Coded Secure Delivery for Anonymous Information Retrieval

Omer Lauer
Viterbi Dept. of Electrical and Computer Engineering
Technion – Israel Institute of Technology
omer.lauer@campus.technion.ac.il

Yuval Cassuto
Viterbi Dept. of Electrical and Computer Engineering
Technion – Israel Institute of Technology
ycassuto@ee.technion.ac.il

***Abstract*—This paper proposes and studies a scheme of coded secure delivery (CSD), where data objects are delivered upon request in a way that only users that are allowed access can recover them. The security of the scheme is driven by encryption with keys that are pre-distributed to different subsets of the users. As a result, the system can securely serve anonymous requests, enhancing its users' privacy. A CSD instance is defined formally by two binary matrices, and our main result is an algorithm to minimize the number of transmissions needed to accomplish it. Coding allows transmission of linear combinations of objects, and the power of coding is demonstrated by showing instance families with large gaps to uncoded delivery. In addition to minimizing communication costs, the paper studies the key-distribution problem toward minimizing the users' storage costs.**

I. Introduction

Almost all of our digital lives these days rely on retrieving information from centralized services, commonly known as *cloud* services, used mainly for storage of data *objects* and their delivery to *users* upon request. To maintain data privacy, objects are delivered only to users that are allowed access, typically using a mechanism of authentication and access rights verification. A side effect of such mechanism is that services have a full view of which user requests which objects, compromising the privacy of the users. To protect user's privacy, services should offer *anonymous information retrieval (AIR)*.

In the absence of access control, AIR can be trivially implemented using a proxy that aggregates requests from many users and forwards them to the service without user identities. With access control, AIR can be provided by breaking the delivery into two phases: 1) *key distribution* of secret keys to authenticated users, and 2) *object delivery* of objects by the service, after encrypting them with keys available only to allowed access users. This encryption allows the service to respond to object requests without any check of access rights, and thus without knowledge of the requesting user's identity.

A challenge in the 2-phase delivery approach lies in the increased *communication cost*, as an object may need to be transmitted multiple times (each encrypted with a different key), to fulfill all access rights. In addition, there is a *storage cost* associated with users holding the keys they received. The main thrust of this paper is minimizing these costs, and exploring interesting tradeoffs between them.

Work supported by the US-Israel Binational Science Foundation grant 2023627.

In this paper's problem setting, a server receives an (anonymous) request for multiple objects, and its task is to broadcast a batch of transmissions that will allow all users to recover the objects to which they are allowed access. The server is also bound by a security requirement: that users not allowed access to an object will *not* be able to recover it. Minimizing the communication cost is done by finding the smallest number of transmissions fulfilling these requirements. Security is achieved by using some standard symmetric encryption algorithm, with a key shared (in advance) by the server and a subset of the users. The proposed scheme employs *coded transmissions*, in which the server encrypts and broadcasts linear combinations of objects instead of single objects. Security is guaranteed by a strict policy by which a key may be used to encrypt a transmission if it is only held by users that are allowed access to *every object* in the linear combination. The importance of coding in this setting is in reducing communication costs, in a similar way to the well-known problem called *index coding* [1]. In our setting, the object delivery task can be regarded as *covering* the requested objects with a sequence of index-coding instances, where delivered objects become side information for subsequent transmissions.

In Section II we formally define the problem of *coded secure delivery (CSD)*, the delivery method, and the security policy. In Section III we characterize the requirements for a CSD solution, guaranteeing both recovery and security. Our main contribution is a two-step approach for communication-cost minimization. Section IV demonstrates the high power of coding in the CSD problem, by showing a family of instances with a provable gap between the communication costs with and without coding. Section V addresses the key-distribution problem under a storage-cost budget, and Section VI shows empirically the performance benefits of the proposed algorithms.

Prior related schemes have been proposed within the frameworks of *broadcast encryption* [2] and *secure index-coding/groupcast* [3]–[5], [6]. However, none of the prior works provides a scheme for secure delivery with a general access-control structure.

