one or more copies of each role at the same time. PCL use the word thread to refer to a principal executing one particular instance of a role.

Each thread $X$ is a pair $(\hat{x}, n)$ where $\hat{x}$ is a principal and $n$ is a unique session id. A run is a record of all actions executed by honest principals and the attacker during protocol execution. The set of runs of a protocol is determined by the operational semantics of the protocol programming language.

Table I summarizes the syntax of the logic used in this paper. For every protocol action, there is a corresponding action predicate which asserts that the action has occurred in the run. Action predicates are useful for capturing authentication properties of protocols since they can be used to assert which principals sent and received certain messages. For example, Send$(X, t)$ means that the thread $X$ has sent the term $t$, while New$(X, n)$ means $X$ generates fresh nonce $n$. Honest$(\hat{x})$ means that $\hat{x}$ is acting honestly, i.e., the actions of every thread of $\hat{x}$ precisely follows some role of the protocol. Interpretations of other roles can be found in [19]-[22].

| Table I: Type sizes for camera-ready papers |

<table>
<thead>
<tr>
<th>Action formulas</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a ::= \text{Send}(X, t) \mid \text{Receive}(X, t) \mid \text{New}(X, t)$</td>
<td></td>
</tr>
<tr>
<td>$\text{Sign}(X, t, k) \mid \text{Verify}(X, t, k)$</td>
<td></td>
</tr>
<tr>
<td>$\text{Enc}(X, t, k) \mid \text{PkEnc}(X, t, k)$</td>
<td></td>
</tr>
<tr>
<td>$\text{SymDec}(X, t, k) \mid \text{PkDec}(X, t, k)$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Formulas</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi ::= a \mid \text{Has}(X, t) \mid \text{Honest}(\hat{x}) \mid \text{SafeMsg}(M, s, \chi) \mid \text{SendsSafeMsg}(X, s, \chi)$</td>
<td></td>
</tr>
<tr>
<td>$\phi \wedge \psi \mid \text{SafeNet}(s, \chi)$</td>
<td></td>
</tr>
</tbody>
</table>

2) Proof system of the PCL

Protocol proofs usually use modula formulas of the form $\psi[P]_A \phi$ in PCL. The modal formulas informally means that if $X$ starts from a state in which $\phi$ holds, and executes the program $P$, then in the resulting state the security property $\phi$ is guaranteed to hold irrespective of the actions of an attacker and other honest principals.

The proof system extends first-order logic with axioms and proof rules for protocol actions, temporal reasoning, knowledge, and a specialized form of invariance rule called the honesty rule. The honesty rule is essential for combining facts about one role with inferred actions of other roles. Intuitively, if Alice receives a response from a message sent to Bob, the honesty rule captures Alice’s ability to use properties of Bob’s role to reason about how Bob generated his reply. In short, if Alice assumes that Bob is honest, she may use Bob’s role to reason from this assumption. Table II and Table III respectively list PCL’s axioms and rules used in security proof.

