Weighted Universal Recovery, Practical Secrecy, and an Efficient Algorithm for Solving Both

Thomas A. Courtade and Richard D. Wesel
Department of Electrical Engineering
University of California, Los Angeles
Los Angeles, California 90095
Email: tacourta@ee.ucla.edu, wesel@ee.ucla.edu

***Abstract*—In this paper, we consider a network of n nodes, each initially possessing a subset of packets. Each node is permitted to broadcast functions of its own packets and the messages it receives to all other nodes via an error-free channel. We provide an algorithm that efficiently solves the Weighted Universal Recovery Problem and the Secrecy Generation Problem for this network.**

In the Weighted Universal Recovery Problem, the goal is to design a sequence of transmissions that ultimately permits all nodes to recover all packets initially present in the network. We show how to compute a transmission scheme that is optimal in the sense that the weighted sum of the number of transmissions is minimized.

For the Secrecy Generation Problem, the goal is to generate a secret-key among the nodes that cannot be derived by an eavesdropper privy to the transmissions. In particular, we wish to generate a secret-key of maximum size. Further, we discuss private-key generation, which applies to the case where a subset of nodes is compromised by the eavesdropper.

For the network under consideration, both of these problems are combinatorial in nature. We demonstrate that each of these problems can be solved efficiently and exactly. Notably, we do not require any terms to grow asymptotically large to obtain our results. This is in sharp contrast to classical information-theoretic problems despite the fact that our problems are information-theoretic in nature.

Finally, the algorithm we describe efficiently solves an Integer Linear Program of a particular form. Due to the general form we consider, it may prove useful beyond these applications.

I. Introduction

Consider a set of n nodes, each in possession of a set of packets. Formally, let $P = \{p_1, \dots, p_m\}$ be an indexed set of m (possibly repeated) packets, where each p_i is an element of a sufficiently large finite field $\mathbb{F}$ equipped with addition and multiplication operations $\oplus$ and $\otimes$ respectively. Let $P_i \subseteq P$ denote the set of packets initially available at node i and assume that $\{P_i\}_{i=1}^n$ satisfies $\bigcup_{i=1}^n P_i = P$. Each node is permitted to broadcast functions of its own packets and the messages it receives to all other nodes via an error-free $|\mathbb{F}|$-ary channel (i.e., nodes are restricted to sending an integer number of packets).

The network model we just described is a realistic one for many situations. For example, a collection of servers will contain many files, some of which are repeated on several servers. Servers are allowed to communicate with one another by broadcasting IP packets to all other servers in the network. To each server, this channel can be viewed as an error-free $|\mathbb{F}|$-ary channel.

Two questions naturally arise for this type of network:

1) The Weighted Universal Recovery Problem: How can we derive a communication scheme so that (i) each node is ultimately able to recover all packets in P, and (ii) the weighted sum of the transmissions by the nodes is minimized?
2) The Secrecy Generation Problem: If an eavesdropper is permitted to observe all transmissions between the nodes, what is the largest secret-key that can be generated among all the nodes? What is the largest private-key that can be generated among a subset of nodes if the complementary subset is compromised by the eavesdropper?

Each of these problems is combinatorial in nature and, as we shall see, they are intimately connected. In this paper, we show that each of these problems can be efficiently (and exactly) solved for the network model we described. Moreover, we provide an explicit algorithm based on submodular function optimization that solves each of these problems via solving an Integer Linear Program (ILP) of a particular form. Due to the general form of the ILP we consider, our algorithm may prove useful for solving a variety of combinatorial problems beyond these particular applications. Before proceeding, we give a very brief introduction to these two problems.

A. The Universal Recovery Problem

Distributed data exchange problems have received a great deal of attention over the past several years. The powerful techniques afforded by network coding [1], [2] have paved the way for cooperative communications at the packet-level.

The unweighted universal recovery problem was originally introduced in [3]. The authors of the present paper characterized the set of transmission strategies that permit universal recovery for multi-hop networks in [4], [5]. Moreover, tight concentration results were given on the number of transmissions required when packets are randomly distributed in the network [4], [5].

A randomized algorithm for solving the unweighted universal recovery problem was given in [6], and the first deterministic algorithm for solving the unweighted universal recovery problem was recently given in [7]. In the concluding remarks of [7], the weighted universal recovery problem was posed as an open problem. The present paper definitively resolves this issue.

B. The Secrecy Generation Problem

In the context of this paper, the secrecy generation problem was originally studied in [8]. In [8], the authors gave single-letter characterizations of the secret-key and private-key capacities for a network of nodes connected by an error-free broadcast channel. While these results are very general, they are insufficient from a practical perspective for two reasons. First, (as with most information-theoretic problems) the results require the nodes to observe arbitrarily long sequences of iid source symbols, which is generally not the case in practice. Second, no efficient algorithm is provided which achieves the respective secrecy capacities. Some recent work [9], [10] has addressed the latter point.

In this paper, we address these two issues. In particular, we leverage the results of [8] and apply them to the network model that we described above. In doing so, we provide an efficient algorithm that achieves the secrecy capacity without requiring any quantities to grow asymptotically large.

C. Organization

This paper is organized as follows. In Section II, we formally introduce the weighted universal recovery problem and discuss the corresponding solution. Section III presents the secrecy generation problem and gives the secret-key and private-key capacities. In Section IV, we derive an algorithm that efficiently solves a particular ILP. Since the universal recovery problem and the secrecy generation problem are instances of this ILP, the algorithm applies to them as well. Complexity of the algorithm is also discussed in Section IV. Section V delivers the conclusions.

II. Weighted Universal Recovery

In the first part of this paper, we study the weighted universal recovery problem. In this problem, one considers a single-hop network of n nodes that all wish to recover m desired packets. A transmission scheme is said to permit universal recovery if every node is ultimately able to recover all m packets initially present in the network. The goal is to find a transmission scheme permitting universal recovery that minimizes the weighted sum of the of transmissions made by each node.

