# Privacy in Content-Oriented Networking: Threats and Countermeasures

Abdelberi Chaabane[1], Emiliano De Cristofaro[2], Mohamed Ali Kaafar[1,3], Ersin Uzun[2]

[1]INRIA Rhone-Alpes, France [2] PARC, Palo Alto, CA [3] NICTA, Australia

## ABSTRACT

As the Internet struggles to cope with scalability, mobility, and security issues, new network architectures are being proposed to better accommodate the needs of modern systems and applications. In particular, Content-Oriented Networking (CON) has emerged as a promising next-generation Internet architecture: it sets to decouple content from hosts, at the network layer, by naming data rather than hosts. CON comes with a potential for a wide range of benefits, including reduced congestion and improved delivery speed by means of content caching, simpler configuration of network devices, and security at the data level. However, it remains an interesting open question whether or not, and to what extent, this emerging networking paradigm bears new privacy challenges. In this paper, we provide a systematic privacy analysis of CON and the common building blocks among its various architectural instances in order to highlight emerging privacy threats, and analyze a few potential countermeasures. Finally, we present a comparison between CON and today's Internet in the context of a few privacy concepts, such as, anonymity, censoring, traceability, and confidentiality.

## Categories and Subject Descriptors

C.2.1 [**Computer communication networks**]: Network Architecture and Design; K.4.1 [**Computers and society**]: Public Policy Issues—*Privacy*

## General Terms

Security

## Keywords

Content-Oriented Networking, privacy

## 1. INTRODUCTION

In the last few years, the increasing penetration of the Web in our society has prompted a tremendous growth of data routed in the Internet, to such an extent that global IP traffic is expected to increase 3-fold over the next 5 years [21]. Besides exceeding its expectations, the Internet has also stretched many of the initial assumptions, creating issues that challenge its underlying communication model. Applications increasingly operate in terms of content, making it difficult to conform to IP's requirement to communicate by discovering and specifying hosts and locations.

Coping with massive amounts of traffic becomes arduous due to a number of issues deep-rooted in the network design. As a result, the quest for improving scalability and (cost) efficiency of content delivery has led to the design of overlay networks, such as, Peer-To-Peer (P2P) and Content Distribution Networks (CDNs). However, overlay networks often complicate network management and application development. Further, endpoint authentication mechanisms (whereby an endpoint can only authenticate the counterpart, but not the message) have been challenged by frequent attacks against SSL [26, 31] and the hacking of certification authorities [34].

Motivated by these issues, new architectures have been proposed, in the last few years, aiming to redesign the Internet [41] and accommodate content-oriented applications. In particular, **Content-Oriented Networking (CON)** [18] has set to decouple contents from hosts, at the network layer, by relying on the publish/subscribe paradigm. CON shifts identification from host to content, so that this can be located anywhere in the network. The content-centric communication paradigm introduced by CON relies on naming the content itself, rather than its location. Content is self-contained, has a unique name, can be retrieved by means of an *interest* for that name, cached in any arbitrary location, and digitally signed to ensure its integrity and authenticity.

CON comes with a potential for several advantages, including reduced congestion and improved delivery speed through content caching, simpler configuration of network devices, and security at the data level. However, it remains an interesting open question whether or not, and to what extent, this emerging networking paradigm bears new **privacy challenges**. While some features, such as, lack of source/destination addresses, might help privacy in CON, a closer look to some of the design choices unveils a number of open questions. This paper studies privacy in CON as a generic paradigm and shows that it introduces several worrisome issues.

First, we analyze the implications of *caching* – one of the crucial features in CON used to reduce traffic and improve delivery speed – on user privacy and show that, as nodes cache frequently requested content, they can infer content consumed by others using timing information. Next, we focus on *content privacy*: since publicly available content is not encrypted in CON and, as routers now handle content, any router can easily inspect content. Furthermore, as content in CON is retrieved using names that most likely are semantically related to the content itself, an attacker could infer sensitive information about a user, by monitoring her requests: we refer to this issue as *name privacy*. Finally, we look at *signature privacy*: since digital signatures of CON packets need to be publicly verifiable, the identity of a content signer may be easily inferred by looking at the signature.

In this paper, we discuss different attack scenarios that threaten privacy in CON. For each setting, we describe the attacker capabilities and related impact on user privacy. We suggest several ideas that could serve as countermeasures and detail their strengths and weaknesses, and make sure that they would bear minimal changes to the CON architecture. To the best of our knowledge, this represents the first step toward a thorough analysis of CON privacy issues. In the process, we also highlight a number of challenging open problems that call for further research.

## 2. CONTENT-ORIENTED NETWORKING

This section provides a high-level description of Content-Oriented Networking (CON). A few future Internet architectures have been proposed so far that realize CON. The most prominent ones include DONA [35], NetInf [3], CCN [33], LANES [49], TRIAD [32], CBCB [13]. We now review their building blocks:

1. **Named content:** In CON, objects are always named to facilitate data dissemination and search. Consequently, the security model is also shifted from host to content authentication.
2. **Content-based routing:** Content routing in CON relies on content rather than hosts, aiming to handle increased amounts of network traffic and be more resilient to network bursts and users' mobility.
3. **Content Delivery:** Content is efficiently delivered using multi-path routing and leveraging in-network caching, in order to minimize network bandwidth and delivery delay, and transparently handle mobile users.
4. **In-network storage:** All CON network components provide caching capability. Note that this is different from packet buffers in today's routers, as cache size is expected to be several orders of magnitude bigger in CON.

**Network Model.** CON involves several entities (see Fig. 1). *End users* express interests and fetch data using a wide range of devices. Interests represent the willingness of the user to retrieve certain data, independently of its location. *Content routers* are responsible of forwarding interests and forwarding back the associated data. Each router is assumed to have a built-in cache. Cache size as well as caching algorithm may differ from one router to the other. Finally, *content producers* (or publisher) generate the content—either static (time-independent) or dynamic (generated upon request). Although data-centers or/and geographically distributed servers may be used to serve content, we simplify the model by considering a single source machine.