II. The Model

A central *server* stores $\bar{n}$ data *objects*, which need to be delivered, upon request, to subsets of its m client *users* $u_1, u_2, \ldots, u_m$. The objects are *access controlled*: to each object some subset of users are allowed access. The server responds to access requests, and delivers the requested objects. Each requested object needs to be delivered to all of the users that are allowed access to it (and no others), without identifying a particular user requesting the object. For the server to fulfill these requirements, it generates N *keys* $k_1, k_2, \ldots, k_N$ and distributes them to users ahead of the request phase. Each user may hold a different subset of keys, and keys can be shared by multiple users. We assume that in the set of N keys, for each user there exists a unique, *private* key, given only to her and to no other user (thus $N \geq m$). This will guarantee that any CSD instance is solvable.

A. Problem Definitions

In a CSD instance there are $n \leq \bar{n}$ requested objects $o_1, o_2, \ldots, o_n$. The instance is defined by two binary matrices:

Definition 1. The *Access Control Set* matrix ACS is an $m \times n$ binary matrix, where $\mathrm{ACS}_{j,l} = 1$ if user u_j is allowed access to object o_l, and $\mathrm{ACS}_{j,l} = 0$ if not.

Definition 2. The *Key Holder Set* matrix KHS is an $N \times m$ binary matrix, where $\mathrm{KHS}_{i,j} = 1$ if key k_i is held by user u_j, and $\mathrm{KHS}_{i,j} = 0$ if not.

B. Delivery Method and Security Policy

1) Delivery Method: It is assumed that communication from server to users is via a broadcast channel: each message from the server reaches all users. To deliver objects according to the problem definitions, each server's message is described by the *primitive* $S(x, k_i)$, where x is the message's data and k_i is the key with which the data is encrypted. We assume all data to be transmitted to have the same length in bits, and thus every transmission has a fixed *communication cost* of 1. Only users holding k_i can decrypt the message and recover x.

We distinguish between uncoded and coded transmissions, as follows. In an uncoded transmission, x is one of the objects: $x = o_l$ for some $l \in [n] \triangleq \{1, 2, \ldots, n\}$. In a coded transmission, x is a *linear combination* of multiple objects $o_{l_1}, o_{l_2}, \ldots, o_{l_D}$: $x = \sum_{d=1}^{D} h_d o_{l_d}$, where the coefficients h_d are over some finite field $\mathbb{F}_q$ (with q a prime power). Note that an uncoded transmission is a special case of a coded one with $D = 1$ and $h_1 = 1$. When calculating the communication cost we neglect the overhead of sending the indices l_d and the coefficients h_d, which is reasonable when the object size is large relative to the logarithms of n and q.

2) Security Policy: To prevent object access by non-allowed users, the server implements the following security policy. An uncoded transmission $S(o_l, k_i)$ can be sent only if all the users that hold key k_i are allowed access to object o_l. This policy can be translated to the following matrix called *Primitive Key Set (*PKS*) matrix*, defining which objects can be encrypted with which keys.

Definition 3. The *Primitive Key Set* matrix PKS is an $N \times n$ binary matrix, where $\mathrm{PKS}_{i,l} = 1$ if for all $j \in [m]$ such that $\mathrm{KHS}_{i,j} = 1$ we have $\mathrm{ACS}_{j,l} = 1$, and $\mathrm{PKS}_{i,l} = 0$ if not.

Example 1. Consider an instance with $n = 2, m = 4, N = 3$:

$$\mathrm{ACS} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}, \mathrm{KHS} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \mathrm{PKS} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}. \tag{1}$$

For compactness, we drop KHS rows of private keys.

The definition of PKS allows the following simple policy for coded transmissions:

Policy 1. *A transmission* $S\left(\sum_{d=1}^{D} h_d o_{l_d}, k_i\right)$ *can be sent by the server only if for every* $d \in [D]$ *such that* $h_d \neq 0$, $\mathrm{PKS}_{i,l_d} = 1$.

Policy 1 offers *extrinsic security*, that is, not relying on the coding to hide objects from non-allowed users.