Formally, let $w \in \mathbb{R}^n$ be a non-negative weight vector and let $P_i \subseteq \{p_1, \dots, p_m\} = P$ be the set of packets originally available at node i. Each node is aware of the indices of the packets available to every other node. Assume $\{P_i\}_{i=1}^n$ satisfies $\bigcup_{i=1}^n P_i = P$, and define $P_i^c = P \backslash P_i$. Each $p_j \in \mathbb{F}$, where $\mathbb{F}$ is some finite field (e.g. $\mathbb{F} = \text{GF}(2^m)$) equipped with addition and multiplication operations $\oplus$ and $\otimes$ respectively. We assume throughout that the p_i's are mutually independent and uniformly distributed over $\mathbb{F}$. Node i is allowed to make at most x_i transmissions and universal recovery is achieved if all nodes are ultimately able to recover all m packets. The goal is to minimize $w^T x$ subject to x permitting universal recovery. This is best demonstrated by a simple example.

Example 1: Suppose we have a network of three nodes, and let $P_i = \{p_1, p_2, p_3\} \backslash p_i$. One transmission is not sufficient, thus $\sum_{i=1}^3 x_i \geq 2$. It can be seen that two transmissions suffice: let node 1 transmit p_2 which lets node 2 have $P_2 \cup p_2 = \{p_1, p_2, p_3\}$. Now, node 2 transmits $p_1 \oplus p_3$, allowing nodes 1 and 3 to each recover all three packets (by "subtracting" off the packet already known to them respectively). This transmission scheme is optimal for any $w_3 \geq w_2 \geq w_1 \geq 0$.

It turns out that the set of transmission schemes permitting universal recovery is described by a polytope. We quote a result from [4], [5] which characterizes this set. Define $E = \{1, \dots, n\}$ throughout.

Theorem 1 (Single-Hop Universal Recovery [4], [5]): If node i is allowed to make at most x_i transmissions, then $x = (x_1, \dots, x_n)$ permits universal recovery if and only if

$$\sum_{i \in U} x_i \geq \left| \bigcap_{i \in E \backslash U} P_i^c \right|, \text{ for all } \emptyset \neq U \subsetneq E. \tag{1}$$

In light of Theorem 1, the weighted universal recovery problem can be solved by optimizing the following Integer Linear Program (ILP):

$$\begin{aligned} \text{minimize} \quad & w^T x \qquad (2) \\ \text{subject to:} \quad & \sum_{i \in U} x_i \geq \left| \bigcap_{i \in E \backslash U} P_i^c \right| \text{ for all } \emptyset \neq U \subsetneq E. \end{aligned}$$

Our first main result is the following:

Theorem 2: The weighted universal recovery problem can be solved in polynomial time.

We derive an algorithm that solves ILPs of the same form as (2) in Section IV. We remark that, given a vector x which permits universal recovery, the encoding and decoding operations can be efficiently computed using the algorithm by Jaggi et al. [12]. See [4], [5] for details. Therefore, the difficult part in solving the universal recovery problem lies entirely in determining an optimal vector x which permits universal recovery.

To our knowledge, the algorithm we derive in Section IV is the first polynomial time algorithm for solving the weighted single-hop universal recovery problem and the second polynomial time algorithm for solving the unweighted single-hop universal recovery problem ($w_i = 1, \forall i$), with the first being the recently proposed Deterministic Data Exchange (DDE) Algorithm [7]. DDE takes a linear-algebraic approach that relies on a maximum-rank subroutine. The stark differences between the algebraic approach of DDE and the combinatorial approach described in Section IV make each interesting in its own right. One advantage of our algorithm is the ability to quickly generate a certificate of optimality or infeasibility as described in Section IV-D and IV-E.

III. Practical Secrecy Generation

In this section we retain the setup of the universal recovery problem, but we consider a different goal. In particular, we wish to generate a secret-key among the nodes that cannot be derived by an eavesdropper privy to all of the transmissions among nodes. Also, like the nodes themselves, the eavesdropper is assumed to know the indices of the packets initially available to each node. The goal is to generate the maximum amount of "secrecy" that cannot be determined by the eavesdropper.

The theory behind secrecy generation among multiple terminals was originally established in [8] for a very general class of problems. Our results should be interpreted as a practical application of the theory originally developed in [8]. Indeed, our results and proofs are special cases of those in [8] which have been significantly streamlined to deal with the scenario under consideration. The aim of the present section is to show how secrecy can be generated in a *practical* scenario. In particular, we show that it is possible to efficiently generate the theoretically maximum amount of secrecy among nodes in the network described in the beginning of Section I. Moreover, we show that this is possible in the non-asymptotic regime (i.e., there are no ϵ's and we don't require $n \to \infty$.). Finally, we show that it is possible to generate perfect secrecy instead of ϵ-secrecy without any sacrifice.

A. Practical Secrecy Results

In this subsection, we state our secrecy results. We begin with some definitions[1]. Let $\mathbf{F}$ denote the set of all transmissions (all of which are available to the eavesdropper by definition). A function K of P is called a secret key (SK) if K is recoverable by all nodes after observing $\mathbf{F}$, and it satisfies the [perfect] secrecy condition

$$I(K; \mathbf{F}) = 0,$$

and the uniformity condition

$$\Pr(K = k) = \frac{1}{|\mathcal{K}|} \text{ for all } k \in \mathcal{K},$$

where $\mathcal{K}$ is the alphabet of possible keys.

We define $C_{SK}(P_1, \dots, P_n)$ to be the secret-key capacity for a particular distribution of packets. We will drop the notational dependence on $P_1, \dots, P_n$ where it doesn't cause confusion. By this we mean that a secret-key K can be generated if and only if $\mathcal{K} = \mathbb{F}^{C_{SK}}$. In other words, the nodes can generate exactly C_{SK} packets worth of secret-key; no more, and no less.

