KEY MANAGEMENT IN INFORMATION CENTRIC NETWORKING

Bander A Alzahrani, Vassilios G. Vassilakis and Martin J. Reed

School of Computer Science and Electronic Engineering, University of Essex, UK

ABSTRACT

Information centric networking (ICN) has been in the spotlight of recent research. It is an emerging communication paradigm that relays on the concept of publish and subscribe. It aims to revise the current Internet with a new clean slate architecture where the design is completely different from today's location based model. To secure the forwarding plan in this network, it is vital to have a time based transient forwarding identifiers by periodically changing the network link identifiers. This assumes shared keys to be distributed prior the communications between an entity termed topology manager (TM) and each forwarder in the network. Exchanging and sharing a secret key between two parties is one of most critical functions in cryptography that needs to be more concerned when integrating cryptographic functions into the system. As ICN is brand new Internet architecture, many existing cryptography protocols may need to be redesigned to fit this new architecture. Therefore, this paper focuses on the security aspect of ICN and proposes an initial design to deploy the integrated Diffie-Hellman-DSA key exchange protocol as a key distributions mechanism.

KEYWORDS

Bloom filter; zFilter; zFormation; Diffie-Hellman-DSA; Information-Centric Networking

1. INTRODUCTION

Information-centric networking (ICN) is an emerging communication paradigm that is receiving an increasing attention by the research community. This is because of the concept that it relays on, where information items are of primary importance rather than the location of end users. The motivation of this clean-slate architecture is due to the many limitations on the traditional IP-based Internet model including security, mobility, scalability, economic [1]. These issues are related to the original design of the Internet in 1960s, where the Internet traffic speed was very slow with a limited number of trusted users.

Security is one of the major concerns and challenges in the current Internet. Current statistics reported by the United States Government Accountability Office shows that there were on total 100,000 security incidents reported in the year 2011[2]. Distributed denial-of-service (DDoS) attacks are observed daily with an average of 7000 attempts. The security issue of Internet has forced many businesses to invest in their infrastructure by adopting various security solutions, e.g. firewalls, intrusion detection systems, antivirus software. One of the main reasons behind the success of these security breaches is the current location based model of the Internet. In this model sender is in complete control of sending any bad and arbitrary data, such as unsolicited messages (spam). The network makes best effort to deliver this data. However, in ICN users only receive data that has been previously requested by them. Therefore, to overcome the aforementioned network limitations, several solutions have been proposed as architectures for

DOI : 10.5121/ijcnc.2013.5609

ICN including content-centric networking (CCN) [3], data-oriented network architecture (DONA) [4], COMET [5] and PSIRP/PURSUIT [6].

ICN architectures rely on the concept of publish/subscribe paradigm as alternative way to the commonly used send/receive paradigm. The basic element of this architecture is information; information is everything and everything is information [5].

Our work is based on the PSIRP/PURSUIT ICN architecture. This architecture is composed of three elements: publisher, subscriber, and network broker [6]. A simplified view of the communication model is shown in Fig 1. Publisher is the information provider where he advertises information by issuing publication messages and serving the requested publications to the forwarding layer to be delivered to a subscriber. Subscriber is information consumer where he subscribes to the desired information item by issuing subscription requests. The network broker is divided into Rendezvous (RV) [7], Topology Manager (TM) [8] and Forwarder (FW). Rendezvous works as middle-man between publisher and subscriber. It is responsible for linking subscriptions with the requested publications. The role of TM is to collect and manage the information about network states by monitoring the network and detecting any changes. Also it is responsible for finding suitable routes to subscribers by creating forwarding identifiers (FID). When information is published it is uniquely identified by assigning a pair of identifiers, Rendezvous identifier (RID) and Scope identifier (SID). Each subscriber must know this pair in order to request a publication. Subscribers send the subscription request to the Rendezvous to find the publication that matches the subscription. Once the publisher of the requested information item is identified then the rendezvous instructs the topology manager to construct the delivery tree from the publisher to the subscriber.

The routing system adopted in PSIRP/PURSUIT ICN is the line speed publish/subscribe inter-networking (LISPIN) [9], a multicast forwarding fabric based on Bloom-filter. As this mechanism has some security issues discussed later, the zFormation mechanism [10] has been proposed to secure the forwarding plane of ICN by offering a DoS resistant forwarding service. The zFormation mechanism requires the TM to have a master key shared with each FW in the network. Therefore in this paper we extend our previous work proposed in [11] by redesigning the integrated Diffie-Hellman-DSA key exchange protocol to fit securely on the ICN architecture. This enables each newly bootstrapped forwarding node to create and share a key with TM to be used be for the subsequent master key.

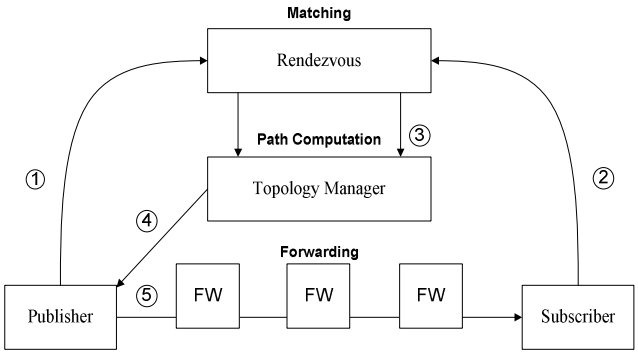

Figure. 1. Basic PSIRP/PURSUIT communication model

The rest of this paper is organized as follows. Section 2 briefly outlines the basic forwarding mechanism used in PISRP/PURSUIT, the issue of false positives and the forwarding vulnerabilities with an existing solution. In Section 3, we describe our recent work and analyse the two proposed security designs. In Section 4, we explain the Diffie-Hellman key exchange, the

digital signature algorithms and DH-DSA protocol. We proceed in section 5 by introducing our initial design of deploying this protocol under ICN. Finally we draw the conclusion in section 6.