In the following, we overview CON's architecture design. For further details, we refer the reader to [20, 2], and to the full version of this paper [14].

**Caching.** A key feature used to increase overall network efficiency in CON is caching. All nodes in the network are expected to participate in the caching effort, from core routers to mobile devices. Caches provide in-network storage and are assumed to be several orders of magnitude larger than today's buffers. This capacity allows to store content for longer periods and enhance network performance. There is a number of efforts aiming at optimizing caching strategies in CON – see, e.g., [4, 44, 47, 46].

**Content Naming.** The main abstraction in CON architecture is the Content Object (CO). All kinds of content, ranging from web pages to documents, and even interactive content, such as VoIP, are abstracted as a CO. An object is always identified by a name, which must be unique as it serves as identifier for searching and disseminating associated content. Moreover, since content can be fetched from anywhere, there should be a *secure binding* between content name and content data (i.e., *name-data integrity*) as well as *object authenticity*. Finally, objects retrieved from cache should carry information about the object owner (publisher). Two naming approaches have been proposed so far – flat and hierarchical.

**Flat naming.** Flat names are *self-certifying*, i.e., CO's *name-data integrity* is bounded to its name, thus, it can be verified without any PKI. In its simplest form, the name is expressed as the cryptographic hash function of the content. Although one can assess the validity of the content, it is impossible to verify its provenance and relevance. However, a few techniques have been proposed to enable provenance verification, by allowing the publisher to have more control on the naming by adding a label (e.g. [36, 35]). Unfortunately, these solutions do not guarantee binding between the name and the content. Thus, we will assume the use of secure binding, as proposed by [48], whereby content is made available as a triple $M_{(N,P,C)} = (N, C, Sign_P(N, C))$, where $N$ is the content name, $C$ the content, and $Sign_P$ producer's signature.

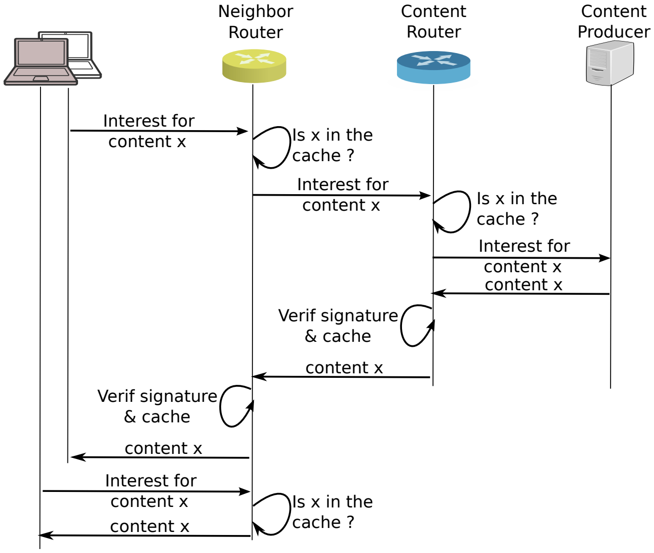

**Figure 1:** An overview of the main CON features: content routing, caching, and content signature. Content is address by name (x).

**Hierarchical naming.** With hierarchical naming, name structure resembles that of today's URLs, where the '/' symbol delimits the name components. In some cases (e.g., CCN [33] and TRIAD [32]), the name is human-readable, which makes it easier for the user to access the content. The major benefit of adopting a hierarchical structure is to enable aggregation, thus, improving routing scalability. Note that, similar to flat naming, we assume the use of secure binding, as proposed by [48].

**Content Routing and Forwarding.** As illustrated in Fig. 1, routing in CON is performed in two phases: (1) routing of CO requests ("interests"), and (2) routing the content back to the user. Naturally, this depends on the naming schema and, in particular, on whether or not name aggregation is possible. For flat-naming based CON, a Name Resolution Service (NRS) is used to retrieve topological information (such as, the data location) based on object name. *Structured routing* algorithms are often used to exploit structured network topologies, such as trees or DHT-s. For example, DONA [35] maintains a tree topology and lets each router store routing information of all his descendants. Thus, any content (re)publication, deletion, or modification is propagated up to the root. With hierarchical naming, efficient routing and discovery is possible without any external service. Request/data aggregation may also facilitate network scalability. This approach may resemble unstructured routing in IP, where IP addresses are replaced with content name and route advertising is achieved through flooding.

**CCN/CCNx overview.** To ease presentation, in the rest of the paper, we will refer to technical details of one specific CON instance, namely, CCNx [43], the open-source project that implements Content-Centric Networking (CCN) [33]. CCNx is considered one of the most mature examples of CON in the research community, with multiple platform implementations, documentations, and implementations, supported by a relatively large community.

In CCNx, whenever a router receives an interest for name $X$, it performs a longest prefix match lookup on its three main tables as follow: First, it looks whether the interest exits in the Content Store (i.e., the main cache), if so, a copy is forwarded back to the user and the routing process terminates. Otherwise, a lookup is launched on the Pending Interest Table (PIT). This table keeps track of all interests waiting to be resolved by upstream nodes. If there is a match, the router collapses the present interest (and any subsequent ones for $X$) storing only the interface on which it was received. Finally, if no match is found, the router searches for the most suitable interface in his Forward Information Base (FIB) to forward the interest and then creates a PIT entry for that interest.

# 3. PRIVACY CHALLENGES IN CON

We now present a systematic analysis of privacy in Content-Oriented Networking (CON) by identifying threats and, when possible, discussing possible solutions. Multiple proposals [35, 3, 33, 49, 32, 13] have been presented in the last couple of years to instantiate CON, with relatively minor differences in their proposed design. As mentioned above, we will discuss technical details while considering the CCNx instance [43], however, threats and proposed countermeasures discussed in this paper apply to CON in general. That is, issues are about fundamental features in CON (such as, caching, naming, data delivery and provenance assurance) and not about the specifics of one implementation.