Let $M_{SK}(P_1, \dots, P_n)$ be the optimum value of the following ILP:

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in E} x_i \qquad (3) \\ \text{subject to:} \quad & \sum_{i \in U} x_i \geq \left| \bigcap_{i \in E \setminus U} P_i^c \right| \text{ for all } \emptyset \neq U \subsetneq E. \end{aligned}$$

Our first result of this section is the following:

Theorem 3: The secret-key capacity is given by: $C_{SK}(P_1, \dots, P_n) = |P| - M_{SK}(P_1, \dots, P_n)$.

Next, consider the related problem where a subset $D \subsetneq E$ of nodes is compromised. In this problem, the eavesdropper has access to $\mathbf{F}$ and P_i for $i \in D$. In this case, the secret-key should also be kept hidden from the nodes in D (or else the eavesdropper could also recover it). Thus, for a subset of nodes D, let $P_D = \bigcup_{i \in D} P_i$, and call K a private-key (PK) if it is a secret-key which is only recoverable by the nodes in $E \setminus D$, and also satisfies the stronger secrecy condition:

$$I(K; \mathbf{F}, P_D) = 0.$$

Similar to above, define $C_{PK}(P_1, \dots, P_n, D)$ to be the private-key capacity for a particular distribution of packets and subset of nodes D. Again, we mean that a private-key K can be generated if and only if $\mathcal{K} = \mathbb{F}^{C_{PK}}$. In other words, the nodes in $E \setminus D$ can generate exactly C_{PK} packets worth of private-key; no more, and no less. Note that, since P_D is known to the eavesdropper, each node $i \in D$ can transmit its respective set of packets P_i without any loss of secrecy capacity.

Let $M_{PK}(P_1, \dots, P_n, D)$ be the optimum value of the following ILP:

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in D^c} x_i \qquad (4) \\ \text{subject to:} \quad & \sum_{i \in U} x_i \geq \left| \bigcap_{i \in E \setminus U} P_i^c \right| \text{ for all } \emptyset \neq U \subsetneq D^c. \end{aligned}$$

Our second result of this section is the following:

Theorem 4: The private-key capacity is given by: $C_{PK}(P_1, \dots, P_n, D) = |P \setminus P_D| - M_{PK}(P_1, \dots, P_n, D)$.

The basic idea here is that the users in D^c should just generate a secret-key from $P \setminus P_D$.

By the definitions of the SK and PK capacities, observe that it is possible to compute these capacities efficiently by solving the unweighted universal recovery problem. Moreover, as we will see in the achievability proofs, these capacities can be achieved by performing universal recovery. Thus, the algorithm we present in Section IV can be employed to efficiently solve the practical secrecy generation problem we consider.

[1] We attempt to follow the notation of [8] where appropriate.

We conclude this subsection with an example to illustrate the results.

Example 2: Consider again the network of Example 1 and assume $\mathbb{F} = \{0,1\}$ (i.e., each packet is a single bit). The secret-key capacity for this network is 1 bit. After performing universal recovery, the eavesdropper knows p_2 and the parity $p_1 \oplus p_3$. A perfect secret-key is $K = p_1$ (we could alternatively use $K = p_3$). If any of the nodes are compromised by the eavesdropper, the private-key capacity is 0.

We remark that the secret-key in the above example can in fact be attained by all nodes using only one transmission (i.e., universal recovery is not a prerequisite for secret-key generation). However, it remains true that only one bit of secrecy can be generated.

B. Proof of Theorems 3 and 4

In this subsection, we prove Theorems 3 and 4. We again remark that our proofs can be seen as special cases of those in [8] which have been adapted to the case at hand. We require the following lemma.

Lemma 1: Given a packet distribution $P_1, \dots, P_n$, let K be a secret-key achievable with communication $\mathbf{F}$. Then the following holds:

$$H(K|\mathbf{F}) = H(P) - \sum_{i=1}^{n} x_i.$$

for some vector $x = (x_1, \dots, x_n)$ which is feasible for ILP (3).

Moreover, if K is a PK (with respect to a set D) and each node $i \in D$ transmits its respective set of packets P_i, then

$$H(K|\mathbf{F}) = H(P|P_D) - \sum_{i \in D^c} x_i.$$

for some vector $x = (x_1, \dots, x_n)$ which is feasible for ILP (4).

Proof: We assume throughout that all entropies are with respect to the base-$|\mathbb{F}|$ logarithm (i.e., information is measured in packets). For this and the following proofs, let $\mathbf{F} = (F_1, \dots, F_n)$ and $F_{[1,i]} = (F_1, \dots, F_i)$, where F_i denotes the transmissions made by node i. For simplicity, our proof does not take into account interactive communication, but can be modified to do so. Allowing interactive communication does not change the results. See [8] for details.

Since K and $\mathbf{F}$ are functions of P:

$$\begin{aligned} H(P) &= H(\mathbf{F}, K, P_1, \dots, P_n) \\ &= \sum_{i=1}^{n} H(F_i|F_{[1,i-1]}) + H(K|\mathbf{F}) \\ &\quad + \sum_{i=1}^{n} H(P_i|\mathbf{F}, K, P_{[1,i-1]}). \end{aligned}$$

Set $x_i = H(F_i|F_{[1,i-1]}) + H(P_i|\mathbf{F}, K, P_{[1,i-1]})$. Then, the substituting x_i into the above equation yields:

$$H(K|\mathbf{F}) = H(P) - \sum_{i=1}^{n} x_i.$$

To show that $x = (x_1, \dots, x_n)$ is a feasible vector for ILP (3), we write:

$$\begin{aligned} \left| \bigcap_{i \in E \setminus U} P_i^c \right| &= H(P_U|P_{U^c}) \\ &= H(\mathbf{F}, K, P_U|P_{U^c}) \\ &= \sum_{i=1}^{n} H(F_i|F_{[1,i-1]}, P_{U^c}) + H(K|\mathbf{F}, P_{U^c}) \\ &\quad + \sum_{i \in U} H(P_i|\mathbf{F}, K, P_{[1,i-1]}, P_{U^c \cap [i+1,n]}) \\ &\leq \sum_{i \in U} H(F_i|F_{[1,i-1]}) + \sum_{i \in U} H(P_i|\mathbf{F}, K, P_{[1,i-1]}) \\ &= \sum_{i \in U} x_i. \end{aligned}$$

In the above inequality, we used the fact that conditioning reduces entropy, the fact that K is a function of $(\mathbf{F}, P_{U^c})$ for any $U \neq E$, and the fact that F_i is a function of P_i (by the assumption that communication is not interactive).

