Summary - Scalable and Private Media Consumption with Popcorn

Bowen Song[1]
November 18, 2019

Abstract—**This summary describes my understanding of [1] with some discussions over the existing works and technique that Popcorn used.**

I. Introduction

Popcorn is a media delivery system that protects consumer privacy relying on *Private Information Retrieval* (PIR) primitives. It targets to hide media requests from eavesdroppers and content distributors. The seemingly innocent media content request may expose user's sexuality, political point of view, social status, and more. All of which can be misused for discriminatory service/charges or even targeted attacks. The challenge is to maintain affordability (IO cost, required number of server replicas, network bandwidth, etc) when at scale and respect controls on content dissemination.

Popcorn integrates the techniques from two existing works CPIR [2] and ITPIR [3] with modifications to scale up the protocol and extends privacy protection. The ITPIR is a scheme that requires mirror servers for the full library of media content and entrusting that these servers would not collude with each other. For each media request, the actual request is obfuscated by adding random other requests and XORing the content of the media from these requests as one response from each server. The obfuscation appends similar random requests for different servers so that the XOR sum of all requests would leave the desired media content. Depending on the number of random requests added with respect to the size of the entire library, this method can only hide the content of media requests under a small number. In the long term, user's predilection for certain content is still deductible if similar content requests are consistently selected. Unless the random requests added follow a consistent trend that covers the entire span of media types. Popcorn introduces a method to bach requests from different users which would solve this problem.

The CPIR is another scheme while successfully hides content request from distributors, sufferers a linear computation overhead over the size of the library and require uniform object size. The query encodes the quest as if it is requesting for the entire media library and the server response is generated in a similar manner. While this method does not leak the request and response content information, it requires a large amount of computing power and network bandwidth which would hardly scale in terms of either the number of requests or the size of the media library.

[1]B. Song was with Department of Electrical and Computer Engineering, Boston University, Boston MA, `sbowen@bu.edu` and is now a Software Developer with Red Hat - OpenShift.

II. Algorithm Overview and Discussion

Popcorn uses the idea of ITPIR for encrypted media content so that the control over content dissemination can be respected. The encryption key for each media content is protected by the CPIR scheme which is far more controllable since the key size is uniform and the size of each key is far shorter than any media content. Moreover, a group of media may even share a key to reduce the key library size.

To amortize the overhead costs and add more natural obfuscation to users' requests, Popcorn batches multiple requests together for same-sized chunks of media content. The larger batches the better for obfuscating requests and lowering costs. However, combing requests expects time synchronization in content consumption for each chunk. Popcorn proposes to use smaller chunks for the head of media content and increase chunk size for later part of the media which the request is not imminent and can afford longer time sync. However, this strategy may sacrifice privacy since small batches can be targeted to extract user requests. There is no need to target larger batches because attackers may just need information about what movie is consumed. Instead, the situation may just be lifted with more users and batches can be picked from users who start to watch the content at the same time.

At last, Popcorn proposes to serve the content of different sizes by changing bit-rate and splitting movies that are too large. This approach seems to be ok since most movies are in similar length. However, this is to assume users do not exhibit a similar trend to certain movies, for example, stopping the streaming after a certain amount of time, jump through, or playback at similar times. These traits could again expose the content of the media.

III. Detailed Analysis

The following analysis targets detailed core designs of *Popcorn* that are not analyzed in the high-level discussion.

A. Architecture

One detail of Popcorn architecture is that all media objects are split into segments of various sizes but the same decomposition. The segments are further divided into 1 MB slices as work units so that upon each request, servers can compute an array of 1 MB size XORed encoded media responses queued up in storage for severing the client in the appropriate playback time.

B. Batching

To reduce I/O cost, Popcorn handles requests in batches, looping through slices of the entire library in time series. This requires Popcorn to have the entire library available on each server since batched requests can spread out to the entire library and query content is designed to be unknown.

Batching also requires synchronization, of clients. The pyramid column size change helps with the synchronization by buffering requests using the previous column playtime. There are details about the actual segmenting size and both the paper and my instinct went for the simple doubling method as the segmenting gets closer to the end.

C. Machine Provision

The machine provisioning while not account for fluctuations for I/O, CPU, and network, also does not account for failures. The chance of failure exists and for the time-sensitive protocol that affects a *cohort*, set of clients, for each failure, a backup plan should exist and be accounted to the actual cost.

D. Failure Mode

Popcorn does not consider the failure mode of using the XOR method. Although movies are generally considered unique in their content, if Popcorn is moving away from controlled content, there could be collisions between library content and XOR cancels added content if it exists. In Netflix, There are also "extended version" movies that has mostly the same content with some extensions to the original one. With same content included in a request, Popcorn XORing such content will fail to deliver the media.

IV. Algorithm Performance Analysis

Against other protocols, Popcorn scales better with more concurrent requests and longer media content. Overall it is more affordable than other schemes included in the results.

V. Conclusion

Overall, Popcorn is considered as a scalable scheme for conserving privacy in a sizeable media content delivery system. The cost is amortized and thrives with more concurrent users. However, It can be difficult to entrust content servers not colluding with each other by just using different administrative domains, since service logs can be traded revealing the hidden privacy content. Although included as one of the weaknesses pointed out by the paper, targeted commercials may be one of the reasons for users to choose private streaming. Since private streaming feature does require more dollar cost, this feature can be part of premium account privilege and targeted commercial service may fall out of scope.

VI. Possible Future Work

Popcorn requires different servers to maintain content privacy; however, it does not take advantage of parallel downloading. Media is split into segments, these segments can be asynchronously downloaded by any pair of non-colluding servers. This way, Popcorn can lower the burden on servers and scale further with larger libraries. Popcorn acknowledges its scalability limitations to providers like YouTube due to library size. However, YouTube still serves content based on regional information. While the Media properties vary, they can still be categorized and bundled together to obscure media queries. Properties such as video length alone can partition the YouTube library down to subsets of smaller libraries tangible for Popcorn. An important note is that the length of the video may leave room for inference, as any added heuristics may open a door for side channels.

Moreover, Popcorn does not account for geo-distributed media content retrievals. Popcorn does not mention any distributed content scenarios where libraries are partitioned on different servers. In the smaller scale scenario, Popcorn handles multiple machines in one server for a single IT-PIR response by going through all content on machines. This might not be very efficient for geo-distributed content. Therefore, a piggyback way for batch adding and subtracting media content between servers maybe a future work to scale popcorn up to process larger media libraries.

Popcorn hides the content of media query and response, however, does not consider obfuscate metadata of the viewer. For example, the viewer watching shows at 1 AM in the morning is still going to stand out and information can still be interpreted. The frequency of media requests and so on can be used to determine the viewer's personality, line of profession, and more which can be used to further deduce the viewer's private information. A further step for this work could try to see if masking metadata is also possible.

References

[1] T. Gupta, N. Crooks, W. Mulhern, S. Setty, L. Alvisi, and M. Walfish, "Scalable and private media consumption with popcorn," in *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, 2016, pp. 91–107.

[2] E. Kushilevitz and R. Ostrovsky, "Replication is not needed: Single database, computationally-private information retrieval," in *Proceedings 38th Annual Symposium on Foundations of Computer Science*. IEEE, 1997, pp. 364–373.

[3] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE, 1995, pp. 41–50.