2. BACKGROUND

2.1. Bloom-filter Based Forwarding Mechanism

Bloom filter can be described as probabilistic data structure function used to examine if an element is present in a set, full description can be found in [12]. Due to the memory efficiency that Bloom filter offers, it has gained recently wide interest in network applications. It has been proposed in the packets forwarding approach LIPSIN [9] to offer an efficient source routing by aggregating the whole delivery tree identifiers. Unlike the current Internet routing system where each node is named with a unique IP address, in LIPSIN a link identifier (LID) is assigned to each unidirectional link between nodes. Therefore, for a point-to-point connection there will be two LIDs, one in each direction. Each LID is an m-bit Bloom filter, with k bits set to 1(i.e. k is the number of hash functions applied to set bit position to 1). Typically (k<<m) and m is relatively large. This makes the LIDs statistically unique and gives up to $m!/(m-k)! \approx 1012$ LIDs for m=265 k=5 [9].

The communication takes place by the publisher host when publishing information item to RV by assigning RID and SID. Then if a subscriber is interested in this information item, he subscribers to the information by sending subscription request to RV. When RV receives a publication request to publish information items, it records the received publication advertisements in a table similar to Table 1. After that, items are considered published. Similar records are kept for subscription requests. The RV records them as shown in Table 2. If publication-subscription matching occurs, the RV instructs the TM to construct a delivery path from the publisher to the subscriber and create a FID representing this delivery path. For example, based on Tables 1 and 2, there will be the following matching: Pub-2 – Sub-1: for (00000003, 00000001).

When the TM receives a request to construct the path and to create the FID, it uses its topology knowledge to create conceptual map between the two parties to determine the best suitable path.

Table 1. Publication Info at the RV

Pub ID	RID	SID
Pub-1	00000001	00000001
Pub-1	00000002	00000001
Pub-2	00000003	00000001

Table 2. Subscription Info at the RV

Sub ID	RID	SID
Sub-1	00000003	00000001
Sub-2	00000004	00000001

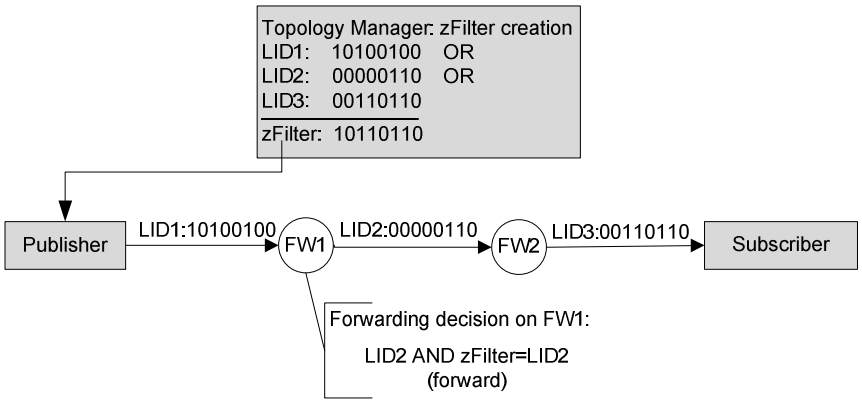

Figure 2. zFilter construction and LIPSIN forwarding mechanism

It takes a bitwise-OR operation over all LIDs of the path to produce an FID which is called a zFilter. This zFilter is sent to the publisher and upon receiving this zFilter, publisher starts sending the requested information item to the relevant subscriber(s) by including the zFilter in the packet header to be used for routing purpose. A simplified example of the forwarding mechanism is shown in Fig.2. When the FW receives a packet it extracts the zFilter from the packet's header and performs the membership test against all its outgoing interfaces to determine where the packet should be forwarded. The test is done by performing a bitwise-AND operation between the outgoing interface and the in-packet zFilter. If the result is the same LID of outgoing interface (i.e. LID AND zFilter = LID), the FW assumes that this interface is part of the delivery tree so it forwards the packer through it. Each FW does the same procedure with all its outgoing interfaces until the packet gets delivered to the subscriber.

2.2. False Positives in Bloom-filter Based Forwarding

One main disadvantage of Bloom filters is the false positive issue, where a low rate of false positives may result when forwarding packets. It is inherited in the design of the Bloom filter and occurs when a packet is sent erroneously through an outgoing link which is not part of the delivery tree. An example of false positives is shown in Fig. 3 (we extend the example of Fig.2 by adding a new forwarding node FW3 to the network topology). After creating the zFilter, the forwarding procedure continues in the same manner as explained previously and upon the packet's arrival to the FW2, the zFilter is ANDed with each outgoing link, LID3 and LID4. As a result the packet is forwarded over LID3 as it is one of the delivery tree elements so the condition is satisfied. Also a copy of the packet is sent over the LID4 (i.e. this is because that the condition LID4 AND zFilter = LID4 is true) although the LID4 is not part of our delivery tree. This is a wrong forwarding decision and considered a false positive. The false positives issue causes packets to be unnecessarily duplicated over some extra links and then introducing wasted resources (i.e. traffic consuming). It may become a security threat if exploited by an attacker as he could disrupt the network. He could deny services to an end node or network links by intentionally causing false positives in sensitive parts of the network. In practice, the attacker could launch several types of attacks such as packet storms, forwarding loops, and flow duplication which at the end leads to a significant increase on the network traffic [13]. Below we briefly review these attacks.