To prove the second part of the lemma, we can assume $D = \{1, \dots, k\}$. The assumption that each node i in D transmits all of the packets in P_i implies $F_i = P_i$. Thus, for $i \in D$ we have $x_i = H(P_i|P_{[1,i-1]})$. Repeating the above argument, we obtain

$$\begin{aligned} H(K|\mathbf{F}) &= H(P) - H(P_D) - \sum_{i \in D^c} x_i \\ &= H(P|P_D) - \sum_{i \in D^c} x_i, \end{aligned}$$

completing the proof of the lemma. $\blacksquare$

Proof of Theorem 3: Converse Part. Suppose K is a secret-key achievable with communication $\mathbf{F}$. Then, by definition of a SK and Lemma 1 we have

$$\begin{aligned} C_{SK} = H(K) = H(K|\mathbf{F}) &= H(P) - \sum_{i=1}^{n} x_i \\ &\leq H(P) - M_{SK} = |P| - M_{SK}. \end{aligned}$$

Achievability Part. By definition, universal recovery can be achieved with M_{SK} transmissions. Moreover, the communication $\mathbf{F}$ can be generated as a linear function of P (see [4]). Denote this linear transformation by $\mathbf{F} = \mathcal{L}P$. Note that $\mathcal{L}$ only depends on the indices of the packets available to each node, not the values of the packets themselves (see [12] and [4]). Let $\mathcal{P}_{\mathbf{F}} = \{P' : \mathcal{L}P' = \mathbf{F}\}$ be the set of all packet distributions which generate $\mathbf{F}$.

By our assumption that the packets are iid uniform from $\mathbb{F}$, each $P' \in \mathcal{P}_{\mathbf{F}}$ is equally likely given $\mathbf{F}$ was observed. Since

$\mathbf{F}$ has dimension M_{SK}, $|\mathcal{P}_{\mathbf{F}}| = \mathbb{F}^{|P|-M_{SK}}$. Thus, we can set $\mathcal{K} = \mathbb{F}^{|P|-M_{SK}}$ and label each $P' \in \mathcal{P}_{\mathbf{F}}$ with a unique element in $\mathcal{K}$. The label for the actual P (which is reconstructed by all nodes after observing $\mathbf{F}$) is the secret-key. Thus, $C_{SK} \geq |P| - M_{SK}$.

We remark that this labelling can be done efficiently by an appropriate linear transformation mapping P to K. ■

Proof of Theorem 4: Converse Part. Suppose K is a private-key. Then, by definition of a PK and Lemma 1,

$$\begin{aligned} C_{PK} = H(K) = H(K|\mathbf{F}) &= H(P|P_D) - \sum_{i \in D^c} x_i \\ &\leq H(P|P_D) - M_{PK} = |P \backslash P_D| - M_{PK}. \end{aligned}$$

Achievability Part. Let each node $i \in D$ transmit P_i so that we can update $P_j \leftarrow P_j \cup P_D$ for each $j \in D^c$. Now, consider the universal recovery problem for only the nodes in D^c. By Theorem 1, M_{PK} is the minimum number of transmissions required among the nodes in D^c so that each node in D^c recovers P. At this point, the achievability proof proceeds identically to the SK case. ■

IV. An Efficiently Solvable ILP

In this section, we introduce a special ILP and provide an efficient algorithm for solving it. This algorithm can be used to efficiently solve the weighted universal recovery problem and, consequently, the secrecy generation problem. We begin by introducing some notation.

Let $E = \{1, \ldots, n\}$ be a finite set with n elements. We denote the family of all subsets of E by 2^E. We frequently use the compact notation $E \backslash U$ and $U + i$ to denote the sets $E \cap U^c$ and $U \cup \{i\}$ respectively. For a vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, define the corresponding functional $x : 2^E \to \mathbb{R}$ as:

$$x(U) := \sum_{i \in U} x_i, \text{ for } U \subseteq E.$$

Throughout this section, we let $\mathcal{F} = 2^E - \{\emptyset, E\}$ denote the family of nonempty proper subsets of E. Let $\mathcal{B} = \{B_1, \ldots, B_n\}$. No special structure is assumed for the B_i's except that they are finite.

With the above notation established, we consider the following Integer Linear Program (ILP) in this section:

$$\min \left\{ \sum_{i \in E} w_i x_i : \sum_{i \in U} x_i \geq \left| \bigcap_{i \in E \backslash U} B_i \right|, \forall\, U \in \mathcal{F}, x_i \in \mathbb{Z} \right\}. \tag{5}$$

It is clear that any algorithm that efficiently solves this ILP also solves ILP (2) by putting $B_i \leftarrow P_i^c$.

A. Submodular Optimization

Our algorithm for solving ILP (5) relies heavily on submodular function optimization. To this end, we give a very brief introduction to submodular functions here.