<table>
<thead>
<tr>
<th>Table II: Axioms of PCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{ENC0} \quad \text{Enc}(X, t, K) \implies \text{Has}(X, t) \land \text{Has}(X, t)$</td>
</tr>
<tr>
<td>$\text{ENC1} \quad \text{SimDec}(X, SYMENC_k([t]), K) \implies \exists y. \text{SymEnc}(Y, t, K)$</td>
</tr>
<tr>
<td>$\text{PENC1} \quad \text{PkDec}(X, PKEnc_k([t]), K^-) \implies \exists y. \text{PkeEnc}(Y, t, K)$</td>
</tr>
<tr>
<td>$\text{VER} \quad \text{Honest}(\hat{x}) \land \text{Verify}(Y, \text{SIG}<em>\hat{x}([x])) \implies \hat{x} \neq Y \implies \exists x, t, \text{Send}(X, t) \land \text{Contains}(t, \text{SIG}</em>\hat{x}([x]))$</td>
</tr>
<tr>
<td>$\text{SAF0} \quad \neg \text{SafeMsg}(s, X, K) \land \text{SafeMsg}(x, K)$</td>
</tr>
<tr>
<td>$\text{SAF1} \quad \text{SafeMsg}(M, s, X) = \text{SafeMsg}(M, s, X) \land \text{SafeMsg}(M, s, X)$</td>
</tr>
<tr>
<td>$\text{SAF2} \quad \text{SafeMsg}(\text{SYMENC}_k([M], s, \chi) = \text{SafeMsg}(M, s, \chi) \lor K \in \chi)$</td>
</tr>
<tr>
<td>$\text{SAF3} \quad \text{SafeMsg}(\text{PKENC}_k([M], s, \chi) = \text{SafeMsg}(M, s, \chi) \lor K^{-1} \in \chi)$</td>
</tr>
<tr>
<td>$\text{SAF4} \quad \text{SafeMsg}(\text{HASH}([M], s, \chi))$</td>
</tr>
<tr>
<td>$\text{NET0} \quad \text{SafeNet}(s, \chi)[s] \implies \text{SendSafeMsg}(X, s, \chi)$</td>
</tr>
<tr>
<td>$\text{NET1} \quad \text{SafeNet}(s, \chi)[r] \implies \text{SendSafeMsg}(X, s, \chi)$</td>
</tr>
<tr>
<td>$\text{NET2} \quad \text{SendSafeMsg}(X, s, \chi)[a] \implies \text{SendSafeMsg}(X, s, \chi)$</td>
</tr>
<tr>
<td>$\text{NET3} \quad \text{SendSafeMsg}(X, s, \chi) \implies \text{SendSafeMsg}(X, s, \chi)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table III: Rules of PCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{HON} \quad X \mid T \varphi \quad \forall \rho \in Q \forall P \in \text{BS}(\rho) \rho[P]_A \varphi$</td>
</tr>
<tr>
<td>$Q \supset \text{Honest}(\hat{x}) \supset \varphi$</td>
</tr>
<tr>
<td>$\text{NET} \quad \forall \rho \in Q \forall P \in \text{BS}(\rho)$</td>
</tr>
<tr>
<td>$\text{SDF} \quad \text{SafeNet}(s, \chi)[s] \text{Honest}(X) \land \varphi \supset \text{SendSafeMsg}(X, s, \chi)$</td>
</tr>
<tr>
<td>$Q \supset \text{KOHonest}(s, \chi) \land \varphi \supset \text{SafeNet}(s, \chi)$</td>
</tr>
<tr>
<td>$\text{POS} \quad \text{SafeNet}(s, \chi) \land \text{Has}(X, M) \land \neg \text{SafeMsg}(M, s, \chi) \supset \exists k \in \chi. \text{Has}(X, k) \lor \text{New}(X, s)$</td>
</tr>
<tr>
<td>$Q \supset \text{Honest}(\hat{x}) \land \varphi \land \text{SafeNet}(s, \chi)[S] \text{SendsSafeMsg}(X, s, \chi)$</td>
</tr>
<tr>
<td>$\text{POS} \quad \text{SendsSafeMsg}(X, s, \chi) \land \text{Has}(Y, M) \land \neg \text{SafeMsg}(X, s, \chi)$</td>
</tr>
<tr>
<td>$Q \supset \text{Has}(Y, M) \land \neg \text{SafeMsg}(X, s, \chi)$</td>
</tr>
</tbody>
</table>

3) Compositional proof of the PCL

Following the modular design of the protocol, the compositional approach is developed in [19-22]. This method can prove properties of the whole protocol by combining proofs of its parts. In [19-22], the authors proposed three kinds of composition operation on protocols-parallel, sequential, and staged. One central concept in composition proof methods is the notion of an
“invariant”. An invariant for a protocol is a logic formula that characterizes the environment in which it retains its security properties. In this paper, we use staged composition in secrecy analysis. The detail of staged composition is described as follows.

**Definition 1 (Staged Composition).** A protocol \( Q \) is a staged composition of protocols \( Q_1, Q_2, \ldots, Q_n \) if each role of \( Q \) is of the form \( RComp(<R_1, R_2, \ldots, R_n>) \), where \( R_i \) is a role of protocol \( Q_i \).

**Theorem 1 (Staged Composition).** If \( Q \) is a staged composition of protocols \( Q_1, Q_2, \ldots, Q_n \), then we can conclude \( Q \) is a stage composition of protocols \( Q_1, Q_2, \ldots, Q_n \).