III. Object Delivery

Given matrices ACS and KHS, it is upon the server to deliver requested objects by transmitting primitive messages that adhere to Policy 1, and are as few as possible to minimize the total communication cost. In such a delivery, the server sends T transmissions, numbered $1, 2, \ldots T$, with communication cost of T, encrypted with keys whose indices are given in the vector $(i_1, i_2, \ldots, i_T) \triangleq (i_t)_{t=1}^{T}$, $i_t \in [N]$. This vector specifies that coded message x_t of transmission t is encrypted with k_{i_t}, i.e., $S(x_t, k_{i_t})$. The linear combinations of the transmissions are specified by the matrix $\mathbf{H} \in \mathbb{F}_q^{n \times T}$. This leads to the following:

Definition 4. Given ACS and KHS matrices, a *CSD Solution* is a combination of a vector of indices $(i_t)_{t=1}^{T}$ and a coefficient matrix $\mathbf{H} \in \mathbb{F}_q^{n \times T}$. If $\mathbf{o} = (o_1, o_2, \ldots, o_n)^\top$ denotes the requested objects and $\mathbf{x} = (x_1, x_2, \ldots, x_T)^\top$ the T coded messages in the transmissions $S(x_t, k_{i_t})$, then $\mathbf{x} = \mathbf{H}^\top \mathbf{o}$.

$\top$ marks matrix transpose. With the definition above, we present the main problem in this paper:

Problem 1. *Given the instance's* ACS *and* KHS *matrices, find a CSD solution* $(i_t)_{t=1}^{T}$ *and* $\mathbf{H}$ *with minimal communication cost* T*, such that the following conditions are met:*

1) Recovery condition: *every* u_j, $j \in [m]$, *can recover from* $\mathbf{x}$ *every* o_l *such that* $\mathrm{ACS}_{j,l} = 1$, *using keys held by* u_j.
2) Security condition: *every* $S(x_t, k_{i_t})$ *satisfies Policy 1.*

A. Validity of CSD Solutions

Toward solving Problem 1, in this subsection we formulate alternative validity conditions that lend themselves better to algorithmic optimization. For that we add a few more notations. For $j \in [m]$ let $\mathrm{ACS}^{(j)}$ be the set of the object indices u_j is allowed access to: $\mathrm{ACS}^{(j)} = \{l \mid \mathrm{ACS}_{j,l} = 1\}$, likewise $\mathrm{KHS}^{(j)}$ be the set of the key indices that u_j holds: $\mathrm{KHS}^{(j)} = \{i \mid \mathrm{KHS}_{i,j} = 1\}$. Also, regarding submatrices, given $r \times s$ matrix $\mathbf{A}$ and two subsets $R \subseteq [r]$ (rows) and $S \subseteq [s]$ (columns), denote the submatrix obtained from rows R and columns S in $\mathbf{A}$ as $\mathbf{A}[R, S]$ (subvectors are defined in a similar way). Furthermore, $\mathbf{A}[R, :]$ is short for $\mathbf{A}[R, [s]]$, likewise for column selection $\mathbf{A}[:, S] \triangleq \mathbf{A}[[r], S]$.

Proposition 1. *The conditions for a valid solution under Problem 1 are equivalent to:*

1) *For every $j \in [m]$, there exists $B^{(j)} \subseteq [T]$, such that $\{i_t\}_{t\in B^{(j)}} \subseteq \mathrm{KHS}^{(j)}$, $\left|B^{(j)}\right| = \left|\mathrm{ACS}^{(j)}\right|$ and $\det\left(\mathbf{H}\left[\mathrm{ACS}^{(j)}, B^{(j)}\right]\right) \neq 0$ (recovery).*
2) *For every $t \in [T], l \in [n]$ such that $\mathrm{PKS}_{i_t,l} = 0$, $\mathbf{H}_{l,t} = 0$ (security).*

We omit the formal proof, and instead note its main idea of exploiting the favorable coupling between the recovery and security conditions. That is, the security condition imposes a structure on the matrix $\mathbf{H}$, that allows a simple recovery condition applying to individual users u_j and the submatrices of $\mathbf{H}$ corresponding to the objects in their access lists $\mathrm{ACS}^{(j)}$.

Note 1. The last equation in the recovery condition, when taken over all of the users, is equivalent to the product $\prod_{j=1}^{m} \det\left(\mathbf{H}\left[\mathrm{ACS}^{(j)}, B^{(j)}\right]\right) \neq 0$, which is a similar condition to the one considered in [7] for multicast network coding, and in [8] for structured MDS codes (among others).