A function $g : 2^E \to \mathbb{R}$ is said to be submodular if, for all $X, Y \in 2^E$,

$$g(X) + g(Y) \geq g(X \cap Y) + g(X \cup Y).$$

Over the past three decades, submodular function optimization has received a significant amount of attention. Notably, several polynomial time algorithms have been developed for solving the Submodular Function Minimization (SFM) problem

$$\min \{ g(U) : U \subseteq E \}.$$

We refer the reader to [13], [14], [17] for a comprehensive overview of SFM and known algorithms. As we will demonstrate, we can solve ILP (5) via an algorithm that iteratively calls a SFM routine. The most notable feature of SFM algorithms is their ability to solve problems with exponentially many constraints in polynomial time. One of the key drawbacks of SFM is that the problem formulation is very specific. Namely, SFM routines typically require the function g to be submodular on *all* subsets of the set E.

B. The Algorithm

We begin by developing an algorithm to solve an equality constrained version of ILP (5). We will remark on the general case at the conclusion of this section. To this end, let M be a positive integer and consider the following ILP:

$$\text{minimize} \quad w^T x \tag{6}$$

$$\text{subject to: } x(U) \geq \left| \bigcap_{i \in E \backslash U} B_i \right| \text{ for all } U \in \mathcal{F}, \text{ and} \tag{7}$$

$$x(E) = M. \tag{8}$$

Remark 1: We assume $w_i \geq 0$, else in the case without the equality constraint we could allow the corresponding $x_i \to +\infty$ and the problem is unbounded from below.

```
Algorithm IV.1: SolveILP($\mathcal{B}, E, M, w$)
  comment: Define $f : 2^E \to \mathbb{R}$ as in equation (9).
  $x \leftarrow$ ComputePotentialX($f, M, w$)
  if CheckFeasible($f, x$)
    then return ($x$)
    else return (Problem Infeasible)
```

Theorem 5: Algorithm IV.1 solves the equality constrained ILP (6) in polynomial time. If feasible, Algorithm IV.1 returns an optimal x. If infeasible, Algorithm IV.1 returns "Problem Infeasible".

Proof: The proof is accomplished in three steps:

1) First, we show that if our algorithm returns an x, it is feasible.
2) Second, we prove that if a returned x is feasible, it is also optimal.
3) Finally, we show that if our algorithm does not return an x, then the problem is infeasible.

Each step is given its own subsection. ■

Algorithm IV.1 relies on three basic subroutines given below:

```
Algorithm IV.2: COMPUTEPOTENTIALX($f, M, w$)

comment: If feasible, returns $x$ satisfying
         (7) and (8) that minimizes $w^T x$.
comment: Order elements of $E$ so that
         $w_1 \geq w_2 \geq \cdots \geq w_n$.
for $i \leftarrow n$ to 2
  do { comment: Define $f_i(U) := f(U+i)$ for
                $U \subseteq \{i, \ldots, n\}$.
       $x_i \leftarrow$ SFM($f_i, \{i, \ldots, n\}$)
$x_1 \leftarrow M - \sum_{i=2}^{n} x_i$
return ($x$)
```

```
Algorithm IV.3: CHECKFEASIBLE($f, x$)

comment: Check if $x(U) \leq f(U)$ for
         all $U \in \mathcal{F}$ with $1 \in U$.
comment: Define $f_1(U) := f(U+1)$ for $U \subseteq E$.
if SFM($f_1, E$) $< 0$
  then return ( false )
  else return ( true )
```

```
Algorithm IV.4: SFM($f, V$)

comment: Minimize submodular function $f$ over
         groundset $V$. See [17] for details.
$v \leftarrow \min\{f(U) : U \subseteq V\}$
return ($v$)
```

C. Feasibility of a Returned x

In this section, we prove that if Algorithm IV.1 returns a vector x, it must be feasible. We begin with some definitions.

Definition 1: A pair of sets $X, Y \subset E$ is called **crossing** if $X \cap Y \neq \emptyset$ and $X \cup Y \neq E$.

Definition 2: A function $g : 2^E \to \mathbb{R}$ is **crossing submodular** if

$$g(X) + g(Y) \geq g(X \cap Y) + g(X \cup Y)$$

for X, Y crossing.

We remark that minimization of crossing submodular functions is well established, however it involves a lengthy reduction to a standard submodular optimization problem. However, the crossing family $\mathcal{F}$ admits a straightforward algorithm, which is what we provide in Algorithm IV.1. We refer the reader to [13] for complete details on the general case.

For M a positive integer, define

$$f(U) := M - \left|\bigcap_{i \in U} B_i\right| - x(U), \text{ for } U \in \mathcal{F}. \tag{9}$$

Lemma 2: The function f is crossing submodular on $\mathcal{F}$.

Proof: For $X, Y \in \mathcal{F}$ crossing:

$$\begin{aligned} f(X) + f(Y) &= M - \left|\bigcap_{i \in X} B_i\right| - x(X) \\ &\quad + M - \left|\bigcap_{i \in Y} B_i\right| - x(Y) \\ &= M - \left|\bigcap_{i \in X} B_i\right| - x(X \cap Y) \\ &\quad + M - \left|\bigcap_{i \in Y} B_i\right| - x(X \cup Y) \\ &\geq M - \left|\bigcap_{i \in X \cap Y} B_i\right| - x(X \cap Y) \\ &\quad + M - \left|\bigcap_{i \in X \cup Y} B_i\right| - x(X \cup Y) \\ &= f(X \cap Y) + f(X \cup Y). \end{aligned}$$

■

Observe that, with f defined as above, the constraints of ILP (6) can be equivalently written as:

$$\begin{gathered} f(U) = M - \left|\bigcap_{i \in U} B_i\right| - x(U) \geq 0 \text{ for all } U \in \mathcal{F}, \text{ and} \\ x(E) = M. \end{gathered} \tag{10}$$

Without loss of generality, assume the elements of E are ordered lexicographically so that $w_1 \geq w_2 \geq \cdots \geq w_n$. At iteration i in Algorithm IV.2, $x_j = 0$ for all $j \leq i$. Thus, setting

$$\begin{aligned} x_i \leftarrow & \min_{U \subseteq \{i, \ldots, n\}} \{f_i(U)\} \\ &= \min_{U \subseteq \{i, \ldots, n\} : i \in U} \{f(U)\} \\ &= \min_{U \subseteq \{i, \ldots, n\} : i \in U} \left\{ M - \left|\bigcap_{i \in U} B_i\right| - x(U) \right\} \end{aligned} \tag{11}$$

and noting that the returned x satisfies $x(E) = M$, rearranging (11) guarantees that

$$x(E \backslash U) \geq \left|\bigcap_{i \in U} B_i\right|, \text{ for all } U \subseteq \{i, \ldots, n\}, i \in U \tag{12}$$

as desired. Iterating through $i \in \{2, \ldots, n\}$ guarantees (12) holds for $2 \leq i \leq n$.