## 3.1 Cache privacy

We start our analysis with cache privacy. Recall that caching is a fundamental component of CON as it benefits both latency and bandwidth consumption. However, it also introduces a fundamental challenge to user privacy. An adversary may use a router's cache to infer content exchanged (consumed) by users in the downstream and possibly link it to a specific user depending on its relative location to the user in the topology. There are different types of attacks that can occur on cache privacy – we review them below.

**Timing attacks.** By measuring time [27], an adversary $Adv$ can determine if a content has been cached at a particular router by measuring the delay to retrieve it. To do so, $Adv$ measures the $RTT_s$ to retrieve any content from the source, the delay $RTT_c$ to get cached content from the closest router, and the delay $RTT_t$ to fetch targeted content. Then, $Adv$ compares the $RTT$ as follow:

- If $|RTT_t - RTT_c| < \epsilon$ (for negligible $\epsilon$): $Adv$ concludes that target content has been cached at the closest router (i.e., has been fetched by a neighboring consumer connected to the same router).
- If $RTT_t > RTT_c$ and $RTT_t < RTT_s$: $Adv$ knows that target content has been fetched from the source recently and cached in the network, but not by one of its immediate neighbors. Based on the difference between $RTT_t$ and $RTT_c$, $Adv$ can still predict how close the consumer of that content was to his location in the network topology.
- Otherwise ($|RTT_t - RTT_s| < \epsilon$), $Adv$ concludes that target content has not been consumed recently.

Such an attack allows $Adv$ to check whether or not content has been recently fetched, but not when. Launiger et al. [38] describe the timing attacks in the context of CON, and show that the cache replacement policy has only a small impact on the attacks' success. Finally, observe that, since in-network caches are shared by all CON designs, this attack is inherent to *all* CON proposed architectures.

**Protocol attacks.** Without a careful design, content retrieval protocols and their features in CON architectures can make access to cache content even easier. After investigating such issues in CCNx [43], we found out that a number of features and options in interest packets [42], and how they are matched with content packets, are particularly worrisome:

- *Prefix-based matching*: CCNx considers a content with name $X$ to satisfy an interest for name $Y$ if $Y$ is a proper prefix of $X$. This can facilitate easy extraction of cache content without knowing exact names. Due to multiple types of content potentially satisfying an interest, an exclusion option is also conveniently provided in CCNx interest packet format to allow exclusion of previously acquired content from subsequent queries.
- *Scoping*: In CCNx, scope for interest packets is used to determine the maximum number of hops it will travel. Such a feature makes it easy to query the caches of particular routers as it controls where (i.e., how many hops away) an interest packet can travel to. For instance, setting the scope to 2 would restrict an interest to propagate to only neighbouring router(s) and allow convenient querying of their caches without relying on any timing information.

As a result, $Adv$ can *monitor* the access to sensitive content within a certain scope or easily *dump* nearby caches' content. The former attack is achieved by periodically issuing an interest $I_m$ for target content $m$ and setting the scope accordingly (e.g., setting scope to 2 to monitor the caches of immediate neighbors). If $I_m$ times out, $Adv$ concludes that $m$ has not been fetched yet. Whenever a consumer within the scope accesses $m$, $I_m$ is satisfied, thus, allowing $Adv$ to be notified. Hence, $Adv$ is able to infer *when* the file was fetched for the *first* time. Dumping attacks can be achieved by sending an interest for the root prefix $/$ or short prefixes, repeatedly, and excluding what has been already received on successive interests. Combined with scoping, this method can easily be used to dump cache contents from nearby caches.

We believe that the above attacks are quite worrying, however, observe that the attack success depends on the relative location of the adversary to the victim in the network topology. Thus, we distinguish two classes of adversaries:

- *Immediate Neighbor:* if an attacker is sharing the first hop CON router with his potential victim, the privacy risk is maximized as it would not only be easy to singularly monitor or dump a close-by router, but also victim's anonymity set would be very small, due to the limited number of users sharing that router.
- *Distant Neighbor:* Considering the tree-like topology in content distribution from its original source to its consumers (where the source is the root, consumers are the leaves, and the intermediate routers are nodes in between), the path from an adversary and a consumer to the root will intersect at least one node. Therefore, the privacy risk decreases as the number of leaves in the sub-tree rooted at that node increases (i.e., anonymity set gets larger).

### *Potential Solutions*

Different algorithms have been proposed to enhance the hit ratio [4, 44, 47, 46] on caches, however, none of them takes into consideration potential related privacy issues. In the following, we discuss some potential countermeasures to mitigate such privacy threats at a high level. Although the detailed design and security analysis for these methods are not within the scope of this paper, we expect the research community to further investigate them in future work.

**Wait before reply.** A simple solution to the cache privacy problem is to delay *all* requests: when the router fetches content $m$, it should store the corresponding RTT $t_m$. Then, whenever a user requests $m$, the router waits $t_m$ before sending the data back. Note that, independently from our work, Acs et al. have recently proposed a similar solution [1]. Wait before reply has three main advantages: (1) it provably achieves perfect privacy since $Adv$ cannot distinguish between cached and not cached data [1]; (2) it does not make any assumptions about content correlation, network topology, or consumers, and (3) it achieves reduced bandwidth thanks to caching. Unfortunately, however, this approach has the main drawback of eliminating the positive effect of caching on content retrieval delay.

**Delay the first *k*.** An alternative to the above solution is to delay the first $k$ requests for content $m$ to ensure that only popular content is cached on edge routers serving small number of customers, similar to what Acs et al. have recently (and independently from our work) proposed in [1]. Note that $k$ should be chosen randomly by routers, otherwise an adversary could break this schema by issuing $k$ requests and timing responses. The main advantage of this approach is that consumers accessing popular content are unlikely to experience any delays introduced by routers. However, $k$ should be carefully and randomly chosen for each content. High values of $k$ result in delaying most of the requests, whereas, a small value will have a negative impact on user's privacy by reducing the anonymity set. Furthermore, the delay on the retrieval of not so popular content will still be high. Finally, we note that Acs et al. [1] provide a formal model that allows to quantify the tradeoff between privacy offered by various caching algorithms and the latency.