Proposition 1 gives concrete algebraic conditions for solution validity, but another step is needed before getting to an algorithm for finding $\mathbf{H}$ and $(i_t)_{t=1}^{T}$ with low communication cost. We introduce more notation for this step: given an $r \times s$ binary matrix $\mathbf{M}$, the bipartite graph $G = (V_L, V_R, E)$ induced by $\mathbf{M}$, denoted $G = \mathrm{gr}(\mathbf{M})$, is a graph with left nodes $V_L = [r]$, right nodes $V_R = [s]$ and edges $E = \{(i,j) \mid i \in V_L, j \in V_R, \mathbf{M}_{i,j} = 1\}$.

We next formulate an objective to be pursued by the algorithm presented in the next subsection. We use standard definitions of *graph matching* [9].

Objective 1. *Find key indices $(i_t)_{t=1}^{T}$ and $n \times T$ binary matrix $\mathbf{M}$ such that:*

1) *For every $j \in [m]$, there exists $B^{(j)} \subseteq [T]$, such that $\{i_t\}_{t\in B^{(j)}} \subseteq \mathrm{KHS}^{(j)}$, $\left|B^{(j)}\right| = \left|\mathrm{ACS}^{(j)}\right|$ and $G_j = \mathrm{gr}\left(\mathbf{M}\left[\mathrm{ACS}^{(j)}, B^{(j)}\right]\right)$ has a perfect matching.*
2) *For every $t \in [T], l \in [n]$ such that $\mathrm{PKS}_{i_t,l} = 0$, $\mathbf{M}_{l,t} = 0$.*

Note that while the second requirement for $\mathbf{M}$ is the same as for $\mathbf{H}$ in Proposition 1, the first concerns graph matchings instead of matrix determinants.

B. Greedy Algorithm for Perfect-Matching Objective

For brevity we denote the *maximum matching size* of graph G by $\mathrm{MMS}(G)$. We present Algorithm 1 that for any CSD instance outputs an $n \times T$ matrix $\mathbf{M}$ that meets Objective 1, as well as the key indices $(i_t)_{t=1}^{T}$ and the subsets $\left(B^{(j)}\right)_{j=1}^{m}$.

Algorithm 1 maintains a matrix $\mathbf{M}$, as well as additional m graphs $G'_j = \mathrm{gr}\left(\mathbf{M}\left[\mathrm{ACS}^{(j)}, \{t' | i_{t'} \in \mathrm{KHS}^{(j)}\}\right]\right)$, $j \in [m]$. The row indices of G'_j are indices of objects u_j is allowed access to, and the column indices are iteration indices that use keys u_j has. In every iteration t it successively adds a PKS row of specific key as a column to $\mathbf{M}$ (line 9), where the chosen

Algorithm 1 Greedy Algorithm for Generating $\mathbf{M}$

```
Require: ACS, KHS; calculate PKS
 1: $\mathbf{M} \leftarrow (), (i_t) \leftarrow (), t \leftarrow 1$
 2: repeat
 3:   for all $i \in [N]$ do
 4:     $\mathbf{M}[:,t] \leftarrow (\mathrm{PKS}[i,:])^\top$, $i_t \leftarrow i$ (temp)
 5:     $\forall j \in [m]$, $G'_j \leftarrow \mathrm{gr}\left(\mathbf{M}\left[\mathrm{ACS}^{(j)}, \{t' | i_{t'} \in \mathrm{KHS}^{(j)}\}\right]\right)$
 6:     $D_i = \sum_{j=1}^{m} \mathrm{MMS}\left(G'_j\right)$
 7:   end for
 8:   $i_{max} \leftarrow \mathrm{argmax}_{i\in[N]} D_i$
 9:   $\mathbf{M}[:,t] \leftarrow (\mathrm{PKS}[i_{max},:])^\top$, $i_t \leftarrow i_{max}$ (final)
10:   $\forall j \in [m]$, $G'_j \leftarrow \mathrm{gr}\left(\mathbf{M}\left[\mathrm{ACS}^{(j)}, \{t' | i_{t'} \in \mathrm{KHS}^{(j)}\}\right]\right)$
11:   $t \leftarrow t+1$
12: until for every $j \in [m]$, $\mathrm{MMS}\left(G'_j\right) = \left|\mathrm{ACS}^{(j)}\right|$
13: for all $j \in [m]$ do
14:   $B^{(j)} \leftarrow$ right nodes in maximum matching of $G'_j$
15: end for
16: return $\mathbf{M}$, $(i_t)$, $\left(B^{(j)}\right)_{j=1}^{m}$
```