II. **YSCD Protocol**

A. **System Model**

1) **Network Model**

To illustrate our new scheme, simplified CCMANET network model is show in Fig. 3. Node \( S \) is content producer, namely data source. It can publish content to the network. Node \( C \) is a user who is interested in some content cached in network, so, we refer to it as a content consumer. The content consumer may subscribe the content cached on the network by broadcasting the interest packet. Node \( Y \) that can be connect with any node is the Yaksha server. The Yaksha server manages all network user information and the relative information of the content on the network. Each user, who wants to join the network, need to register to the Yaksha server. Of course, it is deleted from the Yaksha server when a user leaves the network. At the same time, the Yaksha server monitors the publishing and the subscribing of the content. \( N \) is the network consists of other nodes except that we consider specific nodes such as \( C, S \), and \( Y \). Here, \( N \) models the network following requirements of forwarding interest packet and content packet in CCN/NDN, and also can cache data.

![Fig. 3. Simplified CCMANET network model](image)

2) **Threaten model**

In our threaten model, we consider the consumer \( C \), the data source \( S \) and the Yaksha server are all honest. That is to say, all of them will behave honestly to follow the proposed scheme. We also assume that the network \( N \) will forward the interested packet and the content packet according to our proposed protocol rules. In this paper, we mainly consider the external adversaries. The external adversaries are assumed in our system to eavesdrop the message transmitted on the network. What is more, they could also launch some active attacks, for instance, tampering the message or replaying the old message.

3) **Security requirements**

To prevent attacks from the aforementioned adversaries and ensure the security of content delivery, we require that the proposed scheme should satisfy the security requirements given as follows.

**Data confidentiality:** It requires that only the legitimate network user can access the content on the network, and any other one should be prevented from obtaining the relevant the encrypt key on the content.

**Data authentication:** It indicates the authentication for the user identity and message integrity. The former means that only the legitimate network user can broadcast the interest packet and publish the content packet. While the latter guarantees any altered data during transmission could be detected.

B. **YSCD Protocol**

Our new scheme consists of five stages: Yaksha key generations, the content packet generation, the interest packet generation, packet forwarding and content fetch. We detailley illustrate how they work as follows.

1) **Yaksha key generations**

In the user registration stage, the Yaksha server and user collaboratively compute the long term Yaksha system private key. For example, when the user \( X \) joins the network, first the regular RSA parameters \( p_X, q_X, n_X, \varphi(n_X), e_X, d_X \) are generated. Now it is necessary to generate two values and such that:

\[
d_{XX} \times d_{XY} = d_X \mod \varphi(n_X)
\]

\( D_{XY} \) which will be stored on the Yaksha server is always a regular length RSA private key. \( D_{XY} \) can also be long if the user has a smart card. In this case the system can generate both parameters. If the user has to memorize the password, and if we want to let the user choose this password, then we can let the user pick a character password, which becomes \( d_{XX} \). Another option is to generate a short key (password) for the user. In all cases, after picking one of \( d_{XY} \) or \( d_{XX} \) we solve for the other. Of course, any equation of the form:

\[
aX = b \mod \varphi(n)
\]

is solvable for \( x \) only if \( \gcd(\varphi(n), b) \) divides \( b \), and a few retries may be necessary (in the case where a user picks their password, it may be more user-friendly to re-
generate the RSA keys itself, instead of having the user retry a different password).

After key generation, \( d_{XX} \) is given to the user \( X \), \( d_{CY} \) to the Yaksha server, \( (e_X, n_X) \) is used to generate the associated certificate available on the network. Here, the Yaksha server also plays certification authority role and manages user’s certification, when a user register to network. In addition, we assume that all of users in network use same symmetric cryptography algorithm such as AES or DES.

In our scheme, the process that the Yaksha server verifies the identity of a user \( X \) is as follows:

1) The user \( X \) sends the following message to the Yaksha server:
\[
[ID_X \ || \ m^{e_X} \ \text{mod}\ \phi(n_X)]
\]
where \( m^{e_X} \ \text{mod}\ \phi(n_X) \) is the user \( X \)’s signature on message \( m \).

2) When the Yaksha server receives the message \( [ID_X \ || \ m^{e_X} \ \text{mod}\ \phi(n_X)] \) from a user \( X \), it firstly computes the joint signature:
\[
m^{d_{XX} \cdot ed_{XX}} \ \text{mod}\ \phi(n_X)
\]
and then the Yaksha server uses the public key of the user \( X \) to decrypt \( m^{d_{XX} \cdot ed_{XX}} \ \text{mod}\ \phi(n_X) \). The process of decrypting is:
\[
(m^{d_{XX} \cdot ed_{XX}} \ \text{mod}\ \phi(n_X))^{e_X} \ \text{mod}\ \phi(n_X) \\
= m^{d_{XX} \cdot ed_{XX} \cdot d_{XX}^{-1}} \ \text{mod}\ \phi(n_X) \\
= m^{d_{XX}} \ \text{mod}\ \phi(n_X) \\
= m
\]
Note that the process has two by-products. The Yaksha server and only the Yaksha server can recover the message \( m \) from \( m^{d_{XX}} \ \text{mod}\ \phi(n_X) \), because of \( d_{XX} \) is only possessed by the Yaksha server. In addition, the joint signature of the user \( X \) and the Yaksha server is generated.

