A Privacy Attack on Coded Caching by Colluding Users

Yu Wang and Alhussein A. Abouzeid
ECSE Department
Rensselaer Polytechnic Institute
Troy, NY, USA
wangy52@rpi.edu, abouzeid@ecse.rpi.edu

***Abstract*—Coded caching is beneficial due to its ability to leverage the multicast property of a wireless channel. Under specific network settings, the coded caching scheme has been shown to reveal the user file requests. In our prior work, a privacy attack by a single attacker on coded caching was presented. In this paper, we focus on a relatively secure setting whereby only the intended users access the file. Nevertheless, we show that, even in this setting, privacy is compromised if the users are colluding. The time complexity of the attack on a network with N files, K users, and $l \leq K$ colluding users is shown to be in the order of $\sqrt{\ln(N(K-l))}N^{(K-l)/2}$ steps.**

***Index Terms*—Coded caching, privacy, attack.**

I. Introduction

Due to the periodicity of user daily activities, network traffic has peak periods and off-peak periods. Caching a file during off-peak periods and serving users who request the file during peak periods is an effective technique to smooth the peak period traffic. The reduction of transmission rate due to the usage of the user local cache is regarded as *local gain* [1]. *Uncoded caching* [2], [3], a scheme in which the server sends uncoded files to users, is suitable for applications where the channel connecting the server and the user is unicast such that each user can be treated individually.

In a multicast channel setting, *coded caching* introduced by Maddah-Ali and Niesen in [1] (referred to as the MAN scheme) achieves additional *global gain* by leveraging the multicast property of the channel. The MAN scheme has two phases, a *placement phase* and a *delivery phase*. During the placement phase, each file is split into multiple non-overlapping subfiles, and each user cache is filled with carefully chosen subfiles selected by the server. During the delivery phase, users send their file demands to the server, and then the server transmits linear combinations (bitwise exclusive or) of subfiles to satisfy the user demands. By jointly optimizing the cache fulfilling strategy and the generation of the server transmission, one coded subfile in the server transmission may benefit multiple users. Many variations of this scheme have been proposed to deal with decentralized networks [4], heterogeneous cache sizes [5], arbitrary file popularities [6], each user demanding multiple files [7], and D2D networks [8]–[10].

Some aspects of the privacy leakage issue of MAN have been noticed by some prior work. In [11], the authors show that one user can predict the other user's file request in a two-user network by exclusion. Because the attacker cannot identify who requested a file, the method of exclusion cannot be generalized to schemes with more non-colluding users. Two coded caching schemes presented in [12], [13] consider the additional privacy constraint that no user gains information about any other user's file request. The idea of limiting the number of subpackets and whereby the achieved privacy is studied for a two-user-two-file case in [14]. However, only a few works studied how general the privacy leakage problem exists in coded caching scheme, and how potential attacker(s) benefit from the leakage to breach the system privacy. In our prior work [15], we proposed a three-stage attack for an eavesdropper to acquire the file requests of all users. In this paper, we consider two significant generalizations of the attack scenario. First, we add the constraint that subfiles can only be accessed by the designated users. Second, we consider the case where the users are colluding. The first assumption reduces the possibility of breaching the system privacy. Nevertheless, we show that the system privacy can still be breached by the colluding attackers, and a new attack is presented. The main contributions of this paper are summarized as follows:

- We present a two-stage attack whereby colluding users can predict all other non-colluding users' file requests. The attack assumes that the attackers access is limited to only the designated subfiles in the server transmission.
- We provide an analytical approximation of the attack time.

The remainder of the paper is organized as follows. We describe the coded caching scheme and assumptions on attackers in Sec. II. In Sec. III, we state a necessary condition of a privacy attack and then describe the proposed attack. Then, we analyze the attack time and provide an example. We conclude our work in Sec. IV.

II. Network and Attacker Models

In this section, we formulate the coded caching problem in the presence of colluding users. We summarize the MAN scheme, present the assumptions, and describe the attacker model.