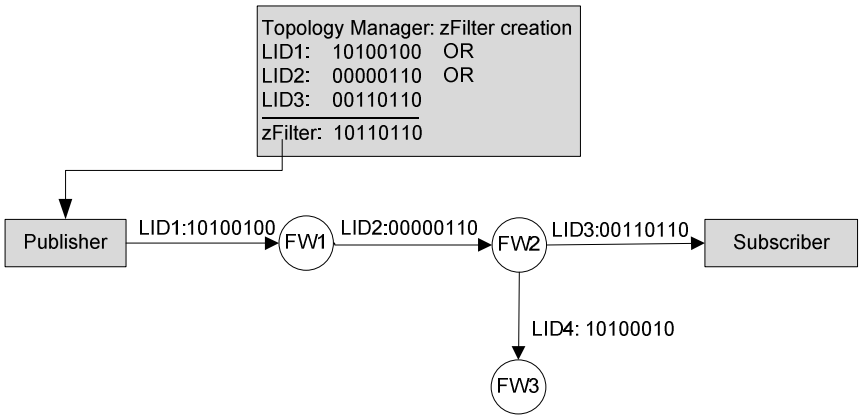

Figure 3. False positives in forwarding decision

Forwarding loops: happen when some false positives occur and lead a packet to be returned to a forwarding node it has already visited. The forwarding node will then forward the packet again to all the nodes downstream of it, including the false positive links that caused the packet to endless loop. An example of this is by assuming a zFilter containing a delivery tree $A \rightarrow B \rightarrow C$ and then after applying the forwarding check to the zFilter a match happen on a separate link $C \rightarrow A$ which is used to forward the packet from node C to node A (i.e. false positive has occurred), so this leads to an infinite loop in the path ABCA.

Packet storms: a high amount of traffic will flood the network if the average number of false positives on a forwarding node exceeds one. This is because under these conditions on average each false positive will lead to more than one false positive in the next forwarding node.

Flow duplication: here the false positives cause accumulated copies of packets to be sent to the intended downstream tree which would lead to an increase in the number of packets according to the Fibonacci sequence.

Several proposals have been introduced to reduce the effect of these issues, including the multistage Bloom filter [14] and Link ID Tag (LIT) mechanism [9]. In the latter one, *d* different LITs are assigned to each outgoing interface. Therefore, for each delivery tree there will be *d* different zFiters created in order to be evaluated in terms of the false positive rate. Then the best performing zFilter is selected and the index *d* is placed in the packet header as a reference to the selected zFilter. From next section we call the link identifier as LID/LIT.

2.3. Security Vulnerabilities in Bloom-filter Based Forwarding

As LIPSIN uses fixed link identifiers and zFilters in the network, it is vulnerable to zFilter replay attacks, computational attacks, and brute-force attacks. All these attacks may lead to DDoS attacks. In zFilter replay attacks, an attacker is able to reach a target by using a previously received valid zFilter that represents the path to the victim, so traffic is sent without subscribers' willingness. Using botnets also the attacker may launch zFilter computational attack by collecting and analysing the correlation between valid related zFilters and their bit patterns. Then the attacker figures the network topology and with this he can attack any victim or even flooding a segment of the network with a huge amount of traffic.

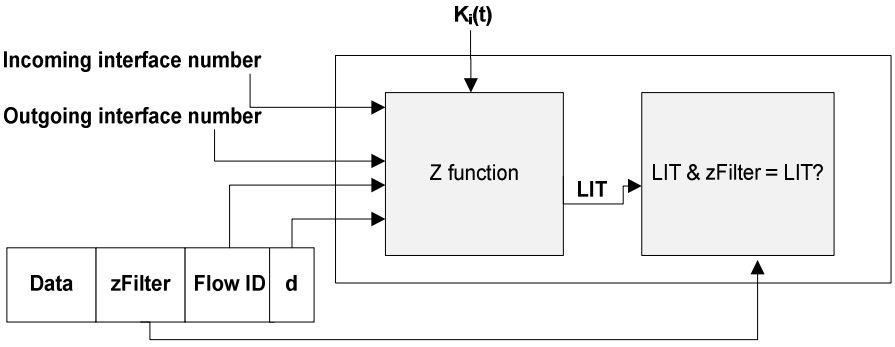

Figure 4. Dynamic computation of LIT in zFormation function

Also brute-force attacks are possible by trying all possible zFilters hoping one of them can cause false positives and passes through the network.

2.4. Security Enhancements to The Forwarding Plane

Our goal is to make the zFilters expire after a certain time by using periodically changing LIDs/LITs. This prevents a malicious publisher from sending data forever. The forwarding decisions here are in a dynamic and computational manner depending on specific inputs. The approach uses a synchronised time bound shared key between TM and each FW to create the corresponding LIDs/LITs, and these keys are changeable in certain time which lead to change LIDs/LITs and then limited life-time zFilter.

In this technique to compute each LID/LIT in the delivery tree, cryptographically secure hash function called zFormation is used. This function accepts four parameters as input and produces changeable LID/LIT with *m*-bit length and *k*-bit set to 1 as shown in Fig. 4. These parameters are: (1) some in-packet-based information ID (flow ID) and here it will be the RID of request, (2) periodically changing time-based secret key K, (3) in-out ports numbers of the hop (where the packet came from and where it is going to traverse), (4) the optimisation index *d* of false positive. Therefore a LIT O= Z(I,K(t), In, Out, d) becomes dynamic and bound to one zFilter with a specific flow, delivery tree and only valid for certain time period.

