A content-delivery protocol, exploiting the privacy benefits of coded caching

Felix Engelmann and Petros Elia

***Abstract*—Coded caching is a communications technique that has elevated the preemptive use of memory (caching) into a powerful ingredient in general communications networks, promising to change the way networking and PHY-based communications are conducted. At the same time though — because this approach is heavily dependent on cooperation between the content provider (CP), and a centralized powerful transmitter of information (ISP), and because it is heavily dependent on users caching a variety of content that is not their own — raises privacy concerns which have the potential to compromise the applicability of coded caching. What we are showing in this early work here, is that in fact coded caching carries a distinct set of salient features that in fact boost privacy. We present a step-by-step privacy-aware content-delivery protocol that utilizes caching and which — at a small cost in performance — can safeguard against unauthorized matching of users to their requests, as well as against unauthorized knowledge of the popularity statistics of files; both crucial privacy issues in different scenarios such as video on demand. These properties include multicasting-only transmissions for continuous obfuscation of the true destination of content, an almost seamless addition of phantom users that can skew the true popularity distribution, popularity-agnostic caches, cache-agnostic ISP, and an overall minimization of data traffic between CP and ISP, and between ISP and users.**

I. Introduction & System Model

The efficient delivery of modern data over communication networks, often requires that multiple parties cooperate. This cooperation – in addition to any performance gains in terms of speed and reliability – also inadvertently introduces the phenomenon that each cooperating party can glean information on the statistics, destination, and nature of the communicated data. This naturally raises many security concerns relating to secrecy and privacy.

In terms of data secrecy, the main goal is to guarantee that only the entities which are authorised to view the content are able to do so. In terms of privacy, there are two well known concerns that we consider here. The first has to do with individual privacy and the fact that the association of a request for content to a specific user, automatically allows for the unauthorized tracking of user behaviour. Hence, the goal should be that only the entity authorising the request should be allowed to know the associated user that made that request. The second privacy concern relates to the unauthorized tracking of the statistics of the file requests themselves. Knowledge of such global statistics (of the requested/transmitted files) can be very profitable, and the goal is to effectively hide these file popularity statistics from unauthorized entities.

The authors are with the Communication Systems Department at EURECOM, Sophia Antipolis, 06410, France (email: elia@eurecom.fr) and the Institute of Distributed Systems, University of Ulm, 89081 Ulm, Germany (email: felix.engelmann@uni-ulm.de). The work is supported by the European Research Council under the European Unions Horizon 2020 research and innovation program (Agreement no. 725929).

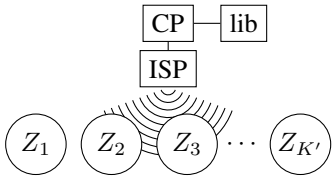

Fig. 1. Simple scenario where a content provider (CP) with an attached library, employs a minimalistic service provider (ISP) to serve content to its end users

Without any concern for performance, such secrecy and privacy concerns can be directly handled by utilizing https/TLS techniques which apply data- and user-dependent keys to completely counter any ability to draw conclusions on the nature of content and about the requests of different users. This though automatically counters gains from cooperation, even of the most simple form like multicasting where — in the presence of https — one could not take advantage of the fact that two or more users end up requesting the same information. Such secrecy, privacy and performance considerations become crucial in various content distribution scenarios.

a) System model: In this context, we here consider a setting where a content provider (CP) has a library of N files (e.g., a library consisting of N popular movies) $W'_1, \dots, W'_N$, each of size $|W'_n| = F$ bits, and where the CP is attached to an internet service provider (ISP) to deliver the files to its users (see Fig. 1). This cooperation introduces efficiency as the ISP (transmitter) has storage to replicate the original library closer to the users. The ISP serves K' users, where each user $k \in \{1, 2, \dots, K'\} \triangleq [K']$ is equipped – again for purposes of increasing communication efficiency – with a cache $Z_k, k \in [K']$ of size MF bits which can fit a fraction $\gamma \triangleq \frac{M}{N}$ of the library. Such efficiency gains can be achieved by having each user k carefully cache – well before the delivery of content – some fraction $|Z_k \cap W'_n|/|W'_n|$ of each library file W'_n.