**Collaborative caching.** Multiple nearby caches could collaborate to create a distributed cache that is serving a bigger set of users. Such an integration would create an illusion of a single cache of bigger size and would also increase the anonymity set for customers. Several algorithms have been proposed in this context [37, 51] and can be categorized as hierarchical or mesh based. The latter refers to a flat structure while the former is used for caches that have a tree-like structure. In a simple mesh scenario, two routers $R_1$ and $R_2$ collaborate as follows: the hash universe (e.g., 256 bits) is divided in two subspaces $s_1$,$s_2$ where $R_1$ and $R_2$ stores the elements in $s_1$ and $s_2$ respectively. Based on computed hash (of the routable prefix of the content name), the router decides whether to cache that content or transmit it to his neighboring router. Similarly, interests are first forwarded to the router responsible for caching the corresponding subspace. Collaborative caching has two main advantages: first, users are likely to fall into larger anonymity sets even if the requested content was found in cache. Second, hit rates for caches will increase as the collaboration would remove redundancy between nearby caches and effectively simulate a cache that is much bigger in capacity. Due to this second property, there may be some economic incentive to deploy this solution besides protecting user privacy as well [50].

**Probabilistic caching.** Introducing randomness in the caching procedure may impact the accuracy of attacks. One possible approach could be probabilistic caching [44], where a router decides to cache content based on his position on the forwarding path as well as the available space in the cache. Since this decision is based on internal states of routers, it would not be known to an adversary. However, not caching a random subset of content can provide only a very limited privacy protection as the cached subset would still violate user privacy.

## 3.2 Content privacy

Unencrypted communication over IP networks can be spied upon using Deep Packet Inspection (DPI) [6] by an adversary on the end-to-end communication path. However, in CON, content privacy becomes an even more serious threat due to the presence of persistent memory (caches) within the network.

**Monitoring and Censorship:** DPI tools are already commonly used by certain governments or Internet Service Provider (ISP) for classifying and censoring content (e.g., based on keywords). However, DPI on IP networks requires powerful adversaries that are strategically located on the main communication path and with enough computation power to perform DPI on line speed. As CON stores data packets for long time and makes it available to anyone that asks for it, neither of these assumptions about the adversary holds. In fact, the adversary might retrieve content from caches for DPI based monitoring, classification and censorship.

### *Potential Solutions*

As the problem is caused by the lack of data confidentiality, encryption would be the de-facto solution. Naturally, the encryption mechanism should provide the best balance between security and efficiency. Also, it should be preserving the benefits of caching mechanism.

**Symmetric/Asymmetric encryption.** A trivial approach might be to use similar mechanisms to SSL/TLS, where a client generates a session key and encrypts it using the producer public key. After receiving this key, the producer use it to encrypt the content and send it back to the user. The main consequence of such approach is disabling caching mechanism as only one user can decrypt the content.

**Broadcast encryption [28, 8]** allows a "broadcaster" to send an encrypted message to a set of receivers $n$, each of which has a different private key. Given any subset of $n$, the broadcaster can construct an encrypted message so that only the receivers in the subset can decrypt it. Using broadcast encryption to guarantee confidentiality in CON presents several advantages. First, the publisher of a content can encrypt it only once, for a known subset of users. Also, the publisher can precompute and store, or generate new decryption keys on the fly, for already published content. Further, since encrypted content can be consumed by many users, the benefit from caching can be preserved in the network. However, the publisher should generate and store as many keys as the number of clients ($n$). Also, producer's public key and ciphertexts would be of size $O(\sqrt{n})$ [8], which may result into a significant communication overhead.

**Proxy re-encryption [7]** allows a third-party (called proxy) to (re)encrypt a ciphertext which has been encrypted for Alice, so that it can be decrypted by another user, e.g., Bob. The proxy is considered "semi-trusted" because it does not see the content of the messages being translated. In the CON scenario, the content provider could generate a pair of public/private key ($PK_P$/$SK_P$) for each content object. The content, $m$, is then encrypted as $m_s = ENC(M, SK_P)$. Whenever a client $C$ (with public/private key pair $PK_C$/$SK_C$) retrieves the content $m_s$, it queries the content publisher to generate a re-encryption key by sending $PK_C$.

$C$ then receives a transformation key $PK_{PC}$ from the publisher that allows him to re-encrypt content $m_s$ so that he can decrypt it (i.e., $DEC(RE-ENC(m_s, PK_{PC}), SK_C)$ is the original content $m$). This allows the message to be encrypted only once and lets the producer retain control over the decryption since he can refuse the delivery of the transformation key. Also, key management is simplified as the producer creates transformation keys on the fly and does not have to store any additional keys besides his $PK_P/SK_P$. The encrypted content is disseminated as $m_s$ to all users and allows them to benefit from nearby caches. However, it requires both asymmetric encryption and re-encryption (key transformation), which are computationally more expensive than commonly used symmetric-key encryption algorithms. Nonetheless, as the data is encrypted only once, this overhead can be acceptable in many cases.

**Cover files.** Arianfar et al. [5] described an algorithm to mix legitimate content with so-called "cover files". The content publisher selects a cover content to mix with legitimate content. Cover files are known to both user and the adversary $Adv$. All files are cut in equally-sized blocks and padding is used when necessary. For all $k$ tuples, composed of cover and legitimate blocks, the publisher computes the exclusive-or and publishes the result. For instance, if $k$=2 and given the blocks $c_1$, $c_2$, $l_1$ and $l_2$ ($c$ for cover and $l$ for legitimate), the publisher computes $c_1 \oplus c_2, c_1 \oplus l_1, c_1 \oplus l_2, c_2 \oplus l_1, c_2 \oplus l_2$ and $l_1 \oplus l_2$ and publishes them. Using a secure side channel, the user retrieves meta information, such as the content hash and length in blocks, cover blocks and the algorithm for generating the names of each block. To retrieve the content, the user requests chunks per name and uses belief propagation or Gaussian elimination to reconstruct the original file. However, this technique requires the cooperation of producers who have to generate large amounts of data and xor them. In fact, as pointed out in [5], the publisher must produce all the chunks in advance and thus, must perform $O((\alpha + \beta)^k)$ operations where $\alpha$ represents the number of legitimate blocks, $\beta$ the number of cover blocks, and $k$ the possible number of permutations. Hence, this method incurs several drawbacks that make it unpractical for real-world use.