When TM receives a request to create a zFilter, it calculates the path as in LIPSIN, and then creates the LID/LIT by using the five parameters. At the end an OR operation is taken over all LIDs/LITs to produce a time based zFilter. On the forwarder side, when a packet is received by a FW it creates the LID/LIT by extracting the flow ID and index *d* from the packet and using the input interface where the packet has come from and the candidate output interface, the current time bound key *k* and putting them into the Z function.

The resulted LID/LIT is examined against the received zFilter to take a forwarding decision. This process is repeated with all outgoing interfaces. Therefore, if an attacker managed to figure a zFilter then he would be limited to a short period of time and then he has to repeat the attack process. In [15], we proposed a solution for updating long lived flows with a new zFilter when the current ones are about to expire while communications still in progress.

However, although zFormation approach has solved the issues of replay attack and computational attack, it has failed to prevent brute force attack. In [16], we have analysed this attack and demonstrated that an attacker could break into the network and attack a victim 5-hops away

within a couple of seconds. To prevent such attack with zFormation approach, link identifiers have to be changed more frequently (i.e. every 10 seconds) and this may increase the load on the network. Therefore, in the next section we propose two proposals to mitigate this attack.

3. SECURITY PROPOSALS

In this Section we propose two solutions that mitigate the brute-force and the replay attacks defined in Section 2.3. Generally, instead of checking the zFilter at each FW we only do the check at the ingress of the network (the first FW node on the path). Two proposals are introduced and discussed in the following lines. In these proposals, a FW that is connected to a user will be referred to as EdgeFW. For example in Fig. 5, the EdgeFWs are FW-1, FW-5 and FW-6.

3.1. Key Based Hash Function Approach

In this approach we assume that the TM shares a (different) secret key, K_i, with each EdgeFW. Full description of creating these shared keys is explained later in Section 5. We also assume that the subscriber may subscribe to a whole scope (SID) or only to an information item (RID). If he subscribed to an RID, then after receiving the item, the subscriber must unsubscribe. If he subscribed to a SID, then he is notified about any information items published under that scope and decides whether to subscribe or not.

For example, assume that a publication-subscription matching has occurred at the RV for a particular item, which requires a routing to be done between Pub2 and Sub1 as shown in Fig. 5. Therefore, TM is requested by the RV to construct the delivery path and create the zFilter containing the encoded path (as described in Section 2.1). When the zFilter is created using LIPSIN, the TM takes a hash over this zFilter along with the current time t and the key K_i that is shared with the ingress EdgeWF (i.e.FW1). The result of this hash is sent to Pub2 along with the zFilter and the time t that has been used when creating the hash. This information is put in each packet by Pub2 in order to be forwarded.

When the EdgeFW receives the packet it knows that the packet has come from an end node and not from an intermediate device so it does what is called a security check. In this check, the EdgeFW (FW-1) validates the zFilter whether it is randomly created by the publisher or not. To do this, it extracts the attached hash, t and zFilter from the packet and along with its own shared key K_i it creates a new hash. This new hash is compared with the received one. This is just an authorization check and is required only at the beginning of the forwarding process. It assures that the publisher is using a valid zFilter. If the two hashes are the same, then the EdgeFW assumes that this zFilter is created by the TM and not a fake zFilter. So it forwards to appropriate links by using the LIPISN mechanism. The EdgeFW also records this zFilter as valid for TTL (time to live) of Δt seconds for this particular publisher (i.e. Pub1) as shown in Table 3. Since the publisher does not know the key K_i he is not able to create a valid hash for a fake zFilter. After authentication process publisher only sends zFilter with packets assuming a similar zFilter has been authenticated previously. After authenticating the zFilter, any coming packet at EdgeFW is checked if it has a valid (recorded in the table) zFilter. If the security check fails, then the publisher is assumed to be an attacker and is blocked. The zFilter remains in the record until the TTL reaches 0 and then is deleted.

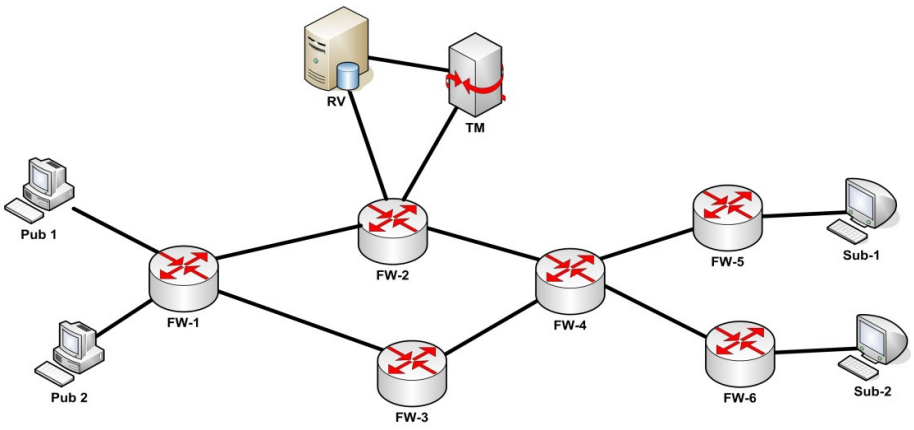

Figure 5. Delivery path from Pub-2 to Sub_1 using EdgeFW

We assume that all zFilters are valid for a time Δt. If time reaches Δt-j seconds (e.g. j is 10 seconds before Δt expires) and the subscriber hasn't unsubscribed yet, then the RV assumes that the subscriber hasn't received the whole item yet and is interested to receive the remaining data. In that case the RV will send a request to the TM to notify EdgeFW to renew the time of this existing zFilter that is linked to an item ID (RID, SID) in the record, by resetting the TTL.