This scenario nicely captures the conflict between performance and security. While the CP relies on the ISP to efficiently deliver content as well as relies on the users to cache some of this content, the performance gains of cooperation (between the CP, ISP and users) must be reaped without requiring any trust in the ISP nor any trust in the neighbouring users, as these users could be colluding with the ISP.

b) Basic characteristics of coded caching: Given our emphasis here on performance, we will naturally consider

978-3-901882-90-6/17 ©2017 IFIP

coded-caching solutions [1] which provide a paradigm shift in the way caches are utilized, and which render the use of caches at the end-users, meaningful. Let us first give some background on coded caching.

Coded caching was proposed by Maddah Ali and Niesen in the seminal work [1] as a technique which — after carefully caching content at the receivers, and properly coding across different users' requested data — provided increased effective throughput and a reduced network load. The proposed solution was motivated by the fact that wireless traffic is heavily video on-demand (over 60%), which entails an ability to predict data requests in advance ("the night before"). This approach was based on caching content from an existing library of many popular files. Each user would pre-store (without knowing the next day's requests) a carefully selected sequence of sub-files from the library, specifically designed to speed up (next day's) communications. Essentially memory is used to disseminate side-information, which can be best utilized by multicasting, i.e., by transmitting signals that must be 'heard' by many.

Based on this insight, coded caching allowes to deliver a scaling number of files with delay that increases only marginally with the number of users, and which quickly converges to a constant delay upper bound, irrespective of the number of users. While we will see more of the details later on, it is very important to note that our proposed solution, which is based on adding extra users, does not impose a substantial cost in the performance (capacity requirements, or speed) of the system, and it never exceeds the above fixed bound, irrespective of the number of (phantom) users that we will add. Hence part of what this paper shows is how — using coded caching — allows us to reap the privacy benefits of having many users' messages combined, with almost no cost in performance. We believe that currently, no such solution comes close to providing such a combination of privacy and performance gains.

c) Related work on secrecy and caches: In current Content Distribution Networks (CDN) the caches must be trusted by the users and can collect detailed statistics about the unicast connections. Therefore they are managed by the CP, while located at the ISP. This is a problematic situation, as physical access by the ISP can circumvent any security measures. A more evolved approach is Content Centric Networking (CCN) where caches are natural nodes forwarding data to multiple requesters. Unfortunately these nodes can trace content and create statistics, undermining the privacy of the requests. Similarely, Leguay *et al.* [2] propose encrypted objects in caches at the edge of the network with bandwidth intensive re-encryption operations to counter linkability of requests for the same content.

d) Related work on coded caching: This work in [1] has inspired a sequence of other works (cf. [3], [4], [5], [6], [7], [8]) which explore the limits of coded caching under different assumptions. Additional interesting works include [9], [10], [11], [12], [13], [14], [15], [5], [16], [17] as well as other work such as [18] on the cache-aided erasure broadcast channel, the works in [19], [20] on the wireless interference channel with transmitter-side caching, and our work in [21].

e) Related work on coded caching and secrecy: With the understanding that coded caching has the potential to offer much needed and unprecedented performance-gains in modern content-distribution scenarios, recent work has sought to explore different secrecy-related aspects of coded caching. The first thing to note is that, as stated above, coded caching manipulates data in order to convert a mainly-unicast set of transmissions into a single-shot multicast transmission that serves many users at a time, even if these users have requested different content. This efficient utilization of coded multicasting, automatically excludes the full use of https.

In terms of secrecy-related aspects of coded caching, the work in [22] analyzes the secure caching problem and introduces a scheme that guarantees secrecy (i.e., guarantees inability to access unauthorized *content*) at a cost that becomes negligible – in information theoretic terms – as the number of users and files increases. This is accompanied by a proof that, even for smaller (feasible) parameters, the delivery delay of the proposed scheme comes within a constant multiplicative factor from the information-theoretic optimal delay. Another interesting related work can be found in [23] which provides an information-theoretic approach for achieving perfect secrecy between users with the help of a secret threshold sharing, that guarantees that each user can decode their requested file only.