## 3.3 Name privacy

Name privacy arises from the semantic correlation between human-readable content name and the content itself. Unlike IP, where addresses represent hosts in the network and are not semantically correlated with the content, CCNx [43] names the content itself and routes data based on content names. Unfortunately, such a property creates an imminent privacy threat as the content names are not only visible but also expected to be semantically related to the content itself (e.g., /US/WebMD/AIDS/Symptoms/html). Although this appears similar to an HTTP connection over IP, it is actually more fundamental in CCNx, as content names cannot be encrypted like the URLs in HTTPS connections. Unlike CCNx, some other architectural proposals such as [35, 3] use flat names that are not human-readable. However, all of these proposals rely on a Name Resolution Service (NRS) that performs the translation from human-readable names. Since NRS information is public and accessible to an adversary, even architectures with non human-readable names create the same inherent threat, due to the naming of content pieces.

### *Potential Solutions*

**Bloom filters.** The main name privacy challenge in CONs is keeping the name private while ensuring accessibility and routability. A possible solution is to use Bloom filters [11] to identify content. The resulting architecture would be composed of three main blocks:

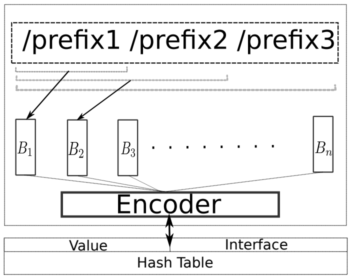

**Figure 2:** Routing Table using hierarchical bloom filter.

1. A hierarchical bloom filter used as the routing table.
2. A counting bloom filter for each interface used as a PIT table [54].
3. A hierarchical bloom filter used as the router storage.

Fig. 2 shows an example architecture for a routing table. Rather than sending the request in the clear for a hierarchically named content, a client would compute the corresponding hierarchical bloom filter as $HB = (B_1, B_2, ..., B_n)$ where $B_i$ is the bloom filter of name components up to the $i$-th component. For instance, when asking for */NYtimes/article/green-econmy*, the client computes a bloom filter $B_1$ of */NYtimes/*, $B_2$ of */NYtimes/article* and $B_3$ of */NYtimes/article/green-econmy*. This scheme would work on CCNx as follows: a router checks whether the last filter ($B_n$ since it contains the exact content name) is in its content store, if so, the content is returned to the customer. If not, the router verifies whether $B_n$ exists in any PIT table. If a match is found, the counting Bloom filter of the corresponding PIT is updated (add one) and the interest is dropped since a request has already been forwarded. Finally, if the interest is not available in the content store nor in any PIT table, the routing process starts: the router checks for longest prefix match on its routing table by starting from $B_n$ going all the way to $B_1$ until it finds a matching routing entry. This approach would enjoy from the obfuscation of content name resulting from transforming it into a random looking string of bits. Also, it can reduce the size needed for storing PIT tables [54], content storage, and routing table, depending on the parameters for the filters and the size of the content name domain. However, Bloom filters introduce false positives and periodically require resetting.

## 3.4 Signature privacy

One of the main goals of CONs is to decouple content from its location and allow retrieving from nearby caches. In order to trust fetched data, some CON architectures, such as, CCNx [43], use digital signatures to provide guarantees on provenance and integrity. Although signatures are powerful tools that bind content to its producer, ordinary digital signatures may leak sensitive identity information about the signer.[1] This is problematic, especially when considering censorship and monitoring, as content from certain publishers can be easily and conveniently identified from the signing key, which is explicitly stated at every data packet in CCNx.

### *Potential Solutions*

**Confirmer signatures.** A first approach to prevent an adversary from verifying a signature would be to use confirmer signatures. Confirmer signatures are undeniable signatures [15] where

[1] CCNx currently offers two choices, RSA and ECDSA, as signature algorithms.

the signer delegates the verification to a third party (the confirmer), thus, signatures cannot be verified without interacting with this party. Using this method, multiple producers may delegate the verification to a third party and increase the anonymity set for publishers. Although this method would be easy to implement [15, 29] and would not require any modification to the CCNx schema, it requires a third party as a confirmer and introduces an another round of communication for signature verification.

**Group signatures [16]** allow the signer to hide in a set of potential signers, thus, providing *signer-ambiguity*. As such, the client can verify that the signature was generated by a member of the group but is unable to tell whom. For instance, a company with multiple employees may use a group signature so that a signature cannot be publicly tracked back to a single user but only to the company. Group signature is efficient since the size of the signature does not depend on the size of the group. However, it assumes the presence of a trusted group manager admitting group members, distributing keys, and revoking anonymity within the group, thus, making it appropriate only in limited settings with collaborating users.

**Ring signatures [45].** As cooperation between users is not always achievable, ring signatures [45] simplify group signatures by removing both group manager and members interaction. Therefore, there is no need to prearrange group of users, nor for special procedures for group management and key distribution. Also, the anonymity of the actual signer is always protected. For instance, a company $X$ could collect the public keys of $n$ other trusted companies $Y_1, Y_2, ..., Y_n$, as these keys are publicly available. Then, $X$ can generate a signature $\sigma$ for content $m$ that keeps the signer unidentifiable among $X$ and other trusted companies. When a client fetches a content $m$, he is able to verify that the content was produced by one of $\{X\} \bigcup \{Y_1, Y_2, ..., Y_n\}$ without knowing which one it is. This schema allows customers to trust content as long as all possible signers are trustworthy, while making anyone observing their traffic unable to singularly discern the signer. However, the communication overhead introduced by ring signatures is linear in the size of the ring. Also, it would still be possible to enforce censorship based on signatures by blocking all content that list certain entities as one of potential signers.