3.2. Local-link Based Approach

In this proposal we distinguish two types of zFilters; EdgeFW-to-EdgeFW: ezFilter (edge zFilter) and publisher-to-subscriber: uzFilter (user zFilter). The zFilter created by the TM is the ezFiler which satisfies the path from the first EdgeFW to the last EdgeFW in the path.

We assume that links from the EdgeFW to the users's end node are local (i.e. not known by the TM). Each publisher is a potential attacker; therefore he is not aware of the ID of his local link identifier (local LID/LIT). Also the SID and RID are sent along with the publisher ID to allow the EdgeFW to know where the received ezFilter should be directed. The local LID/LIT is changed by the EdgeFW per each flow to prevent the replay attack on the local-LID/LIT by the publisher side.

Table 3. Storing zFilters at the EdgeFW

Pub ID	**RID**	**SID**	**zFilter**	**TTL (Sec)**
Pub-2	00000003	00000001	10011101	8
Pub-1	00000005	00000001	11000101	5
Pub-3	00000009	00000002	10110101	3

Table 4. Storing zFilter at the EdgeFW

Pub ID	**RID**	**SID**	**ezFilter**
Pub-2	00000003	00000001	10011101
Pub-1	00000005	00000001	11000101

In this technique, the TM creates the ezFilter from the EdgeFW of the publisher to the EdgeFW of the subscriber to be sent to the EdgeFW of the publisher. When this node receives the ezFilter, it takes an AND operation over the ezFilter and the local-LID/LIT of the publisher. The resulting uzFilter is then sent to the publisher. The EdgeFW records the PubID, RID, SID and the ezFilter

(see Table 4). The publisher, after receiving the uzFilter, places this uzFilter in the packet header to be used when sending data. When EdgeFW gets uzFilter from the publisher, it performs an AND operation of the uzFilter (included in the packet) and the local-LID/LIT. We call this a security check (SC). If matching occurs, the packet is forwarded to appropriate outgoing links by using the corresponding ezFilter and following the LIPSIN mechanism. If no matching occurs, then the EdgeFW assumes that the uzFilter is fake and the publisher is blocked/banned. Each intermediate forwarder directs the packet (after performing the LIPSIN check) until it reaches the last EdgeFW. At this point the packet is forwarded to the subscriber by checking if there is an existing subscription request for the same SID and RID.

We note that in this proposal there would be no updates required for the ezFilters, and this is because the local-LID/LIT are not included in the ezFilter. Also the local-LID/LIT is changeable per each publication and unknown to the publisher. Hence the publisher is not able to construct any uzFilter from the ezFilter. We assume that ezFilters are assumed to be valid until the subscriber unsubscribes so the EdgeFW is notified by the RV to cancel the ezFilter.

However, brute force attacks are still possible but with higher cost. This is because if an attacker has managed to guess the local-LID/LIT and pass the SC (i.e. the created zFilter includes the local-LID/LIT), then he also has to pass the forwarding check with at least one of the outgoing interfaces of the EdgeFW node. If the forwarding check fails then the attacker is blocked from using the network services. So the probability of sending a packet to the second FW is:

$$P(pass\ SC) = \rho_m^{2k} \times x \quad (1)$$

where x is the number of outgoing LIDs, k is the number of hash functions and ρ_m is the maximum allowed fill factor of the Bloom filter.

The attacker's packet can reach the target's EdgeFW if the randomly created uzFliter can pass the SC at the attacker's EdgeFW and the forwarding check along at least one path from the attacker's egdeFW to the target's EdgeFW. The probability of this is:

$$P(reach\ target's\ edgeFW) = \rho_m^{2k} \times x + \rho_m^{k\times(h-1)} \quad (2)$$

where h is the distance (number of hops) from the attacker's egdeFW to the target's EdgeFW.
We analyse the possibility of computational attack, of this approach, to find the ezFilter. This attack can be launched if the attacker collects valid uzFilters and tries to predict the ezFilter to the target. An example of a computational attack is as follows:

For simplicity assume that $k = 2$, $m = 8$ and $\rho_m = 0.5$ and the valid ezFilter that the attacker wants to figure out is 01001110. So the attacker will perform the following steps:

1- The attacker starts with a predicted ezFilter-pred-0 of all ones (i.e. *ezFilter-pred-0 = 11111111*).

2- Assume that initially the local link ID is *local-LID/LIT _1=11000000*, therefore the uzFilter is constructed by the EdgeFW and sent to the attacker and is equal to:

uzFilter_1= ezFilter OR local-LID/LIT _1 = 11001110

3- The attacker creates the new predicted ezFilter as follows:

ezFilter-pred-1= uzFilter AND ezFilter-pred-0 = 11001110

4- The attacker checks the fill factor ρ of the predicted ezFilter whether $\rho \leq \rho_m$ or not. If yes then the ezFilter is found (i.e. the predicted ezFilter contains the valid ezFilter). If $\rho > \rho_m$, then steps 2 to 4 are repeated.