key index i_t (line 8) is the one whose PKS entries maximize the sum of matching sizes over all m graphs. The main loop terminates when every graph G'_j has a maximum matching of size $\left|\mathrm{ACS}^{(j)}\right|$, and the outputs $\left(B^{(j)}\right)_{j=1}^{m}$ are the right nodes in these maximum matchings (hence $\mathrm{gr}\left(\mathbf{M}\left[\mathrm{ACS}^{(j)}, B^{(j)}\right]\right)$ has a perfect matching). Since the matching condition is also necessary for linear coding (proof omitted), the maximization in line 8 implies a local-optimality property: that at every iteration the added transmission maximizes the number of recoverable objects given the previous transmissions.

Using $\mathbf{M}$, a coefficient matrix $\mathbf{H} \in \mathbb{F}_q^{n\times T}$ can be easily found, such that together with $(i_t)_{t=1}^{T}$ it is a valid CSD solution with high probability. This is done using a randomized algorithm similar to [7], where the success probability in our setting is shown to be at least $\left(1 - \frac{m}{q}\right)^n$. This 2-step approach has a major advantage: it allows the greedy Algorithm 1 to decide only *which key* to use in every iteration (line 8), without worrying about which objects to encode with it.

IV. Power of Coding

A natural question about the CSD problem is how much coded transmissions can reduce communication cost over uncoded ones. The answer clearly depends on the specific instance, and in this section we give an extremal answer by showing instance families with large coding advantages.

A. A Family of CSD Instances

We define a family of layered CSD instances specified by their ACS and KHS matrices.

Family 1. $\mathbf{A}_d$, $\mathbf{K}_d$ *(*ACS, ***transposed***[1] KHS *of a d-layer instance, respectively) are defined recursively:*

[1]We specify the transpose of the KHS for convenience, so both $\mathbf{A}_d$ and $\mathbf{K}_d$ have row dimension m, the number of users.

$$\mathbf{A}_1 = 1,$$

$$\mathbf{A}_{d+1} = \begin{array}{c} \overset{d+1}{\longleftrightarrow} \\ \left(\begin{array}{ccccc} \mathbf{A}_d[:,1] & \mathbf{A}_d[:,2] & \cdots & \mathbf{A}_d[:,d] & \mathbf{0} \\ \mathbf{A}_d[:,1] & \mathbf{A}_d[:,2] & \cdots & \mathbf{0} & \mathbf{A}_d[:,d] \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{A}_d[:,1] & \mathbf{0} & \cdots & \mathbf{A}_d[:,d-1] & \mathbf{A}_d[:,d] \\ \mathbf{0} & \mathbf{A}_d[:,1] & \cdots & \mathbf{A}_d[:,d-1] & \mathbf{A}_d[:,d] \\ \hline 1 & 1 & \cdots & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & 1 \end{array}\right) \end{array} \begin{array}{l} \updownarrow \mathbf{d+1} \\ \\ \updownarrow d+1 \end{array}, \tag{2}$$

$$\mathbf{K}_1 = 1,$$

$$\mathbf{K}_{d+1} = \begin{array}{c} \overset{\mathbf{d+1}}{\longleftrightarrow} \\ \left(\begin{array}{ccccc|c} \mathbf{K}_d & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_d & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{K}_d & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{K}_d & \mathbf{0} \\ \hline \mathbf{K}_d[-1,:] & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & 1 \\ \mathbf{0} & \mathbf{K}_d[-1,:] & \cdots & \mathbf{0} & \mathbf{0} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{K}_d[-1,:] & \mathbf{0} & 1 \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{K}_d[-1,:] & 1 \end{array}\right) \end{array} \begin{array}{l} \updownarrow \mathbf{d+1} \\ \\ \updownarrow d+1 \end{array}. \tag{3}$$

$\mathbf{d+1}$ *is an abuse of notation for $d+1$ matrix blocks, in this case a column of the form $\mathbf{A}_d[:,l]$ (in $\mathbf{A}_{d+1}$) or $\mathbf{K}_d[:,i]$ (in $\mathbf{K}_{d+1}$), and $\mathbf{K}_d[-1,:]$ denotes the last row of $\mathbf{K}_d$.*

B. Coded and Uncoded Communication Costs of Family 1

Denote by $T_u^{(d)}$ and $T_c^{(d)}$ the minimal communication costs that can deliver Family 1's d-layer instance with uncoded and coded transmissions, respectively.