**Ephemeral identities.** Any content producer can create ephemeral keys to sign content. This would effectively prevent identifying the publisher of content by looking at its signature. However, this would also prevent customers from verifying the source/publisher of a content without an additional mechanism to authenticate it. Luckily, CCNx allows creating unforgeable links by including the hash of a content object in the link and allows these links to be signed. This would allow publishing content signed with ephemeral keys that are not traceable to a long-lived identity, but would still allow users to establish transitive trust when they fetched it, by following link that is published by a trusted party. For instance, a link to a sensitive content might be published and signed by a trusted blog ($m_s$) but the actual content ($m_a$) might be published and signed with a one-time indentity and be served through an anonymous hosting service (e.g., rapidshare.com). Any user trusting the blog author can trust $m_s$, and thus ($m_a$), but an eavesdropper observing $m_a$ is unable to link it to its publisher. This approach is very easy to deploy and does not need any modification of the current architecture, however, it cannot hide the access to the link and prevent leakage through its signature.

# 4. THE POTENTIAL OF CON PRIVACY

In this section, we compare CON to today's Internet in the context of a few privacy concepts, such as, anonymity, censoring, traceability, and confidentiality. (A more detailed discussion is included in the full version of this paper [14].)

**Anonymity.** A few techniques are possible today to provide anonymous communications. One naïve solution is to rely on a trusted anonymizing proxy relaying traffic while removing identifying information [10]. However, this introduces a single point of failure and trust, thus, proxy-less techniques have been proposed, relying on *mix networks* [17]. Tor [25] is the best-known, and most widely used, low-latency anonymizing tool. Using onion routing and layered encryption, Tor builds a multi-hop circuit, composed of at least three random nodes.

In CON, proxy-based anonymity could be obtained without the need for an external entity: CON architectures would actually provide, natively, the removal (or substitution) of source and destination addresses. A neighboring CON router could actually be seen as an anonymizing proxy. In reality, however, a local active adversary could monitor all connectivity (see Section 3.1). Regardless, the research community has already started investigating whether or not onion routing-like techniques can be used in CON: a first answer has been provided by AND$\bar{a}$NA [24], a Tor-like low-latency anonymizing tool for CCN [33]. Compared to Tor, it only requires two hops and it is resilient against "hijacking" attacks (i.e., attacks where the adversary hijacks and modifies server's answers e.g., [40]) since data is signed. However, due to data encryption, CON caching mechanism could not be used. Also, AND$\bar{a}$NA inherits some of Tor weaknesses, e.g., the difficulty of circumventing censorship while retrieving the directory with all participating nodes. Moreover, the portability of AND$\bar{a}$NA to architectures other than CCN depends on the routing protocol. For instance, PSIRP [49] routers do not store any routing information and rely on a Forwarding Identifier (FI)[2] provided by the client to route back the content, thus, guaranteeing anonymity is very challenging since the routing protocol itself leaks information. As the FI carries information about the user, it should be anonymized while ensuring that data is correctly forwarded. Whereas, in Dona [35] and NetInf [3], the routing protocol is similar to that used in CCN [33], hence, it is safe to assume that AND$\bar{a}$NA [24] can be used on top of them to provide anonymous communications.

**Censorship.** Internet censorship appears to be easier in CON architectures. First, and foremost, naming content facilitates keyword filtering. Then, as CON routers have bigger computational and memory resources, content blocking could be carried more effectively, without the need for (expensive) dedicated hardware. Finally, data-monitoring is easier since both interests and data are not encrypted. Therefore, an attacker only needs to modify the routing protocol so that any "unwanted" interest is dropped. However, if CON traffic is exchanged using AND$\bar{a}$NA [24], there would be a couple of features making it harder for the attacker to censor content. First, both interests and content are encrypted, and, second, as the exit node is usually out of the attacker's control, the latter cannot delete content. Nonetheless, the effectiveness of AND$\bar{a}$NA (or other techniques in CON) to counter Internet censorship remain an open question that calls for further research.

**Traceability.** Today, a number of techniques are employed to track users, the most widespread of which are cookies. While it seems straightforward to port cookies to CON, this is not without obstacles. In fact, while in current architectures browsers automatically send cookies to Web servers when fetching data, following so-called *same-origin policy*, it is not clear how cookies will be im-

[2] FI is a Bloom filter-based technique used by routers to select the forwarding interface.

plemented for static content in CON, since data can be fetched from anywhere. Cookies could be transmitted to the source only when fetching dynamic data and, as such, cookie-based tracking mechanisms in CON will be less aggressive as only dynamic content can be tracked. Similar arguments apply for Javascript-based tracking, Supercookies, and Evercookies. However, more aggressive tracking techniques have recently emerged. For instance, *Stateless* tracking uses both user IP address and browser fingerprint to uniquely track users on the web [53]. Mitigating browser fingerprinting can be achieved by using plug-ins, e.g., NoScripts, however, it is still hard to hide user's IP address, unless using techniques for anonymous communications. However, as CON architectures remove, by design, both parties' identifiers (i.e., source and destination addresses), it would be harder to actually implement IP-based tracking. The lack of traceability might improve user privacy, however, it naturally raises both security and economic challenges. The former are related to the lack of source address: for instance, following a security incident (e.g. DoS attacks), this information is often crucial to identify/counter the attacker. Whereas, economic challenges stem from changes that CON might impose on the advertisement model: ads are usually delivered based on website popularity and user location and estimating website popularity based on the number of visits is ineffective in CON as caching will "hide" a significant amount of traffic.

Finally, observe that, as discussed in Section 3, a few CON network components (including the role played by neighboring routers) can actually be used to better track users.

**Data authenticity and confidentiality.** While today's Internet requires a "one-size-fits-all" trust model, trust in CON is end-to-end, between data producer and data consumer, and does not depend on any physical or temporal frame. This modularity has two main advantages. (1) Different consumers may easily implement different levels of security, and (2) on CON, one can employ both widely accepted and new trust management models as data is independent from the deployed model. However, these features also prompt some challenges. First, we need to identify a set of usable trust mechanisms that can be deployed and used by most users. Second, as all content is signed, it is crucial to assess (and potentially improve) efficiency of signature generation, transmission, verification, and possibly storage. Finally, we believe that providing data confidentiality while keeping caching mechanism is one of the major open challenges in CON.