2) The Content Packet Generation
We assume that the data source \( S \) produces the content \( C_a \) and the CCN/NDN name of this content \( C_a \) is \( a \), the process of publishing this content is as follows:

1) The data source \( S \) firstly requests the key \( k_a \) that used to encrypt the content \( C_a \) by sending the following message to the Yaksha server.
\[
[ID_S \ || \ ID_a \ || \ n_{SY} \ || \ a] \\
(H(C_a)^{e_{SY} \cdot d_{SY}} \ \text{mod}\ \phi(n_{SY}))^{d_{SY} \cdot d_{SY}^{-1}} \ \text{mod}\ \phi(n_{SY})
\]
where \( n_{SY} \) in the above message is a nonce, and the subscript \( SY \) indicates that the source \( S \) sends the nonce to the Yaksha server.

2) The Yaksha server firstly verifies the identity of \( S \), if this verification is fail, it deletes this message. Otherwise, the Yaksha server generates the key \( k_a \) and records these information such as the identity of \( S \), the content name \( a \) and the key \( k_a \).

3) The Yaksha server sends the following message to the data source \( S \). The message contains the key \( k_a \) (used to encrypt the content \( C_a \)) and the joint signature of the source \( S \) and the Yaksha server.
\[
[ID_S \ || \ ID_a \ || \ symenc(n_{SY}) \ || \\
H(C_a)^{e_{SY} \cdot d_{SY} \cdot d_{SY}^{-1}} \ \text{mod}\ \phi(n_{SY})) \ || \ k_a^{d_{SY} \cdot d_{SY}^{-1}} \ \text{mod}\ \phi(n_{SY})]
\]

4) When the source \( S \) receives the message in (2), it firstly recovers the key \( k_a \) by calculating \((k_a^{d_{SY} \cdot d_{SY}^{-1}} \ \text{mod}\ \phi(n_{SY}))^{d_{SY}} \ \text{mod}\ \phi(n_{SY}) \), and then, it decrypts and verifies the joint signature. If the signature verification successes, it publishes the following content to the network \( N \) and deletes this key \( k_a \).
\[
[ID_C \ || \ a] \\
H(C_a)^{e_{SY} \cdot d_{SY}} \ \text{mod}\ \phi(n_{SY})) \\
symenc(C_a) \ || \ H(C_a)^{e_{SY} \cdot d_{SY}} \ \text{mod}\ \phi(n_{SY}), k_a]
\]

3) Interest packet generation
We assume that a content consumer \( C \) is interested in the content \( C_a \) (the name of this content \( C_a \) is \( a \)), the process that the content consumer \( C \) generates and publishes the interest packet is as follows:

1) The content consumer \( C \) sends the following message to the Yaksha server, here, \( n_{CY} \) is a nonce that \( C \) sends to the Yaksha server.
\[
[ID_C \ || \ ID_a \ || \ n_{CY} \ || (H(a)^{e_{YC} \cdot d_{YC}} \ \text{mod}\ \phi(n_{YC}))]
\]

2) The Yaksha server firstly verifies the identity of \( C \), if this verification is fail, it deletes this message. Otherwise, the Yaksha server generates and sends the following messages to the content consumer \( C \):
\[
[ID_C \ || \ ID_a \ || \ symenc(n_{CY}) \ || \\
H(a)^{e_{YC} \cdot d_{YC} \cdot d_{YC}^{-1}} \ \text{mod}\ \phi(n_{YC})) \ || \ k_a^{d_{YC} \cdot d_{YC}^{-1}} \ \text{mod}\ \phi(n_{YC})]
\]

3) When the content consumer \( C \) receives the message in (4), it saves \( k_a^{d_{YC} \cdot d_{YC}^{-1}} \ \text{mod}\ \phi(n_{YC})) \) (It will be used to decrypt the interested content later.) and broadcasts the following interest packet to the network \( N \).
\[
[ID_a \ || \ a] || (H(a)^{e_{YC} \cdot d_{YC}} \ \text{mod}\ \phi(n_{YC}))
\]