A. MAN Coded Caching Scheme

In the MAN scheme, there is a server holding N files, each of length F bits. The server connects to K users, each has

a local cache of size MF bits, where M is the normalized cache size. The server sends data to users through a multicast channel, while each user sends its data via a unicast secure and private channel. The network structure is illustrated in Fig. 1.

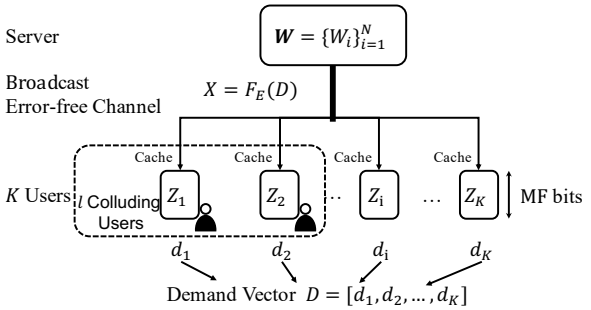

Fig. 1: One central server with N files connects K users using an error-free broadcast channel. Each user has a local cache of size MF bits. There are l colluding users who would like to predict any other user's demand from the information they have access to.

The MAN scheme consists of a placement phase and a delivery phase.

1) Placement Phase: During the placement phase, each of the N files is split into $\binom{K}{t}$ non-overlapping subfiles, where $t = MK/N$, and denote W_n as the set containing all subfiles of file $n \in [N] \triangleq \{1, 2, \cdots, N\}$. We assume t here to be an integer for simplicity. If t is a rational number, the trick of memory-sharing described in [1] can be applied.

The server decides which user to cache which subfile, and the cache of each user is filled in a manner that each subfile is cached by t users. After the placement phase, let Z_u denote the set of subfiles that are cached by user u. We use $\overline{Z_u}$ to denote the complement set of Z_u.

The placement phase is assumed to have no privacy or security issue, and each user only knows its own cache content. Since the placement phase is often conducted during off-peak periods, the secure and private delivery of contents can be achieved via a secure unicast channel.

2) Delivery Phase: During the delivery phase, each user requests one file index from the server. We assume that each file is equally likely to be requested, and the file selection is independent of each other. Note that, file popularity only affects the attack time, and the uniform file popularity is considered by the MAN scheme [1]. Denote the file index sent by user k by d_k. The demand vector $D \in [N]^K$ is defined as

$$D = [d_1, d_2, \cdots, d_K]. \tag{1}$$

The server generates coded subfile X_i for users whose indexes are in group S_i following

$$X_i = \oplus_{s \in S_i} W_{d_s, S_i \setminus \{s\}}. \tag{2}$$

S_i is a set containing $t+1$ user indexes. $W_{d_s, S_i \setminus \{s\}}$ represents the subfile which is in the cache memory of users whose indexes are in $S_i \setminus \{s\}$. $\oplus$ is the operator of bitwise exclusive or. Each user shows up in $\binom{K-1}{t}$ groups, and there are in all $\binom{K}{t+1}$ different groups.

The server transmission sequence is serialized from a set X containing coded subfiles for all groups, i.e.

$$X = \{X_1, X_2, \cdots, X_{\binom{K}{t+1}}\}. \tag{3}$$

In the following discussion, we refer to set X as server transmission sequence for simplicity. We do not assume the serialization of subfiles following any order, i.e., X_i is not necessary the i^{th} subfile in X. We use encoding function $F_E(\cdot)$ to characterize the mapping from demand vector to server transmission, i.e. $X = F_E(D)$.

One implementation of the MAN scheme is shown in Sec. III-D, which later servers as the target scheme to be attacked.

