Post-print of: J. Konorski, "Mitigating Time-Constrained Stolen-Credentials Content Poisoning in an NDN Setting," 2019 29th International Telecommunication Networks and Applications Conference (ITNAC), Auckland, New Zealand, 2019, pp. 1-7, doi: 10.1109/ITNAC46935.2019.9077973.

Mitigating Time-Constrained Stolen-Credentials Content Poisoning in an NDN Setting

Jerzy Konorski
Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology
Gdansk, Poland
jekon@eti.pg.edu.pl

***Abstract*—NDN is a content-centric networking architecture using globally addressable information objects, created by publishers and cached by network nodes to be later accessed by subscribers. Content poisoning attacks consist in the substitution by an intruder publisher of bogus objects for genuine ones created by an honest publisher. With valid credentials stolen from an honest publisher, such attacks seem unstoppable unless object recipients can afford costly object examination. We argue that limited-time validity of stolen credentials gives rise to a mitigation scheme that does without such examination; instead, propagation of trust in an object is carefully designed. We formulate NDN, trust, and intruder models, and specify the mitigation scheme as a Markovian infection process on a graph, whose desirable properties we establish. We validate through simulations that bogus and genuine objects can be distinguished in a probabilistic sense, and evaluate several introduced measures of interest.**

Keywords—NDN, content poisoning, trust, infection process

I. INTRODUCTION

The new content-centric networking paradigm is attractive for users interested in obtaining a specific globally addressable piece of digital data content, referred to as *object*, rather than establishing a connection with its hosting site. Among the content-centric architectures proposed for the future Internet, Named Data Networking (NDN) [1] has recently received much attention. In NDN, publisher entities create and disseminate objects among subscriber entities via a network of content routers. Owing to advanced routing and caching mechanisms, an object can be quickly made accessible to a large set of subscribers; its later versions or updates can be disseminated just as quickly.

The ease of object dissemination makes NDN vulnerable to *content poisoning* attacks, whereby an intruder publisher creates a bogus object, next disseminated to pose as a fresh genuine object from an honest publisher, or to replace an existing genuine object. A successful attack capitalizes on the complexity (hence, reluctant use) or imperfection of mechanisms whereby recipients place trust in a given object. The consequences are especially disastrous in the case of general-interest objects. E.g., if a bogus object posing as a fresh version of an existing one manages to instill trust of each subscriber, the original version will be overwritten everywhere and impossible to recreate. This is close to the fake news problem [2], whose analyses are often inspired by biological research on disease spreading [3] and draw on the theory of infection on graphs [4], [5], [6], [7]. Similar approaches have been applied to malware propagation [8].

Work funded by the National Science Center, Poland, under Grant UMO-2016/21/B/ST6/03146.

Trust building (and subsequent coping with, or blacklisting of distrusted objects or publishers) is usually based on object examination (subsuming the timing and circumstances of creation), which is costly, highly context-dependent [9], [10], [11] and/or requires very large datasets and AI algorithms [2]. Tight cooperation with other trusted recipients may be also needed to vote out bogus objects as in the LOCKSS system [12]. As a preventive measure, care is taken to endow each object with strong enough publisher credentials, e.g., a public key infrastructure (PKI) supported; however, credentials theft is rampant these days [13].

With valid credentials stolen from an honest publisher and object examination beyond the reach of recipients' simple machines, a content poisoning attack is apparently unstoppable—a bogus object cannot be distinguished from a genuine one. In this paper we posit that unlike an honest one, an intruder publisher is time constrained: because of credentials theft detection that may be in progress and/or credentials expiration, stolen credentials feel like a "hot potato" and so can be used only for a limited time. A defense scheme can leverage this to mitigate content poisoning attacks without costly object examination or sole reliance on publisher's credentials. Our proposed novel scheme instead carefully prescribes how trust propagates in the network as a process resembling infection spreading; unlike in a biological context, infection of a content router (corresponding to instilling trust in a given object) may be desirable.

The adopted NDN and trust models are summarized in Sec. II. Sec. III introduces the notion of time-constrained, stolen-credentials content poisoning attacks and formulates defense postulates. We describe the proposed mitigation scheme in Sec. IV and validate it through simulations in Sec. V. Sec. VI concludes the paper and outlines future work.

II. MODEL

A. Network Model

NDN allows three types of actors: *subscribers* are the users or user applications interested in accessing specific objects, *publishers* are entities who create objects and make them accessible to subscribers through the communication infrastructure of NDN, and a network of interconnected content routers, further called *nodes*, who act as intermediaries between the publishers and subscribers, having suitable protocol interfaces to both. The nodes are responsible for routing and forwarding objects from the publishers to the subscribers using NDN *Interest* and *Data* messages, as well as for disseminating objects along inter-node links so that popular objects are cached possibly close to interested subscrib-

© 2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

ers. Fig. 1 illustrates the model. General-interest objects, such as news bulletins, viral videos, software packages, scientific reports or tutorials, new editions of literary works, that are to eventually reach all the nodes and subscribers, make an attractive target of content poisoning. For the present study, the underlying object naming, routing, transfer, and caching policies are irrelevant and will be left out.