4) Packet forwarding
Nodes in the network \( N \) will process the interest packet and the content packet according to the CCN/NDN components such as FIB table, PIT table and content store CS, the detailed of this process is as follows:

1) When a node \( n \) in the network \( N \) receives an interest packet about the content \( C_a \), it firstly verifies the joint signature of the Yaksha server and the consumer \( C \). If this verification is fail, it deletes the interest packet. Otherwise , it checks it’s CS and does as follows:

- If there exists the content \( C_a \) in node \( n \)’s CS, then the node \( n \) deletes the interest packet and directly sends the following content packet to the node who forwarded this interest packet to the node \( n \).
receives a content name \{Consumer( ), Yaksha( ), Source( )\}. We use \(\hat{C} \hat{Y} \hat{S}\) as names for protocol participant the Consumer, the Yaksha server and the data Source. \(C, Y, S\) respectively represents thread of protocol participant \(\hat{C} \hat{Y} \hat{S}\) .Using protocol programming language of PCL, Roles of YSCD are shown below.

Source = \((S, \hat{Y}, \hat{C}, H(C_y))\)

\(\star\) apply encrypted key for content \(C_n\) \* /
new \(n_{C_y}\);
\(H(C_y) := \text{hash } C_y;\)
\(s_{C_y} := \text{sign } H(C_y),d_{C_y};\)
\(\text{enc}_{C_y} := \text{pkenc } n_{C_y},s_{C_y},e_{C_y};\)
send \(S.Y.a.\text{enc}_{C_y};\)
receive \(\hat{Y}.\hat{S}.a.\text{enc}_{C_y}.\text{enc}_{C_y};\)
text_{C_y} := \text{pkdec } \text{enc}_{C_y},d_{C_y};
machine text_{C_y} as \(k_{C_y};\)
text_{C_y} := \text{symdec } \text{enc}_{C_y},k_{C_y};
machine text_{C_y} as \(n_{C_y},s_{C_y};\)
verify \(s_{C_y},e_{C_y};\)

--- publish content ---
\(\text{enc}_{C_y} := \text{symenc } C_y,s_{C_y},k_{C_y};\)
send \(S.a.\text{enc}_{C_y};\)Source

Consumer = \((C, Y, S, a)\)

\(\star\) apply decrypted key for content \* /
new \(n_{C_y}\);
\(H(a) := \text{hash } a;\)
\(s_{C} := \text{sign } H(a),d_{C};\)
\(\text{enc}_{C} := \text{pkenc } n_{C},s_{C},e_{C};\)
send \(\hat{C}.a.\text{enc}_{C};\)
receive \(\hat{Y}.\hat{C}.a.\text{enc}_{C}.\text{enc}_{C};\)
text_{C} := \text{pkdec } \text{enc}_{C}.d_{C};
machine text_{C} as \(k_{C};\)
text_{C} := \text{symdec } \text{enc}_{C},k_{C};
machine text_{C} as \(n_{C},s_{C},e_{C};\)
verify \(s_{C},e_{C};\)

\(\star\) broadcast interest packet to attach content \* /
send \(\hat{C}.a.s_{C};\)
receive \(\hat{S}.a.\text{enc}_{C};\)
text_{C} := \text{symdec } \text{enc}_{C},k_{C};
machine text_{C} as \(C.a.\text{enc}_{C};\)
verify \(s_{C},e_{C};\)
h_{C} := \text{Hash}(C_y);
machine h_{C} as \(H(C_a);\)\(\star\) send the decrypted key for content \(C_n\) to \(C\) \* /
receive \(\hat{C}.\hat{Y}.\hat{S}.a.\text{enc}_{C};\)
text_{C} := \text{pkdec } \text{enc}_{C}.d_{C};
machine text_{C} as \(n_{C},e_{C};\)
\[ s_{cy} := \text{sign}(s_c, d_{cy}) \]  
\[ \text{verify } (s_{cy}, e_c) \]  
\[ enc_{cy} := \text{symenc}(n_{cy}, s_{cy}, k_a) \]  
\[ enc_{cy} := \text{pkenc}(k_a, e_c) \]  
\[ \text{send } Y.C \cdot enc_{cy}, enc_{cy} \]

* send the encrypted key for content \( C_y \) to \( S \) *

\[
\begin{align*}
\text{receive } S.Y \cdot a \cdot enc_{cy}, e_c; \\
\text{math text for } enc_{cy}, e_c; \\
\text{send text enc d}
\end{align*}
\]

The security objectives of YSCD are of two types: authentication and secrecy. The authentication objectives take the form that a message of a certain format was indeed sent by some thread of the expected principal. The secrecy objectives take the form that a putative secret is known only to certain principals. Using the symbol of PCL, the security properties of YSCD are described in Table V.