3) Our Target Scheme and Attacker Model: In this paper, we add the constraint that each subfile is limited to be accessed only by its designated users. Such constraint can be satisfied by encrypting each subfile with a group-share key as [16]. The attackers are l colluding users who would like to acquire the demands of the other $K - l$ non-colluding users for any server transmission. Same as [17], we assume that attackers share their cache contents and accessible subfiles during the attack.

Because of the access constraint, $\binom{K-l}{t+1}$ subfiles in a server transmission are never accessible to any of the l attackers. Hence, the three-stage attack in [15] that requires full knowledge of the server transmission cannot be directly applied to colluding users.

With limited access to the server transmission, but stronger assumptions on attackers, we study whether the colluding users can still breach the system privacy and the corresponding attack time.

III. Privacy Attack from Colluding Users

In this section, we characterize the subfile-level one-to-one mapping that makes a privacy attack on the MAN scheme possible. After that, a two-stage attack algorithm is proposed and analyzed.

We only consider cases when $l < K-1$. When $l = K$, there is no need for an attack (i.e., no non-colluding user). When $l = K - 1$, the demand of the only non-colluding user can be identified by exclusion, then the attack becomes trivial. Note that, when $t \leq l < K - 1$, each non-colluding users is in at least one group that contains t colluding users. The demand of the non-colluding user can be identified by exclusion in such a group. However, attackers cannot identify who requested which file using exclusion, which means our present attack is still needed.

A. Necessary Conditions of A Privacy Attack

We first confirm in Theorem 1 that there exists a lookup table that maps X to D. We then describe how to construct the table to perform a privacy attack sequentially.

Let D_{S_i} denote the demand vector containing only the demands of users in S_i.

Theorem 1. *For a given Z, X_i and D_{S_i} are one-to-one mapped.*

Proof. The proof consists of two parts.

1. One D_{S_i} associates with exactly one X_i. Since X_i is generated following (2), One D_{S_i} associates with exactly one X_i.

2. One X_i associates with one D_{S_i}. Since subfile X_i is generated for group S_i, there is no coded subfile in X that is not intended for any group. Suppose there exist two different demand vectors D_{S_i} and $D_{S_i'}$, both associating to X_i. Then, X_i is decodable by users in $S_i \cup S_i'$. Let user $k \in S_i' \setminus S_i$. To decode X_i, k must cache t out of the $t+1$ uncoded subfiles that are used to generate X_i, which means at least one uncoded subfile is cached by $t+1 > t$ users, i.e., t users from S_i and user k, which contradicts the placement phase rule. □

According to Theorem 1, once X_i is observed, the demand vector of users in S_i must be D_{S_i}. It then becomes obvious that if the attacker somehow knows the corresponding X_i for any D_{S_i}, during the attack, it can reversely lookup D_{S_i} from X_i. Since X_i is generated by following (2), the attack turns to acquire the cache content of non-colluding users to apply (2).

Let U denote the index set of all colluding users. Let X^U denote the set of coded subfiles in X that are intended for at least one user in U. All $\binom{K}{t+1} - \binom{K-l}{t+1}$ subfiles in X^U are accessible to attackers. We call X^U and X'^U a pair of neighbor sequences for U if their corresponding demand vector D and D' differ in one user file request, and the user is not in U.

Theorem 2. *Demand vector D and D' are neighbor demand vectors that differ in user k's file request if and only if the number of coded subfiles that differ in X^U and X'^U is*

$$\begin{cases} \binom{K-1}{t} - \binom{K-l-1}{t} & \text{if } k \notin U, \\ \binom{K-1}{t} & \text{if } k \in U. \end{cases} \tag{4}$$

Proof. When $k \in U$, the statement is the same as Theorem 2 in [15]. We focus only on when $k \notin U$.

Necessity: Among the total $\binom{K-1}{t}$ user groups that include k, $\binom{K-l-1}{t}$ groups do not contain users in U. Therefore, $\binom{K-1}{t} - \binom{K-l-1}{t}$ subfiles are decodable by at least one user in U.