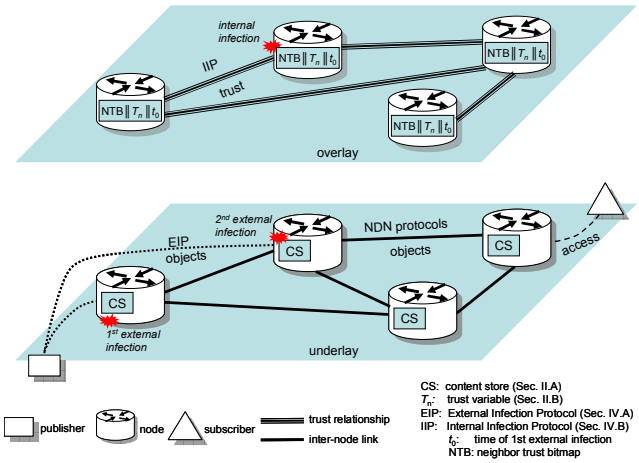

Fig. 1. Elements of the adopted NDN and trust model.

We will use the following notation. Let **P** and **N** be the sets of publishers and nodes, with generic elements p and n, respectively. An object created by a publisher $p \in \mathbf{P}$ consists of a data content file c and a metadata file m, and is denoted $c \| m$ ($\|$ is the concatenation operator). Among other fields, such as object id, descriptors, timestamps, version, freshness, size, access rights, expiration date etc., m contains information relevant to our proposed solution: the publisher id, p, the hash value of the data content, $h = hash(c)$, and the time of creation, t_0 (more precisely, the time of first sending to a network node, cf. Sec. IV.A). Each node $n \in \mathbf{N}$ maintains a content store (CS) for objects received from publishers, to be accessed by subscribers and/or waiting for dissemination.

For the NDN model, we make several assumptions:

- A separate identity management system ensures unique global id's of all the publishers and nodes.
- Each publisher p is assigned (e.g., during network logon procedures) a unique *home node* $n^*(p)$ to which it first sends a created object. Later, p can set up a communication path with any $n \in \mathbf{N}$.
- Each publisher and node uses a pair of PKI supported private and public keys, *priv_key* and *pub_key*; the operations of signing with private key and signature examination with public key are written as $S_{priv\_key}(\cdot)$ and $V_{pub\_key}(\cdot)$, respectively.
- The nodes in **N** maintain network-wide synchronized time, e.g., using NTP [14].
- A public hash function $hash(\cdot)$ is agreed upon.

B. Trust Model

In the context of an NDN setting, the notion of trust translates into a subscriber's conviction that the received object was created by an honest publisher and has not been tampered with on its way. (Note that purely cryptographic means can only assure that the creator was in possession of an honest publisher's credentials.) Hence, a binary trust model is adopted. Our trust-building approach relies on:

- mutual PKI-based authentication during publisher-node information exchange,
- a node's initial perception of an object received from a publisher as trusted,
- the principle that an object in the local CS of a node is made accessible to a subscriber only if the node currently perceives the object as trusted, and
- trust relationships between some node pairs, which permit the nodes to share with one another their current perception of an object as trusted or distrusted.

The last item is crucial to trust propagation within the network. Besides PKI-based authentication, the inter-node trust relationships may have been established in various ways beyond the scope of this paper (one possible mechanism is described in Sec. V.A). Define an undirected *trust graph* $G = \langle \mathbf{N}, \mathbf{E} \rangle$, where $\mathbf{E} \subseteq \mathbf{N} \times \mathbf{N} \setminus \{(n, n) \mid n \in \mathbf{N}\}$ is the set of edges without self-loops; the presence of edge (n, n') signifies a symmetric trust relationship between nodes n and n'. For each $n \in \mathbf{N}$, let $\mathbf{N}_n = \{n' \in \mathbf{N} \mid (n, n') \in \mathbf{E}\}$ denote the set of neighbor nodes[1]. We further assume that

- the trust graph G is connected, i.e., between each pair of nodes there exists an undirected path consisting of edges in **E**, and
- each node n knows $\mathbf{N}_n$.

The trust graph may be thought of as an overlay network on top of the underlay communication infrastructure of NDN inter-node links. While object dissemination uses the underlay part and is independent of trusting or distrusting an object, the sharing of current trust perception among the nodes uses the overlay part, i.e., the trust relationships, as illustrated in Fig. 1. Hence, trust propagates within **N** analogously to objects and much in the way contagious diseases spread within a population; therefore, several terms borrowed from epidemiology seem suitable: with respect to a given object, a node currently perceiving the object as trusted (distrusted) will be called *infected* (*healthy*), and if all the nodes become infected (healthy), we will speak of an *epidemic* (*extinction*). Let each node n maintain a binary variable T_n, set to 1(0) if the node is currently infected (healthy).

As seen from the above description, trust placed in a given object propagates in two ways:

- *External infection*: a publisher sends the object to one or more nodes, each of which stores it in the local CS and places initial trust (sets $T_n = 1$). This process is governed by the *External Infection Protocol* (EIP), which prescribes a publisher-node information exchange and enforces certain conditions of storing the received object at the node.
- *Internal infection*: trust or distrust is shared among the nodes along the trust relationships in G. This process is governed by the *Internal Infection Protocol* (IIP), which defines conditions for setting $T_n = 0$ or 1 at each node n, following an information exchange with neighbor nodes.