Proposition 2. *$T_c^{(d)}$ satisfies the following recursive formula*

$$T_c^{(1)} = 1;\ T_c^{(d+1)} \le (d+1)\,T_c^{(d)} + 1,\ \forall d \ge 1. \tag{4}$$

Proposition 3. *$T_u^{(d)}$ satisfies the following recursive formula*

$$T_u^{(1)} = 1;\ T_u^{(d+1)} \ge (d+1)\,T_u^{(d)} + d + 1,\ \forall d \ge 1. \tag{5}$$

Define the instance's *coding savings ratio* as $r^{(d)} \triangleq T_c^{(d)}/T_u^{(d)}$. Substituting the upper bound on $T_c^{(d)}$ from (4) and the lower bound on $T_u^{(d)}$ from (5) gives an upper bound on $r^{(d)}$, which is a quantitative lower bound on the power of coding.

While Proposition 2 is proved by showing a constructive solution, Proposition 3 requires a tool for showing impossibility. This tool is built upon the concept of *essential keys*.

Definition 5. k_i is *essential* with respect to (w.r.t.) a user-object pair (u_j, o_l) if $\{i' \mid \mathrm{KHS}_{i',j} = 1, \mathrm{PKS}_{i',l} = 1\} = \{i\}$. The *essentiality* of key k_i with respect to a user u_j is defined as $\mathrm{Essen}\,(i,j) \triangleq \{l \mid k_i \text{ is essential w.r.t. } (u_j, o_l)\ \}$.

Let $T(i)$ be the number of times k_i is used in a solution of a CSD instance. We next show that in any valid solution, $T(i)$ is lower bounded depending on the essentiality sets of k_i, and that this bound is different for coded and uncoded solutions.

Proposition 4. *In any valid solution of a CSD instance with key vector $(i_t)_{t=1}^{T}$, for any key index i: in a coded solution $T(i) \ge \max_{j\in[m]} |\mathrm{Essen}\,(i,j)|$, and in an uncoded solution $T(i) \ge \left|\bigcup_{j=1}^{m} \mathrm{Essen}\,(i,j)\right|$.*

Remark 1. Proposition 4 (proof omitted) captures a fundamental combinatorial gap between coded and uncoded solutions, which may be useful beyond proving Proposition 3.

The following measures the power of coding in CSD.

Corollary 5. *There exist CSD instances with coding savings ratio tending to $\frac{e-1}{e} \approx 0.632$.*

The corollary can be proved by substituting into the ratio $T_c^{(d)}/T_u^{(d)}$ the closed-form expressions corresponding to the recursive expressions in the right-hand sides of (4), (5).

V. Key Distribution

In this section the ACS is the full access-control matrix (of all the system's objects), thus $n = \bar{n}$. The problem of key distribution is *how to determine the* KHS *given an* ACS*?*

In the key-distribution phase, an $N \times m$ KHS matrix is decided, and then key k_i is given to user u_j if and only if $\mathrm{KHS}_{i,j} = 1$. We refer to the cumulative number of keys held by all the users (including multiplicities of same keys held by multiple users) as the KHS's *storage cost*[2]:

Definition 6. The *storage cost* of a KHS matrix is the matrix's total weight: $\sum_{i=1}^{N}\sum_{j=1}^{m} \mathrm{KHS}_{i,j}$.

A *key-distribution algorithm* chooses KHS given input ACS, and is subject to a storage-cost *budget*. The key-distribution problem therefore is as follows:

Problem 2. *Given* ACS *and storage-cost budget τ, choose* KHS *with storage cost at most τ that enables valid CSD solutions with low communication costs.*

Without the budget constraint, the problem is trivial because a solution $\mathrm{KHS} = \mathrm{ACS}^{\top}$ enables delivering all n objects with uncoded communication cost of n, which is the lowest possible. Therefore, we will be interested in solutions with small budgets $\tau \ll \sum_{j=1}^{m}\sum_{l=1}^{n} \mathrm{ACS}_{j,l}$. We evaluate our key-distribution algorithm communication-cost performance, averaged over random ACS matrices with a given density p, i.e., each $\mathrm{ACS}_{j,l}$ is 1 with probability p and 0 with probability $1-p$. We restrict the density to $p < 1/2$, as CSD instances with very dense ACS matrices are less interesting due to high communication cost for each object request.