Remark 2: In the feasibility check routine (Algorithm IV.3), we must be able to evaluate $f_1(E)$. The reader can verify that putting $f(E) = 0$ preserves submodularity.

Now, in order for the feasibility check to return **true**, we must have

$$\begin{aligned}\min_{U\subseteq E}\{f_1(U)\} &= \min_{U\subseteq E:1\in U}\{f(U)\}\\ &= \min_{U\subseteq E:1\in U}\left\{M-\left|\bigcap_{i\in U}B_i\right|-x(U)\right\}\\ &\geq 0,\end{aligned}$$

implying that

$$x(E\backslash U)\geq\left|\bigcap_{i\in U}B_i\right|,\text{ for all } U\subseteq E, 1\in U. \qquad (13)$$

Combining (12) and (13) and noting that $x(E)=M$ proves that x is indeed feasible. Moreover, x is integral as desired.

D. Optimality of a Returned x

In this section, we prove that if Algorithm IV.1 returns a feasible x, then it is also optimal. First, we require two more definitions and a lemma.

Definition 3: A constraint of the form (10) corresponding to U is said to be **tight** for U if

$$f(U)=M-\left|\bigcap_{i\in U}B_i\right|-x(U)=0.$$

Lemma 3: If x is feasible, X,Y are crossing, and their corresponding constraints are tight, then the constraints corresponding to $X\cap Y$ and $X\cup Y$ are also tight.

Proof: Since the constraints corresponding to X and Y are tight, we have

$$0=f(X)+f(Y)\geq f(X\cap Y)+f(X\cup Y)\geq 0.$$

The first inequality is due to submodularity and the last inequality holds since x is feasible. This implies the result. ■

Definition 4: A family of sets $\mathcal{L}$ is **laminar** if $X,Y\in\mathcal{L}$ implies either $X\cap Y=\emptyset$, $X\subset Y$, or $Y\subset X$.

At iteration k ($1<k\leq n$) of Algorithm IV.2, let U_k be the set where (11) achieves its minimum. Note that $k\in U_k\subseteq\{k,\dots,n\}$. By construction, the constraint corresponding to U_k is tight. Also, the constraint $x(E)=M$ is tight. From the U_k's and E we can construct a laminar family as follows: if $U_j\cap U_k\neq\emptyset$ for $j<k$, then replace U_j with $\tilde{U}_j\leftarrow U_k\cup U_j$. By Lemma 3, the constraints corresponding to the sets in the newly constructed laminar family are tight. Call this family $\mathcal{L}$. For each $i\in E$, there is a unique smallest set in $\mathcal{L}$ containing i. Denote this set L_i. Since $k\in U_k\subseteq\{k,\dots,n\}$, $L_i\neq L_j$ for $i\neq j$. Note that $L_1=E$ and $L_i\subset L_j$ only if $j<i$.

For each $L_i\in\mathcal{L}$ there is a unique smallest set L_j such that $L_i\subset L_j$. We call L_j the least upper bound on L_i.

Now, consider the dual linear program to (6):

$$\begin{aligned}\text{maximize}\quad & -\sum_{U\in\mathcal{F}}\pi_U\left(M-\left|\bigcap_{i\in U}B_i\right|\right)-\pi_E M\\ \text{subject to:}\quad & \sum_{U\in\mathcal{F}:i\in U}\pi_U+\pi_E+w_i=0,\ \text{for } 1\leq i\leq n\\ & \pi_U\geq 0 \text{ for } U\in\mathcal{F},\ \text{and } \pi_E \text{ free.}\end{aligned}$$

For each $L_i\in\mathcal{L}$, let the corresponding dual variable $\pi_{L_i}=w_j-w_i$, where L_j is the least upper bound on L_i. By construction, $\pi_{L_i}\geq 0$ since it was assumed that $w_1\geq\cdots\geq w_n$. Finally, let $\pi_E=-w_1$ and $\pi_U=0$ for $U\notin\mathcal{L}$.

Now, observe that:

$$\sum_{U\in\mathcal{F}:i\in U}\pi_U+\pi_E+w_i=0$$

as desired for each i. Thus, π is dual feasible. Finally, note that $\pi_U>0$ only if $U\in\mathcal{L}$. However, the primal constraints corresponding to the sets in $\mathcal{L}$ are tight. Thus, (x,π) form a primal-dual feasible pair satisfying complementary slackness conditions, and are therefore optimal.

E. No Returned x = Infeasibility

Finally, we prove that if the feasibility check returns **false**, then ILP (6) is infeasible. Note by construction that the vector x passed to the feasibility check satisfies

$$M-\left|\bigcap_{i\in U}B_i\right|-x(U)\geq 0 \text{ for all nonempty } U\subseteq\{2,\dots,n\},$$

and $x(E)=M$. Again, let U_k be the set where (11) achieves its minimum and let $\mathcal{L}$ be the laminar family generated by these U_k's and E exactly as before. Again, the constraints corresponding to the sets in $\mathcal{L}$ are tight (this can be verified in a manner identical to the proof of Lemma 3). Now, since x failed the feasibilty check, there exists some exceptional set T with $1\in T$ for which

$$M-\left|\bigcap_{i\in T}B_i\right|-x(T)<0.$$

Generate a set L_T as follows: Initialize $L_T\leftarrow T$. For each $L_i\in\mathcal{L}, L_i\neq E$, if $L_T\cap L_i\neq\emptyset$, update $L_T\leftarrow L_T\cup L_i$. Now, we can add L_T to family $\mathcal{L}$ while preserving the laminar property. We pause to make two observations:

1) By an argument similar to the proof of Lemma 3, we have that

$$M-\left|\bigcap_{i\in L_T}B_i\right|-x(L_T)<0.$$

2) The sets in $\mathcal{L}$ whose least upper bound is E form a partition of E. We note that L_T is a nonempty class of this partition. Call this partition $\mathcal{P}_{\mathcal{L}}$.