[1] "Neighbor" pertains to adjacency in the trust graph G and not necessarily in the NDN communication infrastructure, cf. Fig. 1.

MOST WIEDZY Downloaded from mostwiedzy.pl

We remark that object dissemination and trust propagation run in parallel, but not necessarily in lockstep. Our model is minimalist in that no other trust building mechanism is assumed; nodes or subscribers do not attempt any object examination that might influence trust.

III. Content Poisoning Attack

An intruder publisher willing to disseminate a bogus object, or substitute a bogus version of an existing genuine one, may only do so by externally infecting one or more nodes, where it has to present valid credentials (e.g., PKI-based). The adopted intruder model features:

- stolen credentials – the intruder is in possession of valid credentials stolen from an honest publisher,
- time constraint – the stolen credentials can only be used for a finite time called *impunity_period*, either because of validity expiration or for fear of credentials theft detection.

Accordingly, we will speak of *time-constrained, stolen-credentials* (TCSC) content poisoning attacks. Since TCSC attacks cannot be prevented by publisher authentication, and since independent object examination at the nodes or subscribers has been assumed away, the proposed mitigation relies on appropriately designed EIP and IIP. In what follows we assume that both these protocols involve random events, hence the design goals can be expressed in probabilistic terms. Namely, with high probability:

(a) external infection by an honest publisher eventually leads to an epidemic,

(b) external infection by an intruder publisher eventually leads to an extinction, and consequently,

(c) an endemic (partial epidemic), i.e., the presence of both infected and healthy nodes, persists only finitely long.

Besides the above qualitative goals, quantitative goals can also be postulated. Namely, throughout the infection process, the total number of healthy node-cycles in case (a), $\#H$, and of infected node-cycles in case (b), $\#I$, should be low, implying quick and irreversible accessibility of genuine objects, and restricted and transient accessibility of bogus objects, respectively. If t_{ev} denotes the cycle in which an epidemic or an extinction occurs and $i(t)$ denotes the number of infected nodes in cycle t then $\#I = i(t_0 + 1) + \ldots + i(t_{ev})$ and $\#H = (t_{ev} - t_0 + 1)|\mathbf{N}| - (i(t_0) + \ldots + i(t_{ev}))$.

Note that the design of EIP and IIP seeks an infection mechanism having some given asymptotic properties, which is an inverse problem of the analysis of infection processes on graphs [4], [5], [6], [7], where asymptotic properties of a given infection mechanism are sought.

IV. Proposed Mitigation Scheme

With respect to a given object, we observe the following:

- IIP should allow both eventual epidemic and eventual extinction under the same configuration of its underlying infection mechanism, as opposed to the epidemic threshold behavior of classical infection models such as SIS or SIR [5]; the eventual behavior should be decided by the actions of the publisher.
- Unless some idiosyncratic IIP is applied, an eventual epidemic occurs with a probability that increases with the number of externally infected nodes in the course of EIP execution.
- Infecting sufficiently many nodes under EIP to ensure a high probability of an eventual epidemic should take long enough to differentiate the chances to infect all the nodes by an intruder publisher, who is time-constrained, and by an honest publisher, who is not.
- Consequently, it is enough if EIP has nodes externally infected by a publisher one at a time and at predefined time intervals, further called *cycles*.

In theory, a straightforward defense against TCSC content poisoning could utilize a revocation mechanism that, having detected that too few nodes were infected under EIP, broadcasts throughout the network a special message to permanently set $T_n = 0$ at each node n. However, detection of EIP execution termination would have to be arbitrary and the number of externally infected nodes is not locally visible to any node. Moreover, election of a node to generate the special message would be cumbersome, whereas allowing multiple special messages would impose significant communication overhead. Finally, a malicious intruder could easily revoke genuine objects provided by honest publishers. Therefore, our design of IIP does without any reactive measures and instead controls the probability of an eventual epidemic as a function of the number of externally infected nodes.

A. EIP Operation

The goals of EIP are to ensure mutual authentication of a publisher and a node, and sequentiality and time spacing of external node infections. It also aims to prevent smart external infection policies by an intruder aware of the topology of the trust graph G. The former goals are naturally fulfilled using cryptographic means, whereas for the latter, selection of a next node to be externally infected is left to the network side. The following description of EIP as executed between a publisher $p \in \mathbf{P}$ and a node $n \in \mathbf{N}$ is related to a given object $c \| m$; recall that c is the data content and m is the metadata whose relevant fields are p and $h = hash(c)$. For each object, EIP is executed in the same way. Besides information regarding the object, p and n exchange EIP control information $e = n_{last} \| n_{next} \| t_0$, where n_{last} and n_{next} are the last externally infected node and the next node to be externally infected, respectively, and t_0 is the time of the first external infection.

Upon creating a new object, p signs its metadata m with $priv\_key(p)$ and sends it, along with null EIP control information, to its home node $n^*(p)$, thereby initiating the process of external infection. In the course of this process, when infecting a subsequent node n, p appends EIP control information it received from n_{last}, signed with $priv\_key(n_{last})$.