A. ACS-Covering Key-Distribution Algorithms

In this subsection we propose key-distribution algorithms that collectively cover the elements of the ACS. To this end, the algorithms maintain an $m \times n$ *cover matrix* $\mathbf{C}$, where $\mathbf{C}_{j,l}$ equals the number of keys k_i previously distributed to u_j and such that $\mathrm{PKS}_{i,l} = 1$. PKS and $\mathbf{C}$ are updated after each key is distributed, and $\mathbf{C}_{j,l} = \perp$ (undefined) throughout the run if $\mathrm{ACS}_{j,l} = 0$. In principle, the algorithms will prefer choosing

[2]Note that this is also the cost of communicating the keys to the users, but we use the term "storage" to distinguish from the object-delivery communication cost.

keys that increase as many entries with small (e.g., 0) current $\mathbf{C}_{j,l}$ values, because the aim toward low communication cost is that as many u_j, o_l pairs will be covered by the distributed keys. Define the *key degree* of k_i as $\sum_{j=1}^{m} \mathrm{KHS}_{i,j}$.

1) Covering-Maximizing Algorithm: Initialize $\mathbf{C}$. In each iteration i first choose a key degree d . Find a size-d subset of $[m]$ for which a distributed key k_i would result in $\mathbf{C}$ with fewest zeros, and distribute k_i to this subset. Continue to next iteration if storage cost of KHS is less than τ.

A special case of interest for covering-maximizing algorithms is distributing keys with degree $d = 2$. There are two main motivations to consider a small d: 1) small key degrees typically cover $\mathbf{C}_{j,l}$ entries of multiple object indices l, and 2) the key selection is more tractable because the number of user sets to consider grows as m^d.

2) Pair-Covering ($d = 2$) Algorithm: Initialize $\mathbf{C}$. In each iteration i find a size-2 subset $\{j_1, j_2\} \subset \{1, \ldots, m\}$ that maximizes

$$\sum_{l \in \mathrm{ACS}^{(j_1)} \cap \mathrm{ACS}^{(j_2)}} \big(2I\left(\mathbf{C}_{j_1,l} = 0, \mathbf{C}_{j_2,l} = 0\right) + I\left(\mathbf{C}_{j_1,l} = 1, \mathbf{C}_{j_2,l} = 0\right) + I\left(\mathbf{C}_{j_1,l} = 0, \mathbf{C}_{j_2,l} = 1\right)\big), \tag{6}$$

where $I(\cdot)$ is the indicator function of the event in its argument. Distribute k_i to u_{j_1}, u_{j_2} and continue to next iteration if storage cost of KHS is less than τ.

Note that the condition under the sum in (6) guarantees that $\mathbf{C}_{j_1,l} > 0, \mathbf{C}_{j_2,l} > 0$ after adding k_i, because it implies $\mathrm{PKS}_{i,l} = 1$; the first term has weight 2 because two entries change from zero to non-zero. The algorithm counts not only uncovered entries (with $\mathbf{C}_{j,l} = 0$), but also ones with $\mathbf{C}_{j,l} = 1$, leading to $\mathbf{C}_{j,l} > 1$ following the selection. This is justified for coded solutions by the following lemma. For key k_i denote $U(i) \triangleq \{j \mid \mathrm{KHS}_{i,j} = 1\}$, $O(i) \triangleq \{l \mid \mathrm{PKS}_{i,l} = 1\}$: the users who hold k_i and the objects that can be delivered with k_i. Then:

Lemma 6. *If $\mathbf{C}_{j,l} = 1$ for every $j \in U(i), l \in O(i)$, then coded transmissions using k_i cannot reduce its communication cost $T(i)$ relative to uncoded transmissions.*

Thus Lemma 6 (proof omitted) motivates keys that result in $\mathbf{C}_{j,l} > 1$. The selection criterion (6) allows that, but still prioritizes covering j, l pairs that currently have $\mathbf{C}_{j,l} = 0$.