We here place an additional focus on privacy, which can be breached for a variety of reasons, ranging from having more than one user requesting the same file (an event which can be easily detected at the ISP, who could employ the use of colluding users to identify the content requested by non-colluding users), as well as reasons relating to the fact that data is cached throughout various users' caches. Generally privacy issues arise as a result of the cooperation between the CP, the ISP and the users, and these issues are critical because the information about what users consume, has very high value in video-on-demand applications.

II. Privacy drawbacks of cache-aided unicast, and performance drawbacks of https

The above-mentioned security issues in the absence of https, are directly associated to, and aggravated by, the common unicast method typically used by ISPs to communicate with the users, as well as by the common methods typically used to exploit caches at the receivers.

1) Unicast transmissions — where the ISP is asked to send specific content to specific users — naturally disclose the destination of content, thus automatically allowing association of a data pseudo-label[1] to the requesting user, which in turn can reveal the true label of the request of that user. This last step (from the pseudo-label to the real label of the requested file of a specific user) can be achieved simply by having an active set of (virtual) malicious users colluding with the ISP, and requesting each file in the library just once, to get the

[1]Here we make the distinction between the 'label' of a file (e.g. the actual file name, such as the name of a movie), and the 'pseudo-label' of a file which is a secretly scrambled version of the label.

pseudo-label. The ISP, who already knows (due to the use of unicast) which pseudo-label each user has requested, can now deduce the requested labels for a number of users. Hence, in the absence of perfect pseudo-label obfuscation, we will take the worst-case approach and assume that finding a real file name (true label) from the pseudo-label, is feasible.

2) The second problem, of allowing — in this case the ISP — to gain unauthorized access to the file popularity statistics, is further aggravated by the typical 'data-push' uses of caches which must reflect the true popularity statistics of the pseudo-labels and thus of files.

Such traditional methods — where the locally available part of a request is directly retrieved from the cache of the requesting user, while the rest is transmitted via unicast — simply forward data closer to the users during off peak hours in order to reduce the volume of transmission during peak hours. Such methods though, yield local caching gains which suffer severely if caching fails to properly account for, and exploit, the popularity statistics[2]. Hence, in such cases, caches must reflect the true popularity statistics which means that a colluding user can — directly from their own cache — deduce the histogram of the pseudo-labels and thus to a certain extent the histogram (popularity statistics) of the labels themselves.

3) On the other hand, the third problem relates to performance. Known TLS privacy-preserving solutions such as https, currently place a very large burden on the communication links, by limiting cooperation. For example, as we mentioned before, https is not known to allow for efficient multicasting, because the keys are both user- and data-dependent, which means that the same content will be encrypted into entirely different sequences of bits which cannot share a communications link. This limitation becomes even more damaging in coded caching, which would loose almost all achieved gains as it is entirely built around coded multicasting. In fact, https does not allow efficient use of caches, even in the uncoded case because it limits local caching gains by not exploiting common requests, and because it places a very large burden on the placement phase by having to distribute a total of KMF bits across the caches, rather than the total of NF bits of the library.

Furthermore, in the absence of https, attempts to continuously scramble the file names (i.e., to continuously generate new pseudo-labels) require updating also the associated content it self (updating the ISP library), which can place some burden on the link from the CP to the ISP. In our case, this burden becomes very substantial because we would have to continuously update all the users' caches as well, placing additional burden on the ISP-to-users links.

[2]For example, using popularity-agnostic (local)-caching approaches that employ caches that reflect a uniform distribution (rather than the true, often geometric, distribution), would reduce the total duration for handling all K' users' requests, only by a factor of $(1-\gamma)$, compared to unicast transmission that would not use caches. This gain can be very small for typical values of γ. For instance, when $\gamma \approx 0.01$, such popularity-agnostic (local)-caching approaches would result in a delay (or needed bandwidth) reduction of only 1%.

III. Identifying catalytic ingredients for efficient privacy-aware cache-aided communications

The above privacy drawbacks, bring to the fore a set of ingredients/properties that facilitate privacy in high-performance cache-aided communications.