In response to reception of an object from p, node n performs as follows. First, it retrieves p, h, and other fields of m if necessary, to check if the object is already present in the local CS. If not, n checks that $pub\_key(p)$ holds a valid PKI certificate and verifies p's signature in m, as well as checks c for integrity ($h = hash(c)$). If the appended EIP control information is null then it also checks that $n = n^*(p)$; otherwise it retrieves n_{last} and n_{next}, and checks that $pub\_key(n_{last})$ holds a valid PKI certificate and verifies n_{last}'s signature, as well as checks that $n = n_{next}$. If all the above checks are satisfactory, n sets $T_n = 1$, i.e., places initial trust; if $n = n^*(p)$, it also rec-

MOST WIEDZY Downloaded from mostwiedzy.pl

ords the time t_0 of the first external infection as the current clock value. Finally, the object along with T_n and t_0 values is stored in the local CS, and node n waits for a *cycle_duration* before sending an acknowledgment to p. With a properly configured *cycle_duration*, the latter operation deliberately slows down the external infection process and so tightens an intruder publisher's time constraint (which an honest publisher does not mind). If the object was already present in the local CS (e.g., due to earlier infection under IIP being executed in parallel, cf. the next subsection), n proceeds to the acknowledgment phase immediately, the purpose in this case being to just specify for p the next node to be externally infected. During the acknowledgment phase, n_{last} is updated to n, n_{next} is selected according to a defined selection policy to be discussed later, t_0 is appended, and the resulting EIP control information is sent to p, signed with *priv_key*(n). Note that to circumvent the selection policy, an intruder publisher might deliberately send an object to a node where it is already stored until a more convenient n_{next} is returned; this, however, would require the intruder to follow the object dissemination within **N** and would slow down the external infection process even more, further tightening the intruder's time constraint.

Reception of an acknowledgment from n causes p to retrieve $n_{\text{last}} = n$ and n_{next}, check that their private keys hold valid PKI certificates, and verify n_{last}'s signature. If these checks are satisfactory, p sets up a communication path to n_{next} and sends the object to n_{next}, along with the received EIP control information including the unchanged t_0.

A semiformal specification of EIP is presented in Fig. 2. The protocol is configured with *cycle_duration* and the selection policy. Note that p cannot modify the sequence of externally infected nodes as these are selected and signed on the network side, or speed up the infection process by sending the object to a next node without an acknowledgment from a previous one. Neither can n create spurious objects, spoof other nodes' acknowledgments, or withhold acknowledgment indefinitely without p noticing. Clearly, EIP can be abused if p and n collude (a situation we assume away and leave to future work), in which case all kinds of content poisoning are possible even without credentials theft.

B. IIP Operation

Once stored in the CS of at least one node, any object is disseminated within **N** along inter-node links to eventually become accessible to all interested subscribers (recall that the trust graph G is connected). However, trust placed in the object does not automatically propagate along; instead, it constitutes a separate internal infection process governed by IIP. With respect to each object, IIP prescribes an exchange of IIP control information between neighbor nodes in G, and may be executed in parallel with both object dissemination and EIP. IIP should ensure a desired dependence of the probability of an eventual epidemic upon the number of nodes externally infected under EIP as postulated in Sec. III: a small enough number should almost always lead to an extinction, and a large enough number to an epidemic.

On behalf of each object $c \| m$, IIP runs a finite state machine at each node n whose local CS contains the object. Its states dictate the setting of T_n, which may vary cycle-by-cycle as the internal infection progresses; the duration of a cycle, *cycle_duration*, is the same as in EIP. We propose a heuristic finite state machine based on several intuitions:

- The main challenge is to ensure that the number of nodes externally infected under EIP controls eventual behavior of the infection process (epidemic or extinction) even though IIP is executed in parallel. To this end, sojourn times in states featuring $T_n = 0$ and $T_n = 1$ (referred to as *healthy* and *infected*, respectively), should be carefully balanced.
- A long period without an epidemic is indicative of an imminent extinction; to speed it up it is reasonable to disallow further node infections.
- Trusting an object comes about as a result of a large enough proportion, say at least ξ, of neighbor nodes who trust it.
- Likewise, distrusting an object comes about as a result of a large enough proportion, say at least ξ', of neighbor nodes who distrust it, where typically $\xi' < \xi$.

⟨p: creation of new object $c \| m$⟩

```
e ← null;                          // EIP control information
sm ← S_priv_key(p)(m);             //sign metadata
send c‖m‖sm‖e to n*(p)             //first external infection
```

⟨n: reception of $c \| m \| sm \| e \| se$ from p⟩
// $se = S_{priv\_key(n_{last})}(e)$, se = null if $n = n^*(p)$

```
retrieve p, h, and t0 from m;
if c‖m not in CS
    check certificate validity of pub_key(p);
    check V_pub_key(p)(sm) = m;          //verify p's signature
    check h = hash(c);                   //content integrity
    if e = null
        check n = n*(p)                  //n is home node
    else
        retrieve n_last and n_next from e;
        check certificate validity of pub_key(n_last);
        check V_pub_key(n_last)(se) = e;  //verify n_last's signature
        check n = n_next;
    if all checks OK
        T_n ← 1;                         //place initial trust in object
        if e = null
            t0 ← clock;                  // time of first external infection
        add c‖m‖T_n‖t0 to CS;
        wait(cycle_duration);
n_last ← n;
n_next ← select;                         //according to selection policy
e ← n_last‖n_next‖t0;
se ← S_priv_key(n)(e);                   //sign EIP control information
send m‖e‖se to p                         //acknowledgment
```

⟨p: reception of $m \| e \| se$ from n⟩
// acknowledgment

```
retrieve n_last = n and n_next from e;
check certificate validity of pub_key(n_last);
check certificate validity of pub_key(n_next);
check V_pub_key(n_last)(se) = e;         //verify n_last's signature
if all checks OK
    send c‖m‖sm‖e‖se to n_next
```

Fig. 2. Specification of EIP.