Sufficiency: Suppose X^U and X'^U differ in $\binom{K-1}{t} - \binom{K-l-1}{t}$ subfiles, but D and D' differ in $s \geq 2$ user file requests. Let k' be another user whose file request differs in D and D'. There are two situations for k':

1) If $k' \in U$, X^U and X'^U differ in at least $\binom{K-1}{t} \neq \binom{K-1}{t} - \binom{K-l-1}{t}$ subfiles, which contradicts the given number. Thus, at most one of the s users is in U.
2) If all s users are not from U, the number of subfiles X^U and X'^U differ in is

$$\underbrace{\left(\binom{K}{t+1} - \binom{K-s}{t+1}\right)}_{(a)} - \underbrace{\left(\binom{K-l}{t+1} - \binom{K-l-s}{t+1}\right)}_{(b)}, \tag{5}$$

where (a) is the number of groups containing at least one of the s users; (b) is the number of groups containing at least one of the s users, but none from U. Since the term $-\binom{K-s}{t+1} + \binom{K-l-s}{t+1}$ is a decreasing function of s, only when $s = 1$ satisfies the given number $\binom{K-1}{t}$, which contradicts $s \geq 2$.

Since both situations are impossible, k' does not exist, which proves the theorem. □

Proposition 1. *Consider attacker set U, a pair of neighbor sequences X^U and X'^U where user $k \notin U$ requests file d_k in X^U and d_k' in X'^U. Attackers are able to deduce the following cache content that are not in Z_k*

$$(W_{d_k} \cup W_{d_k'}) \cap (\cup_{u \in U} Z_u) \cap \overline{Z_k}. \tag{6}$$

Proof. Consider a group S_i that contains user k and at least one attacker u from U, the corresponding coded subfile $X_i \in X$ and $X_i' \in X'$ differ only in the subfile that is from W_{d_k}, which implies the subfiles that are cached by attacker u but not by user k, i.e.

$$(W_{d_k} \cup W_{d_k'}) \cap Z_u \cap \overline{Z_k}. \tag{7}$$

By combining the results from all attackers in U, the attackers know $(W_{d_k} \cup W_{d_k'}) \cap (\cup_{u \in U} Z_u) \cap \overline{Z_k}$. □

Proposition 2. *The cache contents of any $l \leq t$ users share $\binom{K-l}{t-l}$ subfiles from a file in common.*

Proof. In the placement phase, the subfile allocation process can be viewed as letting all possible combinations of t users cache the same subfile. For $l \leq t$ users, they all show up in $\binom{K-l}{t-l}$ different groups, so they share $\binom{K-l}{t-l}$ subfiles in common. □

Corollary 1. *Consider a set of l users and two users k and k'. The following two situations cannot happen simultaneously:*

1) *both k and k' are not in the set of l users;*
2) *there exists a file n such that both k and k' cache the same $\binom{K-l-1}{t-l-1}$ subfiles of file W_n as any other users in the set.*

Proof. Assume there exist user k and k' who both cache the same set of $\binom{K-l}{t-l}$ subfiles with all $l-1$ users in the set for file n. Then, there exists a larger set of $l+1$ users, including k, k', and all $l-1$ users in the set, in which all users cache the same $\binom{K-l}{t-l}$ subfile of W_n. This set violates Proposition 2. Hence, the assumption is invalid. □

Corollary 2. *If user k and k' share the same $\binom{K-l}{t-l}$ subfiles of one file with another set of $l-1$ users, $k = k'$.*

Corollary 2 is a straightforward application of Corollary 1. When l attackers predict part of the cache content about a user from multiple pairs of neighbor sequences, Corollary 2 helps to link prediction results from multiple neighbor sequences to the same user.

B. Two-stage Attack Algorithm from Colluding Users

In this section, we present the two-stage attack for colluding users. Colluding users are in set U, and they would like to predict the other $K-l$ non-colluding user file requests.