<table>
<thead>
<tr>
<th>Table IV: Security Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{AUTH}_{ts} ): ( \exists \eta. \text{Send}(Y, \eta, Y, S) )</td>
</tr>
<tr>
<td>( E_{cy}[k_c</td>
</tr>
<tr>
<td>( \text{AUTH}_{sc} ): ( \exists \eta. \text{Send}(Y, \eta, Y, C) )</td>
</tr>
<tr>
<td>( E_{cy}[k_c</td>
</tr>
<tr>
<td>( \text{AUTH}_{cs} ): ( \exists \eta. \text{Send}(S, \eta, S, a) )</td>
</tr>
<tr>
<td>( E_{cy}[k_c</td>
</tr>
<tr>
<td>( \text{AUTH}<em>{source} ) ( \text{source} \cdot \text{Honest}(S, Y) \Rightarrow \text{AUTH}</em>{ts} )</td>
</tr>
<tr>
<td>( \text{AUTH}<em>{source, consumer} ) ( [\text{consumer}] \cdot \text{Honest}(C, Y) \Rightarrow \text{AUTH}</em>{sc} )</td>
</tr>
<tr>
<td>( \text{SEC}_{cy} ) ( \text{Honest}(C, Y, S) \Rightarrow (\text{Has}(X, k_a) \Rightarrow X \in (\hat{C}, Y, \hat{S})) )</td>
</tr>
<tr>
<td>( \text{SEC}<em>{cy} ) ( \text{[yaksha], SEC}</em>{cy} )</td>
</tr>
<tr>
<td>( \text{SEC}<em>{cy} ) ( \text{[source], SEC}</em>{cy} )</td>
</tr>
<tr>
<td>( \text{SEC}<em>{cy} ) ( \text{[consumer], SEC}</em>{cy} )</td>
</tr>
</tbody>
</table>

The security objectives of YSCD are of two types: authentication and secrecy. The authentication objectives take the form that a message of a certain format was indeed sent by some thread of the expected principal. The secrecy objectives take the form that a putative secret is known only to certain principals. Using the symbol of PCL, the security properties of YSCD are described in Table V.

Where \( \text{AUTH}_{source, consumer} \) states that when the thread \( S \) finishes executing the role Source(), some thread \( Y \) of principal \( Y \) indeed sent the expected message that contains the key \( k_a \). \( \text{AUTH}_{source} \) states that when the thread \( C \) of principal \( C \) finishes executing the role Consumer(), some thread \( S \) of principal \( S \) indeed publishes the content \( C \) that the consumer is interested in. \( \text{SEC}_{cy} \) states that the key \( k_a \) is secret after execution of the consumer role by \( C \); the other security properties are analogous.

### Environment Assumptions

We assume that the Yaksha private key of protocol participant only is known by owner in our scheme YSCD, namely they don’t leave to any other participant. Using the symbol of PCL, this assumption is described as follows:

\[
\Gamma : \forall X, Z. \text{Honest}(X) \land (\text{Has}(Z, d_{cy}) \land \lnot \text{Has}(Z, d_{cy})) \\
\land (\lnot \text{Has}(Z, d_{cy}) \lor \lnot \text{Has}(Z, d_{cy})) \Rightarrow (\hat{Z} = \hat{X})
\]

### Security Proof

**Theorem 2 (The Key \( k_a \) Authentication)** On execution of the Source and the Consumer roles by the relative principal \( Y \), it is guaranteed that the intended Yaksha server indeed sent expected response under the assumption Honest(\( C, Y, \hat{S} \)). Formally,

\[
\text{YSCD} \Rightarrow \text{AUTH}_{source, Yaksha} \cdot \text{AUTH}_{source, Yaksha}
\]