VI. Empirical Results

We implemented the algorithms presented in the paper [3], and in this section we show their empirical performance. The core of the presented results is Algorithm 1 for finding coded CSD solutions, which is compared to uncoded solutions, found by a greedy set-cover algorithm [11] for each object o_l. The main performance metric is the *normalized communication cost* denoted $\bar{T}$, which is the communication cost T divided by the weight (number of ones) of the ACS matrix. We also define the *normalized storage-cost budget* $\bar{\tau}$, which equals the storage-cost budget τ divided by the weight of the ACS matrix.

[3]Simulation code is available at [10].

Our experiment evaluates the performance of algorithm V-A2 applied to random ACS instances with parameters $m = n = 50$, $p = 0.25$. We plot these results in Fig. 1.

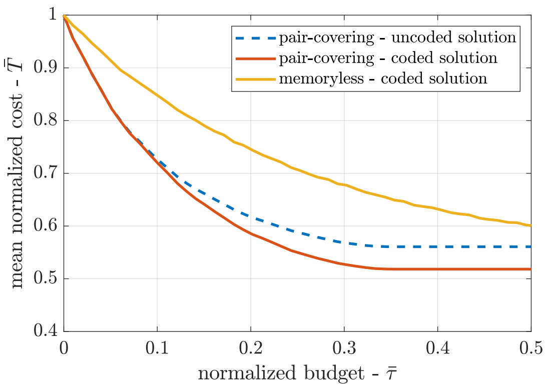

Fig. 1. Normalized communication cost versus normalized storage budget for pair-covering and memoryless key distribution.

Fig. 1 demonstrates that coded CSD solutions achieve lower communication cost compared to uncoded, even for random i.i.d. ACS matrices. Alternatively, for a given communication cost, coding allows significant reduction of storage budgets in the practically interesting intermediate range (e.g., $0.1 < \bar{\tau} < 0.3$). For example, coded solutions achieve communication cost of $\bar{T} = 0.55$ with 25% lower budget compared to uncoded. We also plot in Fig. 1 the results of a key-distribution algorithm that selects keys in a memoryless fashion, without attempting to cover the ACS (details omitted). The large gap attests to the strength of the proposed covering approach.

References

[1] Z. Bar-Yossef, Y. Birk, T. Jayram, and T. Kol, "Index coding with side information," *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1479–1494, 2011.

[2] A. Fiat and M. Naor, "Broadcast encryption," in *Advances in Cryptology—CRYPTO'93: 13th Annual International Cryptology Conference Santa Barbara, California, USA August 22–26, 1993 Proceedings 13*. Springer, 1994, pp. 480–491.

[3] S. H. Dau, V. Skachek, and Y. M. Chee, "On the security of index coding with side information," *IEEE Transactions on Information Theory*, vol. 58, no. 6, pp. 3975–3988, 2012.

[4] M. M. Mojahedian, M. R. Aref, and A. Gohari, "Perfectly secure index coding," *IEEE Transactions on Information Theory*, vol. 63, no. 11, pp. 7382–7395, 2017.

[5] V. Narayanan, J. Ravi, V. K. Mishra, B. K. Dey, N. Karamchandani, and V. M. Prabhakaran, "Private index coding," *IEEE Transactions on Information Theory*, vol. 68, no. 3, pp. 2020–2049, 2021.

[6] H. Sun, "Compound secure groupcast: Key assignment for selected broadcasting," *IEEE Journal on Selected Areas in Information Theory*, vol. 3, no. 2, pp. 379–389, 2022.

[7] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on information theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[8] S. H. Dau, W. Song, Z. Dong, and C. Yuen, "Balanced sparsest generator matrices for MDS codes," in *2013 IEEE International Symposium on Information Theory*. IEEE, 2013, pp. 1889–1893.

[9] J. H. Van Lint and R. M. Wilson, *A Course in Combinatorics*, 2nd ed. Cambridge University Press, 2001.

[10] O. Lauer. (2025) Coded Secure Delivery. [Online]. Available: https://github.com/omerlauer/csd_for_air

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. MIT Press, 2022.