In our example $\rho = 5/8 > \rho_m$, therefore the attacker needs to wait for another uzFilter to repeat the same previous procedure. When a new uzFilter is received by the attacker, he will start the procedure from step 2. Assume that the new local-LID/LIT =01100000, therefore the uzFilter (i.e. uzFilter=01101110) is constructed by the EdgeFW and sent to the attacker's node. In step 3, the attacker creates the new predicted ezFilter which will be 01001110. In the last step that attacker checks the fill factor and in this case it is 4/8 which is equal to ρ_m. This means that ezFilter=ezFilter-pred-2.

Changing the Local-LID/LIT per each request can be considered as countermeasures to prevent the uzFilter computational attack. This attack might be launched where an attacker is trying figure out the local link identifier and then can send bad traffic through a path. In this attack several machines of the attacker's botnet subscribe to the same information item that is available in the attacker machine. Within the same Δt of the local link identifer, the attacker will receive several valid zFilters that are already ORed with the local link. Then the attacker will be able to figure out the local link by analysing the zFilters.

4. DEFFIE-HELLMAN-DSA KEY EXCHANGE PROTOCOL

4.1. Deffie-Hellman Key Exchange

DH [17] can be used as a key distributions protocol and is the first and the most important key establishment mechanism that defined public key cryptography. The purpose of this protocol is to enable two parties to securely establish a shared key, over a public channel, which can be used for subsequent security keys. The generated key is a result of secret values exchange and its effectiveness depends on the difficulties of computing discrete algorithms. This protocol uses two publicly known numbers, a large prime number q and an integer g that is primitive of root q. Assuming user A and B wish to establish a shared key, user A selects a random private integer $v < q$ and then calculates the public key as $y_A = g^v \ mod \ p$ and similarly user B computes y_B using its private number w. Then, A computes the shared key using its private key with the public of user B as $K= (y_B)v \ mod \ p$, and the same way user B calculates the key $K= (y_A)^w \ mod \ p$. By this way, the two parties generate identical results of K. Due to the use of private keys, an attacker is only able to know the public values: q, g, y_A, y_B. Therefore, to compute the key K the attacker is forced to take a discrete algorithm to find the private key of any user which is considered infeasible.

However, this proposed scheme does not provide authentication mechanism for the exchange keys and is also vulnerable to a man-in-the-middle attack, where the communicating parties cannot ensure who is on the other side actually when running the DH key exchange algorithm. Therefore, several modifications have been purposed to this protocol to integrate authentication to

the DH. Arazi [18] proposed a method where the key exchange messages are authenticated using digital signature standard (DSS) [19]. Another attempt is Alexandris-Burmester-Chrissikopoulos-Desmedt (ABCD) system[20] and in 2004, an improvement was made by Harn et al. on Arazi's scheme [21] to provide some security resilience. Integrating Diffie-Hellman protocol into Digital signature algorithm (DSA) leads to some security weakness to the protocol such as key freshness [22], forward secrecy [22], unknown key-share attack [23] and known-key security [20]. The

work on this paper relies on the most secured scheme of the integrated Diffie-Hellman-DSA key exchange protocol [24].

4.2. Digital Signature Algorithm

The DSA consists of three chosen parameters that are public and known to a group of users. A 160-bit prime number q (i.e. $2^{159} < q < 2^{160}$), and a prime number p with length between 512 and 1024 bits such that q divides $(p-1)$, and an integer g such as $g = h^{(p-1)/q} \bmod p > 1$ where h is between 1 and $(p > 1)$. The user selects a random value as private key x where $(q-1) > x > 1$ and generates public key y as $y = g^x \bmod p$, H is secure hash function of message m, and k is randomly chosen integer with $0 < k < q$. (p, q, g, y) are public values. To sign a message m, a user computes two parameters, r and s, as $r = ((g^k \bmod p) \bmod q)$ and $s = (k^{-1} (H(m) + x.r)) \bmod q$. Here k^{-1} is the multiplicative inverse of $(k$ mod q). At the receiver end, to verify of the signature let m', r' and s' are the received version of m, r and s respectively. The receiver checks the signature and it is rejected if r and s are not between 0 and q. Otherwise, the receiver computes $a = (s')^{-1} \bmod q$, $u1 = [H(m')a] \bmod q$, $u2 = (r' a) \bmod q$ and $b = [(gu1\ yu2) \bmod p] \bmod q$. Finally the receiver tests if $b=r'$ then the signature is validated.

4.3. Integrating DH with DSA

We use the three rounds protocol proposed in [24] which includes key conformation at the third round and modify it four our purpose. So, it produces only one shared key for both directions instead of two keys. Fig. 6 shows the procedure of generating a shared key between two users A and B where v and (w1, w2) are short term integer secret values (ephemeral) chosen by the two parties A and B respectively for that session. In step 1 user A selects a random secret value v such as $0 < v < q$ and the values g, p, q according to their definition above, and computes mA and sends the values to user B. Upon receiving (mA, g, p, q) and selecting the values w1, w2 in step 2, user B computes the session key k and mB, rB, sB and sends them to user A. Next user A creates the session key K and then verifies B's signature (rB, sB) on the message (mB‖ K) if the verification holds it means the shared secret key is established. Step 4 acts as a key confirmation to prevent key replay attack so user B can verify the signature (sA, rA) on the message mA ‖ K hence user B convinced that mA received is not a replay attack and indeed sent by user A.