**Proof:** we firstly prove that \( \text{YSCD} \Rightarrow \text{AUTH}_{source, Yaksha} \). We assume that the thread \( S \) of the protocol participant \( \hat{S} \) completely performs the role Source(). We make induction proof on basic sequence \( S \) of Source():

\[
\text{source} = \{ \text{receive } Y.S.a \cdot enc\_y, enc\_y; \\
\text{text enc d}; \\
\text{match text for } enc\_y, enc\_y \}
\]

The fact that the Source can decrypt \( enc\_y = k_a \cdot d_{cy} \mod \phi(n_c) \) using the user partition \( d_{cy} \) of the Yaksha private key, indicates that there exists the thread \( X \) of the protocol participant \( \hat{X} \) generates and sends the message contains the ciphertext \( enc\_y \) encrypted with the Yaksha private key \( d_{cy} \) according to the axiom \( \text{PENC1} \). On the environment assumption \( \Gamma \), the assumption Honest(\( C, Y, \hat{S} \)) and the honest rule HON, however, the protocol participant \( \hat{X} \) possessed the Yaksha server key \( d_{cy} \) must be the Yaksha server. Such we prove that the equation as below:

\[
\text{YSCD} \Rightarrow \text{AUTH}_{source, Yaksha}
\]

Similar to the above proof process, we can conclude that

\[
\text{YSCD} \Rightarrow \text{AUTH}_{source, Yaksha}
\]

**Theorem 3 (The Key \( k_a \) Secrecy)** On execution of the Consumer and the Source roles by the relative principal, secrecy of the Key \( k_a \) is preserved under the assumption Honest(\( C, Y, \hat{S} \)). Formally,

\[
\text{YSCD} \Rightarrow \text{SEC}_{cy} \cdot \text{SEC}_{cy} \cdot \text{SEC}_{cy}
\]

**Proof:** under the protection of the set \( K \) of the Yaksha private keys \( K = \{ d_{cy}, d_{cy}, d_{cy}, d_{cy}, d_{cy} \} \), the secrecy property of
key \( k_a \) can be proved. We firstly prove that the execution of the Consumer, the Source or the Yaksha server roles discharges the invariant assumptions \( \Phi \), that is:

\[
\begin{align*}
[\text{Consumer}()]_k & \quad \text{Honest}(C, \hat{S}, \hat{Y}) \Rightarrow \Phi \\
[\text{Source}()]_k & \quad \text{Honest}(C, \hat{S}, \hat{Y}) \Rightarrow \Phi \\
[\text{Yaksha}()]_k & \quad \text{Honest}(C, \hat{S}, \hat{Y}) \Rightarrow \Phi
\end{align*}
\]

The invariant \( \Phi \) of YSCD protocol defines the operations that what the thread \( Y \) generated the key \( k_a \) can do and can’t. The assumed condition \( \Phi \) is the conjunction of the following formulas \( \Phi_i \) (i=1, 2).

\[
\begin{align*}
\Phi_1 : \forall X, m, \text{New}(X, k_a) \Rightarrow \\
\quad \neg(\text{Send}(X, m) \land \text{ContainsOpen}(m, k_a))
\end{align*}
\]

\[
\begin{align*}
\Phi_2 : \forall \mathcal{Y}, \hat{C}_a, S_\mathcal{Y}, \hat{Y}, m, \text{New}(X, k_a) \land \text{pkenc}(X, k_a, d_{\hat{S}_\mathcal{Y}}, e_{k_\mathcal{Y}}) \\
\quad \land \text{pkenc}(X, k_a, d_{\hat{Y}}, e_{k_\mathcal{Y}}) \Rightarrow X = Y = S_\mathcal{Y} = \hat{S}_\mathcal{Y} = C
\end{align*}
\]

The predicate Contains Open \( (m, k_a) \) in \( \Phi_1 \) asserts that \( k_a \) can be obtained from \( m \) by a series of un-pairings only-no decryption required. In addition, assumes that some thread \( Y \) of \( \hat{Y} \) generated the key \( k_a \). Therefore, we can conclude that:

\[
\text{KOHonest}(k_a, \mathcal{Y}) = \text{Honest}(\hat{Y}, C, \hat{S})
\]

Clearly, \( \Phi \) is prefix closed and the set \( \mathcal{Y} \) of the Yaksha private keys only can be accessed by the protocol participants \( C, \hat{S}, \hat{Y} \). These proofs of the predicate \( \Phi_i \), (i=1, 2) are derived from the authentication properties

\[
\begin{align*}
\text{AUTH}_{\text{Yaksha}}^{\text{Consumer}} \land \text{AUTH}_{\text{Table}}^{\text{Source}}
\end{align*}
\]