Fig. 3 depicts the proposed finite state machine at node n. Four states are distinguished: Healthy-Quarantine (HQ), Healthy-Susceptible (HS), Infected-Acute (IA), and Infected-Recoverable (IR). Per-cycle state transitions are governed by

MOST WIEDZY Downloaded from mostwiedzy.pl

four IIP parameters: ξ, τ, z_{HQ}, and z_{IA} explained below; t_0 is the time of the first external infection, $\%_n = |\mathbf{I}_n|/|\mathbf{N}_n|$ denotes the proportion of infected neighbors of node n, where $\mathbf{I}_n = \{n' \in \mathbf{N}_n \mid T_{n'} = 1\}$, and $random(\pi)$ denotes a random event with probability π. Initially, the HS state is set unless node n is externally infected, in which case the IA state is set.

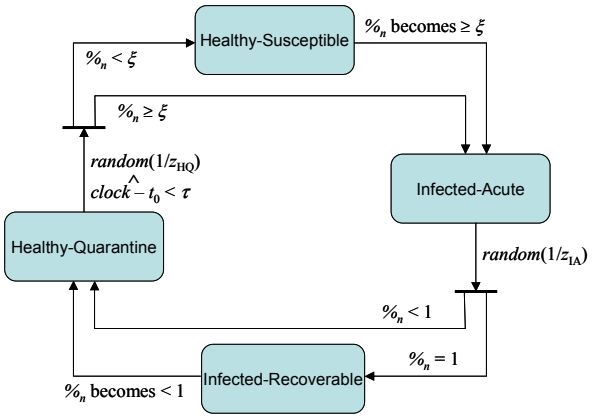

Fig. 3. IIP finite state machine at a node.

In the HQ state, $T_n = 0$ is set and, in line with the first of the above intuitions, a sojourn time, possibly of variable duration with mean z_{HQ} cycles, is enforced (where $z_{HQ} \geq 1$) to prevent n from infecting other nodes too soon; for simplicity, the sojourn time is taken to be memoryless, i.e., geometrically distributed. In line with the second of the above intuitions, transitions towards infected states are no longer allowed after τ cycles have elapsed since the first external infection under EIP (where $\tau \geq 1$).

As long as fewer than τ cycles have elapsed, transitions out of the HQ state are allowed and governed by the proportion of infected neighbor nodes according to the third and fourth of the above intuitions. If the proportion is small, i.e., below a critical ξ (where $0 < \xi \leq 1$), then node n moves to the HS state, where $T_n = 0$ is retained until more neighbor nodes are infected and a transition to the IA state follows; otherwise it moves directly to IA with $T_n = 1$, i.e., becomes infected. In the latter case, a geometrically distributed sojourn time with mean z_{IA} cycles is enforced (where $z_{IA} \geq 1$) in line with the first intuition, to give node n a chance to infect sufficiently many other nodes.

Transitions out of the IA state are again governed by the proportion of infected neighbor nodes. If all of them are infected, $T_n = 1$ is retained and node n waits (perhaps indefinitely) for at least one neighbor node to become healthy, whereupon it returns to the HQ state. Otherwise, a transition to the HQ state follows promptly after one cycle.

To implement the above finite state machine related to a given object $c \| m$, each node n maintains a neighbor trust bitmap (NTB) containing $T_{n'}$ values for all $n' \in \mathbf{N}_n$, and each time T_n changes, IIP control information $m \| T_n \| t_0$ is sent to $\mathbf{N}_n$ along overlay trust relationships using underlay internode links. The current NTB controls transitions triggered by the proportion of infected neighbor nodes, while t_0 controls the period when transitions out of the HQ state are allowed. Two binary random number generators, configured with $1/z_{HQ}$ and $1/z_{IA}$, are also necessary. Finally, certain optimizations are recommended to reduce the communication overhead, such as aggregating IIP control information related to different objects before sending to a neighbor node, or piggybacking on disseminated objects whenever possible.

With respect to a given object $c \| m$, denote by x_n the current state of node n ($x_n \in \mathbf{X} = \{HQ,HS,IA,IR\}$); a network state vector (x_n, $n \in \mathbf{N}$) with $x_n \in \mathbf{Y} \subset \mathbf{X}$ for all $n \in \mathbf{N}$ will be called all-$\mathbf{Y}$. It is easy to see that the cycle-wise evolution of the network state vector according to Fig. 3 follows a discrete-time multidimensional Markov chain starting from all-{HS}. External infection under EIP occasionally causes one of the nodes to assume the IA state, i.e., forces a change of the network state vector. Clearly, the multidimensional Markov chain is non-ergodic: all-{IR}, corresponding to an epidemic, is an absorbing network state vector (at no node will $\%_n$ ever drop below 1), and unless the publisher of $c \| m$ is allowed to externally infect arbitrarily many nodes, so is all-{HQ,HS}, corresponding to an extinction (at no node will $\%_n$ ever exceed ξ). Our content poisoning mitigation scheme builds upon the hope that a TCSC intruder publisher will have externally infected too few nodes to reach all-{IR}.