Property 1. Multicast-only transmissions. Unlike unicast, multicast can have the advantage that each transmission is simultaneously meant for many users at a time. In the presence of many such users, such multicasting can make it harder for the ISP to associate content to users.

Property 2. Popularity-agnostic caches. The goal is to be able to derive high caching gains that are robust, even to the most severe deviation between the true statistics and those observed in the caches, which would thus reveal little about the true popularity statistics (of even the pseudo-labels).

Property 3. Cache-agnostic ISP. Furthermore, although a bit premature at this point, an additional property that facilitates privacy-aware cache-aided communications, is having the ISP – who in fact places the content during the cache-placement phase – be oblivious of the content in each user's cache, and also be unaware of the pseudo-labels in each user's cache. This will eventually guarantee, as we will see later on, that even though the ISP knows the pseudo-labels of the transmitted data during the delivery phase, it cannot associate users and requests.

Property 4. Vanishing cost of additional user load. Additionally we seek a method that can conceivably handle an increasing number of users, with only a minimal additional delay (required bandwidth) cost. As we will see, this property will allow us to give the CP the opportunity to use (virtual) phantom users to defend against an attack for deriving the true popularity statistics. The privacy usefulness of this property will be clarified later on.

Property 5. Minimization of traffic between CP and ISP, and of traffic between ISP and users. Finally — corresponding to the third aforementioned problem — the above properties should yield a method that minimizes the traffic load between CP and ISP (backhaul) and minimizes the load between the users and the ISP. This last requirement is important as it reflects a high received throughput of the users, and it becomes particularly crucial in wireless content delivery, where the wireless link capacity is the main bottleneck.

Such a protocol which satisfies all four first properties, which requires very small increases in the CP to ISP traffic load, and which is approximately optimal (in information-theoretic terms) over the last ISP-to-users link, will be presented here, and will employ the decentralised coded caching method in [4].

IV. Secrecy-and-privacy-aware coded-caching protocol

We here describe the sequence of steps of the content-delivery protocol, identifying how the aforementioned ingredients/properties are exploited to achieve obfuscation of users' requests and of popularity statistics.

1) The CP sends the library, encrypted, to the ISP. To achieve secrecy in our setting, we use a basic encryption approach with keys that are file-based (rather than file- and user-based). In our case, each file W'_n is symmetrically encrypted with a key E_n to $W_n = f_{E_n}(W'_n)$ where f is the reversible encryption function (e.g. AES/OFB) which does not reveal any information on the plain-text with partial cipher-text. The keys E_n are randomly initialised and securely stored on the server. Key rotation can be performed by re-encrypting the files under the new key and re-distribute the newly encrypted files. Therefore the caches for this file must be invalidated. For a general introduction to cryptographic schemes, see Schneier [24].

2) Each user participating in the scheme must be authorized by the CP. Authentication can be performed over a secured channel by e.g. a token or credentials. Based on this channel, a secret parameter S_k is exchanged between each user k and the CP. This will be used as a seed for the pseudo-random algorithm that will fill-up each user's cache. The seed can be changed as necessary, as long as the CP and the users known which seed was used to cache a specific file.

3) The ISP broadcasts the encrypted library, and each user (based on their seed S_k) caches a certain part of the encrypted content. The pseudo-random nature of the cache-placement algorithm — which can be a secure pseudo random number generator whose output defines which users should cache a sub-packet — keeps us in linewith the idea of distributed coded caching [4]. Hence each cache Z_k is filled with encrypted content, and the contents (the pseudo labels, and thus the labels) of the content in each Z_k, is unknown to all except to user k and the CP. On average the cache is equally occupied by all the files, i.e., that

$$\mathbb{E}(|Z_k \cap W_n|/|W_n|) = \gamma, \ \forall n \in [N]. \tag{1}$$

This satisfies our required *Property 2*.

4) The users send, over secure communications with the CP, directly to the CP, their requests. Each individual request is denoted as r'_k so each user requests file $W_{r'_k}$. All requests r'_k are received at the CP, who combines and shuffles them (using a secret permutation only known to the CP) to form $\mathbf{r}'$.