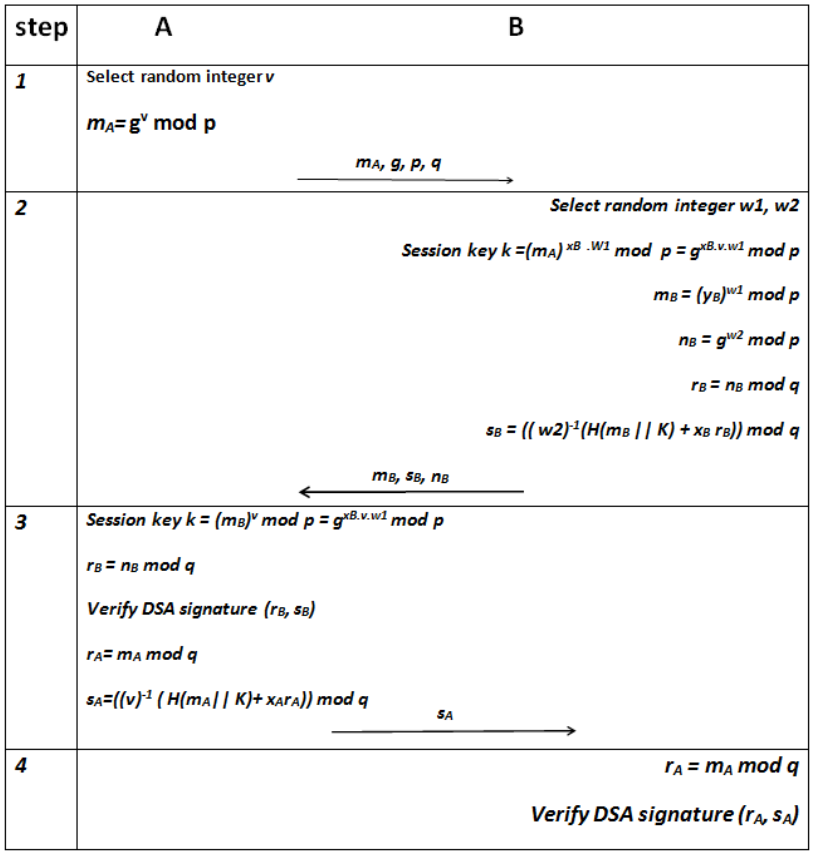

Figure 6. The classic DH-DSA protocol exchange process

5. DEPLOYING DH-DSA UNDER ICN

As the concept of ICN is based on publish-subscribe paradigm the exchange of the secret values is published to specific scope to be subscribed by the other user. To support our proposal in section 3.2 and the technique proposed in [10] which requires a shared key between each FW and TM to make the link IDs dynamically changing, we apply integrated protocol (i.e. DH-DSA) between these two nodes.

5.1. Scopes Setup

We assume some restrictions are applied on the publish-subscribe functionality. This is because if an attacker is able to publish to any sub-scope then he may feed the TM with fake information. Secure Layered Rendezvous can be used so no one can publish and subscribe without authentication (i.e. only the owner of the sub-scope is able to publish or subscribe to the scope). Fig. 7 shows the scopes organization and the procedure of this protocol. There are two main scopes, Node-Scope and TM-scope, created by the TM. TM subscribes to the two scopes, so it is notified if there is an item has been published under any of the two main scopes. Also the FW subscribes to the TM-Scope in order to get any information published or sub-scope created by TM. To avoid any confusion that may occur in the TM side when establishing key exchange process with more than one FW, we add the identity of each node. Therefore, the FW publishes its ID under the two sub-scopes just created. By this stage, the TM is notified about the two sub-scopes created by FW and subscribes to them. Now the two parties are ready to exchange information in order to establish the shared key in public without compromising the key.

5.2. The Protocol Exchange Process

The exchange process is shown if Fig. 7 and started by the FWs side in the following procedure:
Step 1: we consider this step to be done on the in scopes Setup phase where FW-1 publish its ID under each scope.

Step 2: FW-1 selects the public values g,p,q and calculates m_A , then it publishes the values (g,p,q, m_A) to the sub-scope 8754 under Node-Scope. As TM previously subscribed to this sub-scope in the setup phase then it will receive the values. He can recognize the corresponding node from the node ID published previously.

Step 3: T.M creates the session key K using the received values and also calculates m_B *t*o be signed along with the session key. Then it publishes the message m_B along with its signature (i.e. m_B, s_B, r_B) to the sub-scope 7582 under TM-Scope. FW-1 node as subscriber will receive these published information.

Step 4: FW-1 calculates the session key K, using m_B and then it verifies the signature. Then it confirms the key to the T.M by singing it along with m_A and send the signature s_A.

Step 5: TM verifies the signature to be sure that it received a valid value of mA in step 1 and it was not a replay attack. Then it randomly creates a master key K_{master} using cryptography secures pseudo-random function (CSPRNG) and encrypts it using Advanced Encryption Standard algorithm (AES) [25] along with the shared session key. Finally T.M publishes the master key K_{master} in encrypted manner which can only be decrypted by FW-1 using the shared session key. By this, the two parties will share the same key to be used for a secure channel.