Proposition 1: If there are only finitely many forced changes of the network state vector (each publisher is allowed to externally infect finitely many nodes under EIP) and the trust graph G is connected then all-{HQ,HS} and all-{IR} are the only absorbing network state vectors.

Proof: Suppose to the contrary that no further external infections occur and a network state vector with both infected and healthy nodes persists forever. None of the infected nodes can be in the IA state, since a transition to HQ or IR would have almost surely occurred, hence all the infected nodes must be in the IR state. However, this is impossible, since at least one of them must have a healthy neighbor node due to the connectedness of G, thus a transition to HQ must have occurred. ■

Proposition 2: If $\tau < \infty$ and the trust graph G is connected then each of the two absorbing network state vectors, all-{HQ,HS} and all-{IR}, is reachable starting from any other network state vector.

Proof: Suppose $t > \tau$, then at a sufficiently large $t' > t$, no further state transitions are possible. Indeed, nodes in the HQ state will remain there, those in the HS state for which $\%_n \geq \xi$ is fulfilled will move to the IA state, and from there will almost surely have moved to the IR state. If $\%_n < 1$ for at least one node in the IR state then, by the connectedness of G, all the nodes in the IR state would have moved to the HQ state, at least one per cycle, and all-{HQ,HS} will persist forever; otherwise all-{IR} will persist forever. ■

Propositions 1 and 2 jointly ensure that the postulates (a), (b), and (c) of Sec. III are fulfilled.

V. Simulations

Monte Carlo simulations carried out in an NDN setting were to answer the question whether, and with what configurations of EIP and IIP, the proposed scheme indeed can mitigate TCSC content poisoning attacks. In a generic simulation run, only k nodes were externally infected under EIP and further trust propagation occurred under IIP, with the understanding that an intruder publisher can afford a small k, whereas an honest one an arbitrarily large k. The probability of eventual epidemic, $Pr[epidemic]$, was measured as a function of k varying from 1 to $|\mathbf{N}|$, and so were object accessibility characteristics, $\#H$, and $\#I$.

MOST WIEDZY Downloaded from mostwiedzy.pl

A. Simulation Setting

Throughout the simulations, $|\mathbf{N}| = 1000$ was fixed. Each node acted as a home node to one publisher and was connected to an arbitrary number of subscribers. The underlay communication infrastructure permitted to set up a path between any two nodes. Objects were assumed to disseminate via NDN *Interest* and *Data* messages with negligibly small delays compared to trust propagation within the overlay trust graph G. In particular, if a node n was to be internally infected upon $\%_n$ becoming at least ξ, the related object had already been stored in the local CS of n.

Two types of trust graph G were considered:

- **Probabilistic Duplication** (PD) graph, a minor modification of [15], reflects a principle that a newcomer node invited to the network by a network node trusts the inviting node and its existing trust relationships. Graph creation proceeds as follows: 1) create any connected subgraph with $j << |\mathbf{N}|$ nodes and at least l edges, 2) link a new node to a randomly chosen existing node and, with a fixed probability φ, each of its neighbor nodes, and 3) repeat the previous step until a connected $|\mathbf{N}|$-node graph is obtained. We take $j = 10$, $l = 20$, and $\varphi = 0.5$.
- **Random** (R) graph, created in the spirit of Erdös–Rényi model, is a connected graph reflecting "accidental" trust building. Graph creation proceeds as follows: 1) add successive nodes, linking each one to a randomly chosen existing node to obtain a connected $|\mathbf{N}|$-node tree, and 2) add successive edges between randomly chosen pairs of the tree nodes until there are as many edges as in the above PD graph.

For EIP, the configuration of *cycle_duration* is not relevant, since in the simulation it merely plays the role of a time unit. In real life, it should satisfy $k_1 \cdot$*cycle_duration* $\geq$ *impunity_period*, where k_1 is the minimum k at which Pr[*epidemic*] is regarded as significant (distinctly non-zero).

Three selection policies were considered to decide the next node to be externally infected:

- **random**, with the next node selected at random by the node currently being infected,
- **prop-deg**, selecting a node with a probability proportional to its degree (number of neighbor nodes) in G, and
- **max-deg**, selecting a node of maximum degree.

The prop-deg and max-deg policies require that each node know the entire topology $\mathbf{E}$ of the trust graph G.

For IIP, $\xi = 0.25$ and $z_{HQ} = 1$ were uniformly assumed, and $\tau = 200$ cycles was experimented to produce small (large) Pr[*epidemic*] for small (large) k in most simulations ($\tau < 200$ often prevented an epidemic for large k, and $\tau > 200$ often prevented or delayed extinction for small k). Finally, z_{IA} varied from 50 to 500 cycles.