Again consider the dual constraints, however, let $w_i=0$ (this does not affect feasibility). For each $L\in\mathcal{P}_{\mathcal{L}}$ define the associated dual variable $\pi_L=\alpha$, and let $\pi_E=-\alpha$. All other dual variables are set to zero. It is easy to check that this π is dual feasible. Now, the dual objective function becomes:

$$\begin{aligned}
&-\sum_{U\in\mathcal{F}}\pi_U\left((M-\left|\bigcap_{i\in U}B_i\right|\right)-\pi_E M\\
&=-\alpha\sum_{L\in\mathcal{P}_{\mathcal{L}}}\left(M-\left|\bigcap_{i\in L}B_i\right|-x(L)+x(L)\right)+\alpha M\\
&=-\alpha\left(M-\left|\bigcap_{i\in L_T}B_i\right|-x(L_T)\right)-\alpha x(E)+\alpha M\\
&=-\alpha\left(M-\left|\bigcap_{i\in L_T}B_i\right|-x(L_T)\right)\\
&\to+\infty \text{ as } \alpha\to\infty.
\end{aligned}$$

Thus, the dual is unbounded and therefore the primal problem must be infeasible.

As an immediate corollary we obtain the following:

Corollary 1: The optimal value of the ILP:

$$\min\left\{x(E):x(U)\geq\left|\cap_{i\in E\setminus U}B_i\right|,U\in\mathcal{F},x_i\in\mathbb{Z}\right\}$$

and the corresponding LP relaxation:

$$\min\left\{x(E):x(U)\geq\left|\cap_{i\in E\setminus U}B_i\right|,U\in\mathcal{F},x_i\in\mathbb{R}\right\}$$

differ by less than 1.

Proof: Algorithm IV.1 is guaranteed to return an optimal x if the intersection of the polytope and the hyperplane $x(E)=M$ is nonempty. Thus, if M^* is the minimum such M, then the optimal value of the LP must be greater than M^*-1. ■

F. Solving the General ILP

Finally, we remark on how to solve the general case of the ILP without the equality constraint given in (5). First, we state a simple convexity result.

Lemma 4: Let $p_w^*(M)$ denote the optimal value of ILP (6) when the equality constraint is $x(E)=M$. We claim that $p_w^*(M)$ is a convex function of M.

Proof: Let M_1 and M_2 be integers and let $\theta\in[0,1]$ be such that $M_\theta=\theta M_1+(1-\theta)M_2$ is an integer. Let $x^{(1)}$ and be $x^{(2)}$ optimal vectors that attain $p_w^*(M_1)$ and $p_w^*(M_2)$ respectively. Let $x^{(\theta)}=\theta x^{(1)}+(1-\theta)x^{(2)}$. By convexity, $x^{(\theta)}$ is feasible, though not necessarily integer. However, by the results from above, optimality is always attained by an integral vector. Thus, it follows that:

$$\begin{aligned}
\theta p_w^*(M_1)+(1-\theta)p_w^*(M_2)&=\theta w^Tx^{(1)}+(1-\theta)w^Tx^{(2)}\\
&=w^Tx^{(\theta)}\geq p_w^*(M_\theta).
\end{aligned}$$

■

Noting that $p_w^*(M)$ is convex in M, we can perform bisection on M to solve the ILP in the general case. For our purposes, it suffices to have relatively loose upper and lower bounds on M since the complexity only grows logarithmically in the difference. A simple lower bound on M is given by $M\geq\max_i|B_i|$.

G. Complexity

Our aim in this paper is not to give a detailed complexity analysis of our algorithm. This is due to the fact that the complexity is dominated by the the SFM over the set E in Algorithm IV.3. Therefore, the complexity of Algorithm IV.1 is essentially the same as the complexity of the SFM solver employed.

However, we have performed a series of numerical experiments to demonstrate that Algorithm IV.1 performs quite well in practice. In our implementation, we ran the Fujishige-Wolfe (FW) algorithm for SFM [15] based largely on a Matlab routine by A. Krause [16]. While the FW algorithm has not been proven to run in polynomial time, it has been shown to work quite well in practice [15] (similar to the Simplex algorithm for solving Linear Programs). Whether or not FW has worst-case polynomial complexity is an open problem to date. We remark that there are several SFM algorithms that run in strongly polynomial time which could be used if a particular application requires polynomially bounded worst-case complexity [17].

In our series of experiments, we chose $B_i\subset F$ randomly, where $|F|=50$. We let $n=|E|$ range from 10 to 190 in increments of 10. For each value of n, we ran 10 experiments. The average computation time is shown in Figure 1, with error bars indicating one standard deviation. We consistently observed that the computations run in approximately $O(n^{1.85})$ time. Due to the iterative nature of the SFM algorithm, we anticipate that the computation time could be significantly reduced by implementing the algorithm in C/C++ instead of Matlab. However, the $O(n^{1.85})$ trend should remain the same. Regardless, we are able to solve the ILP problems under consideration with an astonishing 2^{190} constraints in approximately one minute.