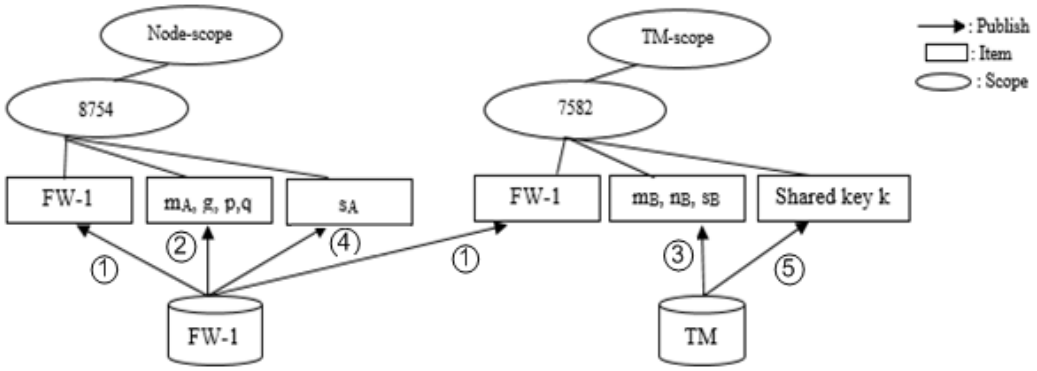

Figure 7. Scopes organization and protocol procedure

6. CONCLUSION

ICN is a clean-slate model aiming to solve many issues existing in the current Internet model. As this architecture is brand new and completely different from that in the current Internet, this paper aimed to redesign the integrated Diffie-Hellman-DSA key exchange protocol as a key distributions mechanism to be applicable under ICN. This protocol can be used in the forwarding plane to offer a better security level. In practical, it can be used with the zFormation approach where LIDs are changeable and created upon a shared key. It also can be used with our recent

proposal that is based on a hash function. With this protocol, network nodes are able to exchange a mater key in a public channel.

REFERENCES

[1] A. Feldmann, "Internet clean-slate design: what and why?," ACM SIGCOMM Computer Communication Review, vol. 37, pp. 59-64, 2007.

[2] G. Wilshusen, "INFORMATION SECURITY Cyber Threats Facilitate Ability to Commit Economic Espionage," U. S. G. A. Office, Ed., ed, 2012.

[3] S. Arianfar and P. Nikander, "On content-centric router design and implications," presented at the Proceedings of the Re-Architecting the Internet Workshop, Philadelphia, Pennsylvania, 2010.

[4] T. Koponen, et al., "A data-oriented (and beyond) network architecture," in Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications, Kyoto, Japan, 2007, pp. 181-192.

[5] COntent Mediator architecture for content-aware nETworks (COMET). Available: http://www.comet-project.org/.

[6] N. Fotiou, et al., "Developing information networking further: From PSIRP to PURSUIT," in The7th International ICST Conference on Broadband Communications, Networks and Systems, 2010.

[7] N. Fotiou, et al., "Towards a secure rendezvous network for future publish/subscribe architectures," presented at the Proceedings of the Third future internet conference on Future internet, Berlin, Germany, 2010.

[8] B. Gajic, et al., "Intra-domain topology manager for publish-subscribe networks," in Telecommunications (ICT), 2011 18th International Conference on, 2011, pp. 394-399.

[9] P. Jokela, et al., "LIPSIN: line speed publish/subscribe inter-networking," presented at the Proceedings of the ACM SIGCOMM conference on Data communication, Barcelona, Spain, 2009.

[10] C. E. Rothenberg, et al., "Self-Routing Denial-of-Service Resistant Capabilities Using In-packet Bloom Filters," in Computer Network Defense (EC2ND), European Conference on, 2009, pp. 46-51.

[11] B. Alzahrani, et al., "Securing the Forwarding Plane in Information Centric Networks" in 5th Computer Science and Electronic Engineering Conference (CEEC), 2013.

[12] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," Communications of the ACM, vol. 13, pp. 422-426, 1970.

[13] M. Sarela, et al., "Forwarding anomalies in Bloom filter-based multicast," in INFOCOM, Proceedings IEEE, 2011, pp. 2399-2407.

[14] J. Tapolcai, et al., "Stateless multi-stage dissemination of information: Source routing revisited," in Global Communications Conference (GLOBECOM), IEEE, 2012, pp. 2797-2802.

[15] B. Alzahrani, et al., "Enabling z-Filter updates for self-routing denial-of-service resistant capabilities," in 4th Computer Science and Electronic Engineering Conference (CEEC), 2012, pp. 100-105.

[16] B. A. Alzahrani, et al., "Mitigating brute-force attacks on Bloom-filter based forwarding," in Conference on Future Internet Communications (CFIC), 2013, pp. 1-7.

[17] W. Diffie and M. Hellman, "New directions in cryptography," Information Theory, IEEE Transactions on, vol. 22, pp. 644-654, 1976.

[18] B. Arazi, "Integrating a key distribution procedure into the digital signature standard," Electronics Letters, vol. 29, pp. 966-967, 1993.

[19] N. I. o. S. a. Technolog, "Federal Information processing Standards Publication: Digital Signature Standards," 1991.

[20] K. Nyberg and R. A. Rueppel, "Weaknesses in some recent key agreement protocols," Electronics Letters, vol. 30, pp. 26-27, 1994.

[21] L. Harn, et al., "Integrating Diffie-Hellman key exchange into the digital signature algorithm (DSA)," Communications Letters, IEEE, vol. 8, pp. 198-200, 2004.

[22] M. A. J., et al., Handbook of Applied Cryptography. Boca Raton: CRC press, 1997.

[23] J. Burton S. Kaliski, "An unknown key-share attack on the MQV key agreement protocol," ACM Trans. Inf. Syst. Secur., vol. 4, pp. 275-288, 2001.

[24] L. Jie and L. Jianhua, "A Better Improvement on the Integrated Diffie-Hellman-DSA Key Agreement Protocol," International Journal of Network Security, vol. 11, pp. 144-177, 2010.

[25] F. P. Miller, et al., Advanced Encryption Standard: Alpha Press, 2009.