Stage 1: Predict the cache content of each user. All users in U keep collecting server transmission sequences and filter out coded subfiles intended for them. By applying Theorem 2

to each pair of sequences in their collection, users in U know whether two sequences are a pair of neighbor sequences. By applying Proposition 1 to each pair of neighbor sequences, users in U gain some knowledge about the cache content of a user not in U. At the end of Stage 1, users in U know

$$(\cup_{n\in[N]}W_n)\cap(\cup_{u\in U}Z_u)\cap\overline{Z_k},\forall k\in[K]. \tag{8}$$

Stage 2: Link D_{S_i} to X_{S_i}. After knowing (8), users in U are able to build a lookup table between D_{S_i} and X_{S_i} for any group S_i containing at least one user from U. By checking the corresponding D_{S_i} for each intended coded subfile in a newly received X, users in U know any other users' file requests.

C. Complexity Analysis of the Attack

To complete Stage 1 of the proposed attack in Sec. III-B, attackers need to collect neighbor sequences contributing to the estimation of other $K-l$ non-colluding users for all N files. The effect is equivalent to attacking a coded subfile scheme with $K-l$ users and N files. When file popularity follows uniform distribution, the expected number of server transmission sequences that need to be collected is

$$E_X\approx\sqrt{\ln N(K-l)}N^{(K-l)/2}. \tag{9}$$

The prediction of the whole set $\{Z_k\}_{k\in[K]}$ is achieved by predicting $Z_k\cap W_n$ for each $k\in[K]$ and $n\in[N]$. Then, the prediction of $\{Z_k\}_{k\in[K]}$ can be viewed as a set-covering problem.

Theorem 3. *Consider a set S having $|S|$ elements. For each experiment, one element is chosen from S with replacement, and every element is equal-likely to be chosen. The expected number of experiments to conduct until all elements are seen is approximated by $|S|\ln|S|$ for a sufficiently large $|S|$.*

Proof. Let C_m denote the number of experiments that still need to be conducted when m distinct elements are unseen. Obviously, $\mathbb{E}[C_0]=0$. The relationship between $\mathbb{E}[C_m]$ and $\mathbb{E}[C_{m-1}]$ is

$$\mathbb{E}[C_m]=1+\frac{m}{|S|}\mathbb{E}[C_{m-1}]+\frac{|S|-m}{|S|}\mathbb{E}[C_m]. \tag{10}$$

Simplifying (10) and then recursively applying the formula between $\mathbb{E}[C_m]$ and $\mathbb{E}[C_{m-1}]$, we have

$$\begin{aligned}\mathbb{E}[C_{|S|}]=&\mathbb{E}[C_0]+\sum_{m=1}^{|S|}(\mathbb{E}[C_m]-\mathbb{E}[C_{m-1}])\\ =&0+\sum_{m=1}^{|S|}\frac{|S|}{m}\overset{(*)}{\approx}\int_1^{|S|}\frac{|S|}{m}dm=|S|\ln|S|.\end{aligned} \tag{11}$$

In step (*), we use integration to approximate summation. □

Back to the complexity analysis. Let S be the overall $(N-1)K$ predictions about $\{Z_k\}_{k\in[K]}$. By applying Theorem 3, the expected number of collections to reveal all pieces of information in S is

$$(N-1)K\ln\big((N-1)K\big). \tag{12}$$

Among all $N^K\cdot N^K=N^{2K}$ different pairs of server transmission sequences (order matters), $K(N-1)N^K$ of them are neighbor sequences. The probability that a randomly selected pair of sequences is neighbor sequences is

$$K(N-1)/N^K. \tag{13}$$

The expected number of pairs of sequences to cover S is (12) divided by (13), which is

$$N^K\ln\big((N-1)K\big)\approx N^K\ln(NK). \tag{14}$$

Thus, the expected number of sequences to collect to complete Stage 2 E_X is

$$E_X=\sqrt{\ln(NK)}N^{K/2}. \tag{15}$$

D. An example attack from colluding users

TABLE I: Cache content for scheme $K=5$, $N=3$ and $t=3$. The cache pattern is the same for file B and C. "x" means the subfile is cached.