## 5. RELATED WORK

Propelled by the increasing interest for next-generation Internet architectures and, in particular, Content-Oriented Networking (CON), the research community has produced a large body of work dealing with CON building blocks [33, 35, 49, 3, 32], performance [54, 46, 19, 44], and scalability [47, 9]. However, the quest for analyzing and enhancing security in CON is only at the beginning – in particular, very little work has focused on privacy and anonymity. In this section, we review relevant prior work.

**Security in CON.** Wong and Nikander [52] address security of naming mechanisms by constructing content name as the concatenation of content provider's ID, cryptographic ID of the content and some meta-data. Dannewitz et al. [22] adopt a similar approach where content name is defined as the concatenation of the hash of the public key and a set of attributes. Both schemes rely on cryptographic hash functions to name the content, which results in a human-unreadable flat naming. Smetters et al. [48] show that these schemes have several drawbacks, including the need for an indirection mechanism to map and the lack of binding between name and producer's identity. To resolve these shortcomings, they propose to keep hierarchical human readable names while signing both content name and the content itself, using producer's public key. Gasti et al. [30] study DoS and DDOS in CCN [33] by presenting attacks and proposing some initial countermeasures. In another context, Burke et al. [12] propose a secure lighting systems over Named-Data Networking (NDN), providing control access to fixtures via authorization policies, coupled with strong authentication.

**Privacy Issues in CON.** To the best of our knowledge, the only related privacy study is the recent article by Lauinger et al. in [38, 39], that covers security and privacy issues of CCN [33]. Specifically, they highlight a few Denial-of-Service (DoS) vulnerabilities as well as different cache-related attacks. From the privacy perspective, work in [38, 39] identifies the issue of information leakage through caches in CCN. It proposes a few simple countermeasures, following *detection* and *prevention* approaches. The former can be achieved using techniques similar to those addressing cache pollution attacks in IP [23], although such an approach can be difficult to port to CON due to the lack of source address. The latter can actually be global, i.e., treating all traffic as sensitive, delaying all traffic, or deploying a shared cache to circumvent the attack. Alternatively, a selective prevention approach may try to distinguish between sensitive and non-sensitive content, based on content popularity and context (time, location), and then delay or tunnel sensitive content. Our work extends that in [38, 39] by encompassing *all* privacy aspects: caching, naming, signature, and content. Also, it is more general as it does not only consider CCN [33], but CON in general, independently of the specific instantiation. Furthermore, when suggesting countermeasures, we only propose techniques that can be applied with a minimal change to the architecture.

**Anonymity in CON.** AND$\bar{a}$NA [24] proposes a Tor-like anonymizing tool for CCN [33] to provide provable anonymity. It also aims to privacy protection via simple tunneling. However, AND$\bar{a}$NA is an "all-in-one" solution that introduces latency and impedes caching. Whereas, fine-grained privacy solutions are needed, since a widespread use of tunneling would inherently take away most of CON benefits in terms of performance and scalability.

## 6. CONCLUSION

Content-Oriented Networking (CON) proposes a major transition from today's Internet to a new content-based architecture. This radical change calls for a thorough analysis of both security and privacy guarantees. CON comes with a potential benefit to security, including a security-by-design approach based on digital signatures that provides data integrity and origin authentication, as well as trust support. However, prior to our work, it remained an open question whether or not, and to what extent, this emerging networking paradigm bears new privacy threats.

This paper presented a first-of-its-kind, systematic analysis of privacy issues in CON as a generic paradigm, discussing different attacks and detailing their impact on user privacy. We also proposed several countermeasures while attempting to balance the trade-off between privacy, performance, and changes to the architecture. In the process, we identified a number of interesting research challenges that call for further work in the area.

Naturally, our work does not end here: first, and foremost, we are working toward further evaluating the feasibility of our proposed countermeasures and their effective deployment; next, we plan to provide an in-depth study of multiple encryption and signature techniques and their impact on network performance; finally, we intend to analyze the impact of privacy-enhancing and CON-native technologies on Web economy and advertisement models.

# 7. REFERENCES

[1] G. Acs, M. Conti, P. Gasti, C. Ghali, and G. Tsudik. Cache Privacy in Named-Data Networking. In *ICDCS*, 2013.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of Information-Centric Networking. *IEEE Communications Magazine*, 50(7), 2012.

[3] M. Ambrosio, M. Marchisio, V. Vercellone, and et al. Second NetInf Architecture Description. 4WARD Deliverable D6.2, http://www.4ward-project.eu/index.php?id=192, 2010.

[4] J. Ardelius, B. Grönvall, L. Westberg, and A. Arvidsson. On the effects of caching in access aggregation networks. In *ICN*, 2012.

[5] S. Arianfar, T. Koponen, B. Raghavan, and S. Shenker. On preserving privacy in Content-Oriented Networks. In *ICN*, 2011.

[6] R. Bendrath and M. Mueller. The End of the Net as We Know It? Deep Packet Inspection and Internet Governance. *New Media and Society*, 13(7), 2011.

[7] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, 1998.

[8] D. Boneh, C. Gentry, and B. Waters. Collusion-resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *CRYPTO*, 2005.

[9] S. Borst, V. Gupta, and A. Walid. Distributed Caching Algorithms for Content Distribution Networks. In *INFOCOM*, 2010.

[10] J. Boyan. The Anonymizer – Protecting User Privacy on the Web, 1997.

[11] A. Broder, M. Mitzenmacher, and A. Broder. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 1, 2002.

[12] J. Burke, P. Gasti, N. Nathan, and G. Tsudik. Securing Instrumented Environments over Content-Centric Networking: the Case of Lighting Control. In *NOMEN*, 2013.