Secondly, we show the proof of the following expression:

\[
\text{YSCD} \Rightarrow \text{Honest}(\hat{C}, \hat{S}, \hat{Y}) \land \Phi \Rightarrow \text{SafeNet}(k_a, \mathcal{Y})
\]

The above formula indicates that the message sent/received on the network and protected by the key set \( \mathcal{Y} \) is safe if the assumptions of \( \text{Honest}(\hat{C}, \hat{S}, \hat{Y}) \) and \( \Phi \) is satisfied. The proof of this part uses the staged composition theorem (Theorem 6 in [14]).

Finally, the following conclusion can be inferred by combining the above two proof steps and using the rules NET, POS and POSL:

\[
\text{YSCD} \Rightarrow \text{SEC}^{\text{Consumer}}_{k_a} \land \text{SEC}^{\text{Source}}_{k_a} \land \text{SEC}^{\text{Table}}_{k_a}
\]

**Theorem 4 (Source Authentication and Integrity Authentication for a Content \( C_a \))** On execution of the Consumer role by a principal is guaranteed that the intended Source indeed sent the expected content \( C_a \) under the assumption \( \text{Honest}^{\hat{C}, \hat{S}, \hat{Y}} \). Formally,

\[
\text{YSCD} \Rightarrow \text{AUTH}^{\text{Source}}_{\text{Consumer}}(C_a)
\]

**Proof:** We assume that the thread \( C \) of protocol participant \( \hat{C} \) completely performs the role Consumer(),

\[
\text{consumer}_2 = \{ \text{receive} \hat{S}.a.enc_{g_2} \}
\]

\[
\text{text}_{c_3} = \text{symdec} enc_{g_3}, k_a;
\]

\[
\text{math} text_{c_3} \Rightarrow C_s, e_{c_3};
\]

\[
\text{verify} e_{c_3}, e_{c_2};
\]

\[
\text{hash} h_{c_3} = \text{Hash}(C_c);
\]

\[
\text{math} h_{c_3} \Rightarrow H(C_c);
\]

The fact, that the consumer can verify the joint signature \( H(C_a)^{d_{\hat{S}_\mathcal{Y}} \mod(p_{\mathcal{Y}})} \) using the source \( S \)’s public key \( e_s \) indicates that there exists the thread \( X \) of the protocol participant \( \hat{X} \) generates this joint signature in collaboration with the Yaksha server according the PCL axiom VER. On the environment assumption \( \Gamma \) and the assumption \( \text{Honest}^{\hat{C}, \hat{S}, \hat{Y}} \), however, the protocol participant \( \hat{X} \) possessed the Yaksha server key \( d_{\hat{S}_\mathcal{Y}} \) must be the data source \( S \). Such we prove that the equation as below:

\[
\text{YSCD} \Rightarrow \text{AUTH}^{\text{Source}}_{\text{Consumer}}(C_a)
\]

Finally, the confidentiality of the content \( C_a \) can be guaranteed by theorem 1-3. In addition, Using of the joint signature can efficiently defend the interest flooding attack and the content pollution attack.

V. CONCLUSION

CCN/NDN is a competitive architecture for future internet. In this paper, based on Yaksha system we proposed a secure content delivery scheme YSCD for Content Centric MANET. We prove security properties of YSCD in PCL. In YSCD, it is guaranteed that only legitimate users can publish and access content on network. In addition, our scheme can efficiently defend two important attacks type such as the interest flooding and the content pollution because we use the joint signature in YSCD. The implement of YSCD is our future work.

ACKNOWLEDGMENTS

This work is supported by NSFC No. 61461027, No. 61462060; Gansu province science and technology plan project under grant No. 145RJZA078, 1308RJZA277; Specialized research fund for the doctoral program of Lanzhou University of technology; We thank the referees for helpful comments.

REFERENCES


**Wang Jing** received his Master's degree in Signal and Information Processing from Lanzhou University of Technology, Lanzhou, China in 2007. He is a lecturer in the School of Computer and Communication, Lanzhou University of Technology, China. His research interests include network and information security.

**Ye Lu** received his Master's degree in Control theory and control engineering from Lanzhou University of Technology, Lanzhou, China in 2014. He is now a PhD candidate in Lanzhou University of Technology, China. His research interests include industrial network and information security.