User	File A									
	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
1	x	x	x	x	x	x				
2	x	x	x				x	x	x	
3	x			x	x		x	x		x
4		x		x		x	x		x	x
5			x		x	x		x	x	x

In this section, we use an example to show how colluding users apply the proposed attack in Sec. III-B to deduce all other user file requests.

The example scheme has $K=5$, $N=3$, $M=1.8$, and $t=MK/N=3$. The three files are labeled as A, B, and C for simple notations. The cache content of all five users is depicted in Table I. Follow the placement phase in Sec. II-A, each file is split into $\binom{K}{t}=10$ subfiles, and each subfile is cached by $t=3$ users. Part of the ground truth X-D table is shown in Table II. The demand vectors and their server transmissions are labeled in lexicographic order from $X001$ to $X243$. For instance, let's consider demand vector $D=[A,A,A,A,B]$. For the group of $t+1=4$ users which has user 1, 2, 3, and 4, each subfile in set $\{A1,A2,A4,A7\}$ is cached by three users except the other one. Following (2), the server sends $A1\oplus A2\oplus A4\oplus A7$.

As can be seen from Table II, subfiles for group S_5 in $X002$, $X083$ and $X164$ are all $A8\oplus A9\oplus A10\oplus B7$, because the file demands for user 2, 3, 4, 5 are $[A,A,A,B]$. This is a verification of Theorem 1. In this example, we consider attacks performed by colluding user 1, 2, and 3, and the goal is to predict file requests from users 4 and 5. For simplicity, we assume all attackers always request file A.

In Stage 1, all three attackers keep collecting server transmission sequences. Since $l=t$, d_4 can be read by attackers from S_1, and d_5 from S_2. But the attackers may not link d_4 with S_1 because we assume all X_is in X are ordered randomly. Suppose neighbor sequences $X001$ and $X002$ are received, attackers perform the following predictions:

1) From X_1 in $X001$, one user needs $A1$ and has $\{A2,A4,A7\}$. From X_2 in $X001$, another user needs $A1$ and has $\{A3,A5,A8\}$.

TABLE II: Part of the ground truth X-D table for coded caching scheme with $K = 5$, $N = 3$ and $t = 3$.