5) The CP introduces phantom users. While the number of real users is K', the CP can decide to introduce L additional phantom users, and encode as if there were more users $K = K' + L$, each with their own request. The CP introduces a set of phantom requests $r'_{K'+1}, \ldots, r'_K$, which now are appended to the original (permuted) vector of requests $\mathbf{r}'$, to create a request vector $\mathbf{r} \triangleq [\mathbf{r}', r'_{K'} + 1, \ldots, r'_K]$, which is designed to reflect a different file popularity distribution, and to obfuscate the original popularity distribution. By appending the phantom requests to the end of $\mathbf{r}'$ they could be detected and removed. Shuffling $\mathbf{r}$ again removes the possibility to decide if a request is real or phantom.

The introduction of the phantom users will also help to defend against attacks meant to associate real users to their requests (i.e., to associate users to the pseudo-labels of the files they requested) in cases where only a very small number of real users request data (in which case, the obfuscation of the requests could have been weak). As we describe further down, this latter obfuscation (relating to associating users to file pseudo-labels) will be achieved because increasing K will automatically (as a result of the structure of coded caching) increase the degree of multicasting, i.e., will increase the number of users ($K\gamma+1$) that are instantaneously served by a single transmission. In the end, as we also show further down, serving the phantom users will cause very modest additional communication load, which in fact diminishes as K' and K increase.

6) The CP sends transmission instructions to the ISP. These instructions are meta-data that describe *how to create* each linear combination (each XOR). For each XOR, this meta-data simply describes the pseudo-labels of the sub-files that are to be xored together. These labels are of substantially smaller volume than the actual data (this applies toward satisfying *Property 5*). The sequence of instructions (one instruction per XOR), which we denote here by X', is calculated at the CP using function ψ^{sec} which takes as input all the $r'_k, k = 1, \ldots, K$, and all the keys (S_k) which describe the structure (not the data) of each Z_k, and the structure of the library. ψ^{sec} does not require any actual data, and it reflects the coded caching algorithm in [4].

7) The ISP follows the instructions X' and applies it to its copied library to create the actual XORs (the actual data). This can be done without knowledge of the cache-structure at the ISP (thus fulfilling *Property 3*). Then the ISP proceeds to transmit the sequence of all such XORs (data), which we denote as X. Each XOR is designed (as in [4]) to serve, on average, $K\gamma + 1$ users at a time (*Property 1*). As mentioned above, the addition of the phantom users (in addition to skewing the statistics) increases the multicasting degree from $K'\gamma + 1$ to $K\gamma + 1$, thus increasing the elements (pseudo-labels) in each linear combination, which makes it harder for the ISP to associate pseudo-labels to users[3].

8) At this point, the ISP also broadcasts the meta data X', which allows each user to know what is in the XOR intended for them. This does not affect privacy or secrecy as we designed X' so that it does not contain any useful information. Looking at X', the users and the ISP will simply be able to conclude the corresponding request-histogram (of pseudo-labels) which though reflects (by design) the skewed statistics which — by adding phantom users with specific requests — deviate substantially from the true popularity statistics. Furthermore X' does not reveal an association of pseudo-labels to users; only in combination with the other S_ks, can anyone associate requests to users. As each user k only knows their own S_k, they will only deduce their own r'_k, which they already know.

[3]This avoids attacks where, for example, all but one (or very few) of the real users are colluding with the ISP, in which case the ISP could have de-constructed the XOR (by taking out of the linear combination, the requests of its own colluding users) thus associating a pseudo-label to the remaining real user.

9) Each user sees each transmission, which contains a linear combination of different subfiles, as well as the meta information that describes the pseudo-labels of the files that were xored together. If a user sees a useful subfile inside a received XOR, and has all other sub-packets that form that XOR, the user can decode the packet; with knowledge of metadata X', each user k can follow the exact decoding procedure that characterizes decentralized coded caching, and can combine X and Z_k (under the instructions given in X', and as a function of r'_k) to decode and recover all missing sub-files of their requested file $W_{r'_k}$.

Finally to get the plain-text $W'_{r'_k}$, the user can decrypt the file after retrieving the file key $E_{r'_n}$ from the CP. The CP can thereby control the access to the plaintext.