[13] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A Routing Scheme for Content-Based Networking. In *INFOCOM*, 2004.

[14] A. Chaabane, E. De Cristofaro, M. Kaafar, and E. Uzun. Privacy in Content-Oriented Networking: Threats and Countermeasures. http://arxiv.org/abs/1211.5183, 2013.

[15] D. Chaum. Desinated-confirmer signature systems, 1994. US Patent 5,373,558.

[16] D. Chaum and E. Van Heyst. Group signatures. In *EUROCRYPT*, 1991.

[17] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), 1981.

[18] K. Cho, J. Choi, D. Ko, T. Kwon, and Y. Choi. Content-oriented networking as a future internet infrastructure: Concepts, strengths, and application scenarios. In *Future Internet Technologies*, 2008.

[19] K. Cho, M. Lee, K. Park, T. Kwon, Y. Choi, and S. Pack. WAVE: Popularity-based and collaborative in-network caching for Content-Oriented Networks. In *INFOCOM Workshops*, 2012.

[20] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi. A Survey on Content-Oriented Networking for Efficient Content Delivery. *IEEE Communications Magazine*, 49(3), 2011.

[21] Cisco, Inc. Cisco Visual Networking Index: Forecast and Methodology, 2011–2016. http://preview.tinyurl.com/3p7v28, 2012.

[22] C. Dannewitz, J. Golic, B. Ohlman, and B. Ahlgren. Secure Naming for a Network of Information. In *INFOCOM Workshops*, 2010.

[23] L. Deng, Y. Gao, Y. Chen, and A. Kuzmanovic. Pollution attacks and defenses for internet caching systems. *Comput. Netw.*, 2008.

[24] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun. Andana: Anonymous named data networking application. In *NDSS*, 2012.

[25] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Usenix Security*, 2004.

[26] T. Duong and J. Rizzo. Here come the ⊕ ninjas. In *Ekoparty Security Conference*, 2011.

[27] E. W. Felten and M. A. Schneider. Timing Attacks on Web Privacy. In *CCS*, 2000.

[28] A. Fiat and M. Naor. Broadcast Encryption. In *CRYPTO*, 1994.

[29] S. Galbraith and W. Mao. Invisibility and anonymity of undeniable and confirmer signatures. In *CT-RSA*, 2003.

[30] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. DoS and DDoS in Named-Data Networking. In *ICCCN (To Appear)*, 2013.

[31] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov. The most dangerous code in the world: validating SSL certificates in non-browser software. In *CCS*, 2012.

[32] M. Gritter and D. R. Cheriton. An Architecture for Content Routing Support in the Internet. In *USITS*, 2001.

[33] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In *CoNEXT*, 2009.

[34] G. Keizer. Hackers may have stolen over 200 SSL certificates. http://www.computerworld.com/s/article/9219663/Hackers_may_have_stolen_over_200_SSL_certificates, 2011.

[35] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A Data-Oriented (and beyond) Network Architecture. *SIGCOMM Computer Communication Review*, 37(4), 2007.

[36] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: an architecture for global-scale persistent storage. *SIGPLAN Notes*, 35(11), 2000.

[37] N. Laoutaris, S. Syntila, and I. Stavrakakis. Meta Algorithms for Hierarchical Web Caches. In *IPCCC*, 2004.

[38] T. Lauinger, N. Laoutaris, P. Rodriguez, T. Strufe, E. Biersack, and E. Kirda. Privacy Implications of Ubiquitous Caching in Named Data Networking Architectures. Technical report, TR-iSecLab-0812-001, iSecLab, 2012.

[39] T. Lauinger, N. Laoutaris, P. Rodriguez, T. Strufe, E. Biersack, and E. Kirda. Privacy risks in named data networking: what is the cost of performance? *SIGCOMM Computer Communications Review*, October 2012.

[40] S. Le Blond, P. Manils, A. Chaabane, M. A. Kaafar, C. Castelluccia, A. Legout, and W. Dabbous. One bad apple spoils the bunch: exploiting P2P applications to trace and profile Tor users. In *LEET*, 2011.

[41] National Science Foundation. NSF Future Internet Architecture Project. http://www.nets-fia.net/, 2010.

[42] Palo Alto Research Center, Inc. Project CCNx: Interest Message. http://www.ccnx.org/releases/latest/doc/technical/InterestMessage.html, 2012.

[43] Palo Alto Research Center, Inc. Project CCNx: Open-Source Implementation and Documentation. http://www.ccnx.org/, 2012.

[44] I. Psaras, W. K. Chai, and G. Pavlou. Probabilistic in-network caching for information-centric networks. In *ICN*, 2012.

[45] R. L. Rivest, A. Shamir, and Y. Tauman. How to Leak a Secret. In *ASIACRYPT*, 2001.

[46] G. Rossini and D. Rossi. A dive into the caching performance of content centric networking. In *CAMAD*, 2012.

[47] G. Rossini and D. Rossi. On sizing CCN content stores by exploiting topological information. In *NOMEN*, 2012.

[48] D. Smetters and V. Jacobson. Securing Network Content. Technical Report, www.parc.com/content/attachments/securing-network-content-tr.pdf, 2009.

[49] K. Visala, D. Lagutin, and S. Tarkoma. LANES: An Inter-Domain Data-Oriented Routing Architecture. In *ReArch*, 2009.

[50] D. Wessels. Configuring Hierarchical Squid Caches. http://old.squid-cache.org/Doc/Hierarchy-Tutorial/tutorial-1.html.

[51] D. Wessels and K. Claffy. Internet Caching Protocol – RFC2186. http://tools.ietf.org/html/rfc2186, 1997.

[52] W. Wong and P. Nikander. Secure Naming in Information-Centric Networks. In *ReARCH*, 2010.

[53] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi. Host fingerprinting and tracking on the web: Privacy and security implications. In *NDSS*, 2012.

[54] W. You, B. Mathieu, P. Truong, J.-F. Peltier, and G. Simon. Realistic Storage of Pending Requests in Content-Centric Network Routers. In *ICC*, 2012.