Index	D	$S_1 = (1,2,3,4)$	$S_2 = (1,2,3,5)$	$S_3 = (1,2,4,5)$	$S_4 = (1,3,4,5)$	$S_5 = (2,3,4,5)$
		X_1	X_2	X_3	X_4	X_5
X001	[A,A,A,A,A]	$A1 \oplus A2 \oplus A4 \oplus A7$	$A1 \oplus A3 \oplus A5 \oplus A8$	$A2 \oplus A3 \oplus A6 \oplus A9$	$A4 \oplus A5 \oplus A6 \oplus A10$	$A7 \oplus A8 \oplus A9 \oplus A10$
X002	[A,A,A,A,B]	$A1 \oplus A2 \oplus A4 \oplus A7$	$A3 \oplus A5 \oplus A8 \oplus B1$	$A3 \oplus A6 \oplus A9 \oplus B2$	$A5 \oplus A6 \oplus A10 \oplus B4$	$A8 \oplus A9 \oplus A10 \oplus B7$
X003	[A,A,A,A,C]	$A1 \oplus A2 \oplus A4 \oplus A7$	$A3 \oplus A5 \oplus A8 \oplus C1$	$A3 \oplus A6 \oplus A9 \oplus C2$	$A5 \oplus A6 \oplus A10 \oplus C4$	$A8 \oplus A9 \oplus A10 \oplus C7$
X004	[A,A,A,B,A]	$A2 \oplus A4 \oplus A7 \oplus B1$	$A1 \oplus A3 \oplus A5 \oplus A8$	$A2 \oplus A6 \oplus A9 \oplus B3$	$A4 \oplus A6 \oplus A10 \oplus B5$	$A7 \oplus A9 \oplus A10 \oplus B8$
X005	[A,A,A,B,B]	$A2 \oplus A4 \oplus A7 \oplus B1$	$A3 \oplus A5 \oplus A8 \oplus B1$	$A6 \oplus A9 \oplus B2 \oplus B3$	$A6 \oplus A10 \oplus B4 \oplus B5$	$A9 \oplus A10 \oplus B7 \oplus B8$
X006	[A,A,A,B,C]	$A2 \oplus A4 \oplus A7 \oplus B1$	$A3 \oplus A5 \oplus A8 \oplus C1$	$A6 \oplus A9 \oplus B3 \oplus C2$	$A6 \oplus A10 \oplus B5 \oplus C4$	$A9 \oplus A10 \oplus B8 \oplus C7$
X007	[A,A,A,C,A]	$A2 \oplus A4 \oplus A7 \oplus C1$	$A1 \oplus A3 \oplus A5 \oplus A8$	$A2 \oplus A6 \oplus A9 \oplus C3$	$A4 \oplus A6 \oplus A10 \oplus C5$	$A7 \oplus A9 \oplus A10 \oplus C8$
X010	[A,A,B,A,A]	$A1 \oplus A4 \oplus A7 \oplus B2$	$A1 \oplus A5 \oplus A8 \oplus B3$	$A2 \oplus A3 \oplus A6 \oplus A9$	$A4 \oplus A5 \oplus A10 \oplus B6$	$A7 \oplus A8 \oplus A10 \oplus B9$
X019	[A,A,C,A,A]	$A1 \oplus A4 \oplus A7 \oplus C2$	$A1 \oplus A5 \oplus A8 \oplus C3$	$A2 \oplus A3 \oplus A6 \oplus A9$	$A4 \oplus A5 \oplus A10 \oplus C6$	$A7 \oplus A8 \oplus A10 \oplus C9$
X028	[A,B,A,A,A]	$A1 \oplus A2 \oplus A7 \oplus B4$	$A1 \oplus A3 \oplus A8 \oplus B5$	$A2 \oplus A3 \oplus A9 \oplus B6$	$A4 \oplus A5 \oplus A6 \oplus A10$	$A7 \oplus A8 \oplus A9 \oplus B10$
X055	[A,C,A,A,A]	$A1 \oplus A2 \oplus A7 \oplus C4$	$A1 \oplus A3 \oplus A8 \oplus C5$	$A2 \oplus A3 \oplus A9 \oplus C6$	$A4 \oplus A5 \oplus A6 \oplus A10$	$A7 \oplus A8 \oplus A9 \oplus C10$
X083	[B,A,A,A,B]	$A1 \oplus A2 \oplus A4 \oplus B7$	$A3 \oplus A5 \oplus B1 \oplus B8$	$A3 \oplus A6 \oplus B2 \oplus B9$	$A5 \oplus A6 \oplus B4 \oplus B10$	$A8 \oplus A9 \oplus A10 \oplus B7$
X164	[C,A,A,A,B]	$A1 \oplus A2 \oplus A4 \oplus C7$	$A3 \oplus A5 \oplus B1 \oplus C8$	$A3 \oplus A6 \oplus B2 \oplus C9$	$A5 \oplus A6 \oplus B4 \oplus C10$	$A8 \oplus A9 \oplus A10 \oplus B7$

2) From X_1 in $X002$, one user needs $A1$ and has $\{A2, A4, A7\}$. From X_2 in $X002$, another user needs $B1$ and has $\{A3, A5, A8\}$.
3) According to Corollary 2, the attackers conclude that the user who needed $A1$ and has $\{A2, A4, A7\}$ from $X001$ is the same as the one who needed $A2$ from $X002$.
4) Without using the elimination, the attackers cannot determine that the user who requested $A1$ in $X001$ also the request for $B1$ in $X002$.