V. Concluding Remarks

In this paper, we derive an efficient algorithm that solves an Integer Linear Program of a particular form. We apply our algorithm to the weighted universal recovery problem and the secrecy generation problem for the network described in the introduction. Our algorithm produces exact results for each of these problems in the non-asymptotic regime. Due to the general form of the ILP that is solved, our algorithm may be of interest beyond the applications given here.

References

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, “Network information flow”, IEEE Transactions on Information Theory, July 2000.

[2] S.-Y. R. Li, R. W. Yeung, and N. Cai, “Linear network coding”, IEEE Transactions on Information Theory, Feb. 2003.

[3] S. El Rouayheb, A. Sprintson, and P. Sadeghi, On coding for cooperative data exchange, in Proc. Information Theory Workshop, Cairo, Egypt, 2009, pp. 118-122.

[4] T. Courtade, B. Xie, and R. Wesel, “Optimal Exchange of Packets for Universal Recovery in Broadcast Networks,” MILCOM 2010, San Jose, CA, October 31 - November 3, 2010.

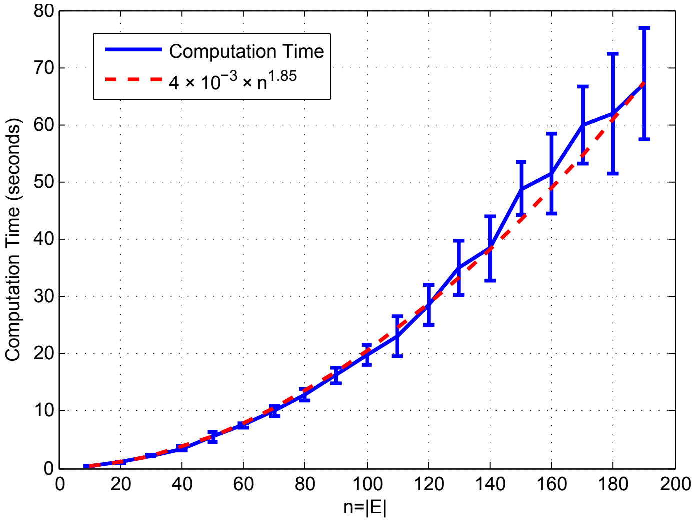

Fig. 1. Experimental results. For the red dotted line, the multiplicative constant α and exponent β were chosen to minimize the MSE $\sum_{i=1}^{n} |\log(\alpha n^{\beta}) - \log(\hat{m}_n)|^2$, where $\hat{m}_n$ is the sample mean of the computation times for $|E| = n$.

[5] T. A. Courtade and R. D. Wesel, "Efficient Universal Recovery in Broadcast Networks". Forty-Eighth Annual Allerton Conference on Communication, Control, and Computing: Monticello, IL, Sept. 29 - Oct. 1, 2010.

[6] A. Sprintson, P. Sadeghi, G. Booker, and S. El Rouayheb. A Randomized Algorithm and Performance Bounds for Coded Cooperative Data Exchange. In 2010 IEEE International Symposium on Information Theory Proceedings (ISIT 2010), Austin, Texas, USA, June 2010.

[7] A. Sprintson, P. Sadeghi, G. Booker, and S. El Rouayheb. Deterministic Algorithm for Coded Cooperative Data Exchange. ICST QShine, Houston, Texas, November 17-19, 2010.

[8] I. Csiszár and P. Narayan. Secrecy Capacities for Multiple Terminals. IEEE Trans. IT, Vol. 50, No. 12, Dec. 2004: 3047-3061.

[9] Y. Chunxuan, P. Narayan. Secret key and private key constructions for simple multiterminal source models, ISIT 2005, pp.2133-2137, 4-9 Sept. 2005.

[10] Y. Chunxuan and A. Reznik. A simple secret key construction system for broadcasting model. 44th Annual Conference on Information Sciences and Systems (CISS), 2010.

[11] I. Csiszár and J. Körner, Information Theory: Coding Theorems for Discrete Memoryless Systems. Academic, New York, N.Y., 1982.

[12] Sidharth Jaggi, Peter Sanders, Philip A. Chou, Michelle Effros, Sebastian Egner, Kamal Jain, Ludo M. G. M. Tolhuizen. Polynomial time algorithms for multicast network code construction. IEEE Transactions on Information Theory, 2005: 1973-1982.

[13] A. Schrijver. Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin. 2003.

[14] S. Fujishige. Submodular Functions and Optimization. Second Edition. North-Holland, 2010.

[15] S. Fujishige, T. Hayashi and S. Isotani. The Minimum-Norm-Point Algorithm Applied to Submodular Function Minimization. Kyoto University, Kyoto Japan, 2006.

[16] A. Krause. SFO: A Toolbox for Submodular Function Optimization. Journal of Machine Learning Research (2010).

[17] S. T. McCormick. Submodular Function Minimization. In Discrete Optimization, K. Aardal, G. Nemhauser, and R. Weismantel, eds. Handbooks in Operations Research and Management Science, Volume 12. Elsevier. (2005).

[18] S. Boyd and L. Vandenberghe. Convex Optimization. Cambridge University Press, 2004.

[19] C. Fragouli, J. Widmer, and J.-Y. L. Boudec, "A network coding approach to energy efficient broadcasting: From theory to practice", in IEEE INFOCOM, Barcelona, Spain, Apr. 2006.

[20] C. Fragouli, J. Widmer, and J.-Y. L. Boudec, "Efficient Broadcasting Using Network Coding," IEEE/ACM Transactions on Networking, Vol. 16, No. 2, April 2008, 450-463.

[21] S. El Rouayheb, M.A.R. Chaudhry, and A. Sprintson. On the minimum number of transmissions in single-hop wireless coding networks. In IEEE Information Theory Workshop (Lake Tahoe), 2007.

[22] S. El Rouayheb, A. Sprintson, and C. N. Georghiades. On the relation between the index coding and the network coding problems. Proc. of IEEE International Symposium on Information Theory (ISIT08), 2008.