B. Results

Throughout the simulations, all the observed ($i(t)$, $t = t_0,\ldots,t_{ev}$) trajectories eventually led to an epidemic or an extinction, in line with Propositions 1 and 2, though with $\tau = \infty$ and a large z_{IA}, extinction could take extremely long. The reason was that if a certain critical $i(t)$ was reached, many nodes, just after a transition out of the IA state, could repeatedly return there after two cycles spent successively in the IR and HQ states. Such a scenario is prevented with $\tau < \infty$.

For a given k, the interesting characteristics: Pr[*epidemic*], $\#H$, and $\#I$, were estimated from 1000 independent runs with different random seeds, with 95% confidence interval widths under 5% of the obtained mean values. Measurements of $\#H$ ($\#I$) were only taken for k at which the percentage of epidemic(extinction)-type $i(t)$ trajectories was large enough to ensure statistical credibility.

Fig. 4 plots Pr[*epidemic*] against k for the PD and R trust graphs and various z_{IA} under the random selection policy. Most of the plots indeed rise from 0 to 1 as k increases, e.g., for the PD graph and $z_{IA} = 500$, less that 0.1% of runs end with an epidemic at $k = 20$ and almost 98% at $k = 80$. (Thus if *impunity_period* is estimated at, say, 12 hours, *cycle_duration* should be set above 36 minutes.) TCSC content poisoning can thus be mitigated if an intruder publisher is not allowed to externally infect k_1 or more nodes and an honest publisher externally infects at least k_2 nodes, where k_2 denotes the minimum k at which Pr[*epidemic*] is regarded as sufficiently close to 1. Clearly, there is a tradeoff here: k_1 should be fairly large (to exploit the intruder's time constraint) and k_2 should be fairly small (to speed up an epidemic for genuine objects). The distinctly visible "phase transition" nature of the plots is therefore an advantage. By configuring z_{IA} (as well as τ and ξ, whose influence we do not discuss here for lack of space) one can control the above tradeoff. However, $\tau = \infty$, too large a ξ, or too small a z_{IA} do not guarantee an epidemic even for large k, especially in the R graph. Compared with PD, the plots for the R graph are shifted toward larger k, i.e., show more resistance to an epidemic, even though both graphs have the same density (average node degree) $2|\mathbf{E}|/|\mathbf{N}|$. The reason is that the two graphs differ in the *distribution* of the node degree: heavy-tailed (scale-free type) in PD [15], and light-tailed in R; the former is known to be more prone to epidemic [5].

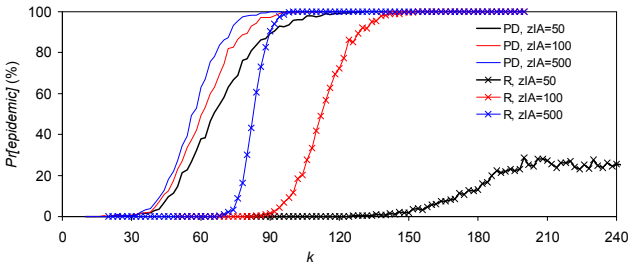

Fig. 4. Probability of epidemic under random selection policy.

Under random selection policy, the two object accessibility measures $\#H$ and $\#I$ (normalized to $|\mathbf{N}|$) are shown in Fig. 5. Both should be kept low. For example, $\#H/|\mathbf{N}| = 100$ would mean that throughout an infection process ending up with an epidemic, thus probably of a genuine object, each node on average refuses access to the object during 100 cycles; likewise, $\#I/|\mathbf{N}| = 100$ would mean that each node on average provides access to a probably bogus object during 100 cycles. While $\#H$ is almost uninfluenced by z_{IA} or k for the PD graph and lower than for the R graph, the opposite is true for $\#I$. This confirms that a smaller z_{IA} should be configured for the PD graph than for the R graph, since an extinction in the former is usually preceded by a long period of $i(t) > 0$.

MOST WIEDZY Downloaded from mostwiedzy.pl

In Fig. 6, *Pr*[*epidemic*] is plotted against k for the PD graph and various selection policies. Favoring high-degree nodes for external infection makes for an earlier epidemic and narrows the "phase transition" (the gap between k_1 and k_2). This illustrates the possibility of abusing our mitigation scheme by a smart intruder publisher knowing G and, possibly in collusion with some network nodes, capable of influencing the sequence of externally infected nodes.

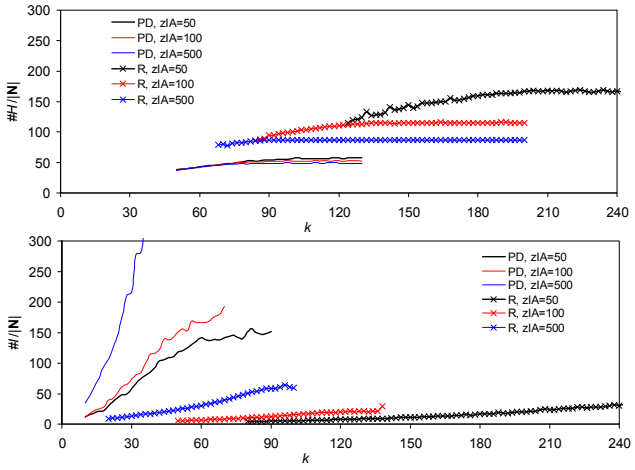

Fig. 5. #*H* (*top*) and #*I* (*bottom*) under random selection policy.