By applying the described prediction process for all the 13 neighbor sequence pairs shown in TABLE II, the attackers are able to identify a user based on its cache content uniquely. In this example, once $X001$, $X004$, and $X007$ are collected, the attackers know that one user (whose ground truth index is user 4) misses $\{A1, B1, C1\}$ but has $\{A2, A4, A7\}$. Once $X001$, $X002$, and $X003$ are collected, another user (whose ground truth index is user 5) misses $\{A1, B1, C1\}$ but has $\{A3, A5, A8\}$.

TABLE III: Mapping Coded Subfile To User Demands

X_i	d_4
$A1 \oplus A2 \oplus A4 \oplus A7$	A
$A2 \oplus A4 \oplus A7 \oplus B1$	B
$A2 \oplus A4 \oplus A7 \oplus C1$	C
X_i	d_5
$A1 \oplus A3 \oplus A5 \oplus A8$	A
$A3 \oplus A5 \oplus A8 \oplus B1$	B
$A3 \oplus A5 \oplus A8 \oplus C1$	C

In Stage 2, based on the estimation about the cache content of users 4 and 5, the attackers generate Table III to associate each coded subfile to a user file request.

IV. Conclusion

This paper characterizes the subfile level one-to-one mapping relation between demands of a group of users and coded subfiles in the MAN scheme. This mapping relation is then utilized to design a two-stage attack for colluding users. This paper shows that even though the scheme designer limits a coded subfile to be accessed by only its designated users, there may still be a privacy leakage issue.

Rather than aiming at providing zero privacy leakage as works in [11], [12], the optimal rate a system can achieve by sacrificing part of the privacy has not been formally studied. The inherent privacy-rate-memory trade-off under the coded caching setup needs to be studied in the future.

References

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

[2] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 1999, pp. 126–134.

[3] S. Shukla and A. A. Abouzeid, "Proactive retention aware caching," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.

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

[5] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Centralized coded caching with heterogeneous cache sizes," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2017, pp. 1–6.

[6] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 349–366, 2017.

[7] M. Ji, A. Tulino, J. Llorca, and G. Caire, "Caching-aided coded multicasting with multiple random requests," in *2015 IEEE Information Theory Workshop (ITW)*. IEEE, 2015, pp. 1–5.

[8] N. Golrezaei, A. G. Dimakis, and A. F. Molisch, "Wireless device-to-device communications with distributed caching," in *2012 IEEE International Symposium on Information Theory Proceedings*. IEEE, 2012, pp. 2781–2785.

[9] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless d2d networks," *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 849–869, 2015.

[10] A. A. Zewail and A. Yener, "Device-to-device secure coded caching," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1513–1524, 2019.

[11] K. Wan and G. Caire, "On coded caching with private demands," *IEEE Transactions on Information Theory*, vol. 67, no. 1, pp. 358–372, 2020.

[12] S. Kamath, "Demand private coded caching," *arXiv preprint arXiv:1909.03324*, 2019.

[13] V. Ravindrakumar, P. Panda, N. Karamchandani, and V. M. Prabhakaran, "Private coded caching," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 685–694, 2018.

[14] V. Aravind, P. K. Sarvepalli, and A. Thangaraj, "Subpacketization in coded caching with demand privacy," in *2020 National Conference on Communications (NCC)*. IEEE, 2020, pp. 1–6.

[15] Y. Wang and A. A. Abouzeid, "On the privacy leakage of coded caching," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[16] A. Sengupta, R. Tandon, and T. C. Clancy, "Fundamental limits of caching with secure delivery," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 355–370, 2015.

[17] H. Sun and S. A. Jafar, "The capacity of robust private information retrieval with colluding databases," *IEEE Transactions on Information Theory*, vol. 64, no. 4, pp. 2361–2370, 2017.