For increasing file sizes F, the fraction of the meta-data X' is very small compared to X, which implies that the delay is dominated (more and more, as F increases) by the cost of communicating data. In accordance to [4], we normalise T so that $T = 1$ corresponds to the delay required to deliver a single file to a single cache-free user, without interference. We recall that in the presence of uncoded caching with uniform cache placement, the delay indefinitely increases with K as $T = K(1-\gamma)$. In comparison, for coded caching, directly from [4], we know that the entire transmission delay takes the form $T_{cc} = K(1-\gamma)\frac{1-(1-\gamma)^K}{K\gamma}$ which very quickly, as K increases, converges to $T_{cc} \to \frac{1-\gamma}{\gamma}$. This shows — in alignment with *Property 4* — the robustness of the performance to the introduction of the phantom users, especially when $K\gamma > 1$, in the sense that the addition of a user only increases the required delay by a decreasing factor $T_{cc}(K+1) - T_{cc}(K)$ which continuously decreases as K increases, and which eventually vanishes to zero for all $\gamma \in (0, 1)$.

V. Performance Evaluation: cost of obfuscating a Zipf popularity distribution

Although in theory, the introduction of the phantom users does not have a substantial effect on the performance of the coded caching algorithm under ideal conditions, this increased number of users results (due to increased subpacketization) in an increase in the corresponding file size [15] which can be a limiting factor. For this reason, it might be preferable to split the library in two parts; the first part with the $\tilde{n}$ most popular files $W_1, \dots, W_{\tilde{n}}$, and the second part consisting of the remaining more rarely queried ones $W_{\tilde{n}+1}, \dots, W_N$ corresponding to the tail of the popularity distribution. The proposed solution here, would then apply coded caching only to the popular part of the library, while the tail would always be delivered by classical secure communications directly from the CP. This would also be the solution of choice for $K' < 1/\gamma$, in which case the gains of coded caching are limited.

Let us assume a Zipf popularity distribution with parameter $\alpha \in (0, 1)$, and let us recall that the distribution's normalisation factor is approximately

$$\sum_{k=1}^{N} \frac{1}{k^\alpha} \approx \int_{k=1}^{N} \frac{1}{k^\alpha} = \frac{N^{1-\alpha}+1}{1-\alpha}. \tag{2}$$

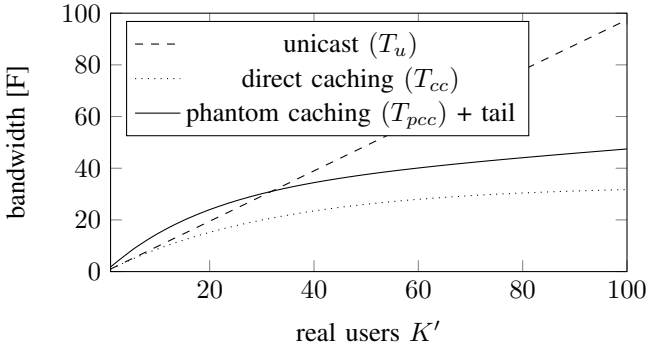

Fig. 2. Performance of our new scheme with $\gamma = \frac{3}{100}, \alpha = 0.8$ and $\tilde{n} = 30$ out of $N = 100$.

Given that the number of real requests is K', the most popular file will be (on average) requested $p(1) = K'\frac{1-\alpha}{N^{1-\alpha}+1}$ times. Also given that we cache only from the first $\tilde{n}$ files, and that the total number of requests served by coded caching, including phanton requests, is K, we have that

$$K = p(1) \cdot \tilde{n} = K'\frac{1-\alpha}{N^{1-\alpha}+1}\tilde{n}. \tag{3}$$

For the files in the tail, this average number of requests takes the form $K'\left(1 - \frac{\tilde{n}^{1-\alpha}+1}{N^{1-\alpha}+1}\right)$ hence the corresponding total delay (required bandwidth) takes the form

$$T_{pcc} = K(1-\gamma)\frac{1-(1-\gamma)^K}{K\gamma} + K'\left(1 - \frac{\tilde{n}^{1-\alpha}+1}{N^{1-\alpha}+1}\right) \tag{4}$$