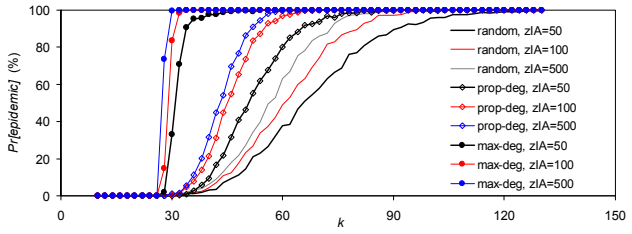

Fig. 6. Influence of the selection policy with the PD graph assumed.

VI. Conclusion

NDN settings are vulnerable to content poisoning by intruder publishers who substitute bogus objects for genuine ones using time-constrained stolen credentials. We have shown that TCSC content poisoning attacks can be mitigated by a combination of two protocols: EIP, allowing external infection of the network nodes one at a time and subject to predefined delays, and IIP, executing internal infection of the network nodes such that an eventual epidemic (network-wide trust) or extinction (network-wide distrust) depends on the number of externally infected nodes. This sets our mitigation scheme apart from existing ones in that it relieves the nodes (and subscribers) from costly object examination, provided that an overlay network of inter-node trust relationships has been built.

We have shown that the proposed Markovian finite state machine of IIP ensures reachability of the two absorbing network state vectors, corresponding to an epidemic and an extinction, and the absence of any other absorbing states. The arising tradeoff between the (slim) chances of an eventual network-wide trust in a bogus object and of an eventual network-wide distrust in a genuine object depends on the topology of the overlay network (e.g., random or scale-free) and can be controlled by a few protocol parameters.

Dynamic self-configuration of IIP, based on the observation of eventual behavior of the infection process (epidemic or extinction) in connection with the number of externally infected nodes, is a logical next step. As the scheme can potentially be abused when some of the publishers and nodes collude, suitable extensions are planned in the future. Among others, elements of object examination may be added to strengthen the proposed mitigation scheme. Finally, integration with mechanisms behind building the inter-node trust relationships in NDN settings, such as, e.g., [16], may be needed for future implementation.

References

[1] L. Zhang et al., "Named data networking," ACM Comput. Commun. Rev., vol. 44, no. 3, pp. 66–73, July 2014.

[2] S. Kumar and N. Shah, "False information on Web and social media: A survey," arXiv:1804.08559, April 2018.

[3] M. Nekovee, Y. Moreno, G. Bianconi, and M. Marsili, "Theory of rumor spreading in complex social networks," Physica A 374, pp. 457–470, 2007.

[4] M. Newman, Networks, 2nd ed., Oxford University Press, 2018 (ch. 16).

[5] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, "Epidemic processes in complex networks," Rev. Modern Phys., vol. 87, issue. 3, pp. 925–980, Aug. 2015.

[6] N.A. Ruhi, Hyoung Jun Ahn, and B. Hassibi, "Analysis of exact and approximated epidemic models over complex networks," arXiv:1609.09565, Sept. 2016.

[7] B. Aditya Prakash, "Propagation and immunization in large networks," XRDS, vol. 19, Issue 1, pp. 56–59, Fall 2012.

[8] V. Karyotis, "A Markov random field framework for modeling malware propagation in complex communications networks," IEEE Trans. Dependable and Secure Computing, vol. 16, issue 4, pp. 551–564, July-Aug. 2019.

[9] C. Ghali, G. Tsudik, and E. Uzun, "Needle in a haystack: mitigating content poisoning in named-data networking," Proc. NDSS Symposium, San Diego, CA, Feb. 2014.

[10] S. DiBenedetto and C. Papadopoulos, "Mitigating poisoned content with forwarding strategy," Proc. IEEE INFOCOM Workshops, San Francisco, CA, April, 2016.

[11] B.K. Saha and S. Misra, Mitigating NDN-based Fake Content Dissemination in Opportunistic Mobile Networks IEEE Transactions on Mobile Computing (Early Access), March 2019.

[12] P. Maniatis, M. Roussopoulos, T.J. Giuli, D.S.H. Rosenthal, and M. Baker, "The LOCKSS peer-to-peer digital preservation system," ACM Trans. Computer Systems, vol. 23, issue 1, pp. 2–50, Feb. 2005.

[13] A. Sharma, Z. Kalbarczyk, R. Iber, and J. Barlow, "Analysis of credential stealing attacks in an open networked environment," Proc. 4th Int. Conf. on Network and System Security, Melbourne, Australia, Nov. 2010.

[14] D.L. Mills, Computer Network Time Synchronization: The Network Time Protocol. CRC Press, 2006.

[15] F. Chung, L. Lu, T.G. Dewey, and D.J. Galas, "Duplication models for biological networks," J. Comput. Biology, vol. 10, no 5, pp. 677–687, May 2003.

[16] J. Konorski and J. Grochowski, "Effect of user mobility upon trust building among autonomous content routers in an information-centric network," Wireless Communications and Mobile Computing, article id 8612817, Nov. 2018.

MOST WIEDZY Downloaded from mostwiedzy.pl