# Summary - Scalable and Private Media Consumption with Popcorn

Bowen Song[1]
November 4, 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.

## 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. Possible Enhancement Design

Popcorn requires different servers to maintain content privacy, however, does not take the 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 varies, they can still be categorized and bundled

[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.

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 video may leave room for inference, as any added heretic may open a door for side channel.

## 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

As foolproof as Popcorn's design is, it fails to account for geo-distributed media content retrievals. We can still distinguish people by their country of origin among other information. Popcorn failed to mention any distributed content scenarios where libraries are partitioned on different servers. Intentional partitions by the provider can directly reveal important information about the customers. Therefore, a piggyback way for adding and subtracting media content between servers maybe a future work to scale popcorn up to process larger media libraries.

Another Popcorn limitation is due to its use of ITPIR. This scheme requires at least two non-colluding servers. However, this problem of deterministic retrieve elements from a set of elements can be easily done by using an IBLT [4]. The IBLT is a data structure based on the bloom filter. It stores key-value pairs of a set of data, in this case, *key* is the query and *value* is the media content. The server can condense a set of query-media value into a "peel-able" IBLT which can be tested locally. It sends the IBLT to the client-side. The advantage of the client is that it knows which query it is actually interested in. After peeling through the IBLT, the client would be able to consume the media content. This strategy adds additional computation requirement and network bandwidth cost. Nevertheless, the same data can be responded to concurrent users which means the set of query-media value can be the response to a batch of different consumer queries transmitted over a gossip protocol.

## 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.

[4] M. T. Goodrich and M. Mitzenmacher, "Invertible bloom lookup tables," in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2011, pp. 792–799.