The delivery can always fall back to point-to-point unicast transmissions with a local caching gain for the first $\tilde{n}$ files and achieve a total delay of

$$T_u = K'\frac{\tilde{n}^{1-\alpha}+1}{N^{1-\alpha}+1}(1-\gamma) + K'\left(1 - \frac{\tilde{n}^{1-\alpha}+1}{N^{1-\alpha}+1}\right) \tag{5}$$

As N,M and γ are fixed, an optimal $\tilde{n}$ can be calculated to minimise the number of real users $\hat{K}'$ in $T_{pcc} = T_u$ from which on our scheme outperforms end-to-end transmission. This is shown in Figure 2, where we see that for the specific example, end-to-end transmission is better for up to around 30 users and from there on our scheme provides a linear gain.

VI. Conclusions

This work is motivated by the fact that the distribution of large amounts of modern content over communication networks, must adhere to strict privacy requirements, while maintaining a reduced traffic between the participating parties. Towards this, we here leverage decentralised coded caching in a way that preserves much reduced communication cost while guaranteeing privacy and secrecy. The solution here does not require any additional cache updates at the ISP or the receivers, and does not require any need for substantial additional data exchange between the CP and the ISP.

It is to note that the introduction of the phantom users increases the computational complexity for computing X' to be in the order of $\mathcal{O}(K \cdot 2^K)$. These benefits relate to the multicasting nature of coded caching, and its coding structure. Multicasting allows us to obfuscate the true destination of content, while coding allows for an almost seamless addition of phantom users that can skew the true popularity distribution, eventually protecting the information on each user's behavior as well as on global popularity statistics – all despite the fact that the majority of data is stored and processed close to the user, inside an untrusted network provider.

This contribution comes at a time when coded caching is elevating the preemptive use of memory (caching) into a powerful ingredient in general communications networks, promising to change the way networking and PHY-based communications are conducted. At the same time though — because this approach is heavily dependent on cooperation between the content provider, and a centralized powerful transmitter of information (ISP), and because it is heavily dependent on users caching a variety of content that is not their own — this approach raises privacy concerns which have the potential to compromise the gains of coded caching. What we are showing in this early work here, is that in fact coded caching carries a distinct set of salient features that boost privacy. The here presented utilization of coded caching (in the form of a privacy-aware communication protocol) shows that — at a small and decreasing cost in performance — we can safeguard against unauthorized matching of users to their requests, as well as against unauthorized knowledge of the popularity statistics of files; both crucial privacy issues in different scenarios such as video on demand.

A. Future directions

A possible direction of future research is to test the system for robustness against a variety of attacks, such as for example an attack where the ISP maliciously misassembles XORs to test reactions which are bound to come only from real users. This of course brings to the fore the interesting question of how much QoS the ISP is willing (and can afford) to sacrifice, to possibly gain partial and sporadic insight on the real popularity distribution. This direction also brings to the fore the need for possible modifications of the proposed protocol here, including message authentication of X', that defend against this attack.

Another direction relates to cooperation between users. In our current model, the CP is responsible for keeping the r'_k secret and for combining them to a scrambled $\mathbf{r}'$. An interesting research topic could be the design of a cryptographic method where the users cooperatively decide (in a distributed manner) on $\mathbf{r}'$ without disclosing their own request.

Another research direction is to explore how coded caching maintains these privacy benefits, in settings where coded caching is fused with modern, feedback-aided interference management schemes that are heavily dependent on signal separation which, in some sense, is the opposite of multicasting which was identified here as a beneficial ingredient for privacy.

REFERENCES

[1] M. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[2] J. Leguay, G. S. Paschos, E. Quaglia, and B. Smyth, "Cryptocache: Network caching with confidentiality," May 2017, *IEEE ICC 2017*, available at: http://jeremie.leguay.free.fr/files/CryptoCache-ICC2017.pdf.

[3] S. Wang, W. Li, X. Tian, and H. Liu, "Fundamental limits of heterogenous cache," *CoRR*, vol. abs/1504.01123, 2015. [Online]. Available: http://arxiv.org/abs/1504.01123

[4] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029–1040, Aug 2015.

[5] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order optimal coded delivery and caching: Multiple groupcast index coding," *CoRR*, vol. abs/1402.4572, 2014. [Online]. Available: http://arxiv.org/abs/1402.4572

[6] H. Ghasemi and A. Ramamoorthy, "Improved lower bounds for coded caching," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, June 2015, pp. 1696–1700.

[7] C. Wang, S. H. Lim, and M. Gastpar, "Information-theoretic caching: Sequential coding for computing," *IEEE Trans. Inf. Theory*, vol. 62, no. 11, pp. 6393–6406, Nov 2016.

[8] A. N., N. S. Prem, V. M. Prabhakaran, and R. Vaze, "Critical database size for effective caching," in *2015 Twenty First National Conference on Communications (NCC)*, Feb 2015, pp. 1–6.

[9] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *INFOCOM, 2012 Proceedings IEEE*, 2012.

[10] B. Perabathini, E. Bastug, M. Kountouris, M. Debbah, and A. Conte, "Caching at the edge: a green perspective for 5G networks," in *IEEE Int. Conf. on Communication Workshop (ICCW)*, June 2015, pp. 2830–2835.

[11] U. Niesen, D. Shah, and G. W. Wornell, "Caching in wireless networks," *IEEE Trans. Inf. Theory*, vol. 58, no. 10, pp. 6524–6540, Oct 2012.

[12] E. Bastug, M. Bennis, and M. Debbah, "A transfer learning approach for cache-enabled wireless networks," in *2015 13th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2015, pp. 161–166.

[13] J. Hachem, N. Karamchandani, and S. N. Diggavi, "Coded caching for heterogeneous wireless networks with multi-level access," *CoRR*, vol. abs/1404.6560, 2014. [Online]. Available: http://arxiv.org/abs/1404.6560

[14] J. Hachem, N. Karamchandani, and S. Diggavi, "Effect of number of users in multi-level coded caching," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Hong-Kong, China, 2015.

[15] K. Shanmugam, M. Ji, A. Tulino, J. Llorca, and A. Dimakis, "Finite length analysis of caching-aided coded multicasting," 2015, submitted to *IEEE Trans. Inform. Theory - July 2015*.

[16] M. Deghel, E. Bastug, M. Assaad, and M. Debbah, "On the benefits of edge caching for MIMO interference alignment," in *Signal Processing Advances in Wireless Communications (SPAWC), 2015 IEEE 16th International Workshop on*, June 2015, pp. 655–659.

[17] R. Timo and M. A. Wigger, "Joint cache-channel coding over erasure broadcast channels," in *2015 International Symposium on Wireless Communication Systems (ISWCS)*, Aug 2015, pp. 201–205.

[18] A. Ghorbel, M. Kobayashi, and S. Yang, "Cache-enabled broadcast packet erasure channels with state feedback," in *Proc. Allerton Conf. Communication, Control and Computing*, Sept 2015, pp. 1446–1453.

[19] M. A. Maddah-Ali and U. Niesen, "Cache-aided interference channels," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT'2015)*, Hong-Kong, China, 2015.

[20] F. Xu, K. Liu, and M. Tao, "Cooperative tx/rx caching in interference channels: A storage-latency tradeoff study," in *2016 IEEE International Symposium on Information Theory (ISIT)*, July 2016, pp. 2034–2038.

[21] J. Zhang, F. Engelmann, and P. Elia, "Coded caching for reducing CSIT-feedback in wireless communications," in *Proc. Allerton Conf. Communication, Control and Computing*, Illinois, USA, Sep. 2015.

[22] A. Sengupta, R. Tandon, and T. C. Clancy, "Fundamental limits of caching with secure delivery," in *2014 IEEE International Conference on Communications Workshops (ICC)*, June 2014, pp. 771–776.

[23] V. Ravindrakumar, P. Panda, N. Karamchandani, and V. Prabhakaran, "Fundamental limits of secretive coded caching," in *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 425–429.

[24] B. Schneier, *Applied cryptography : protocols, algorithms, and source code in C*. New York: Wiley, 1994. [Online]. Available: http://opac.inria.fr/record=b1084163