Alphabet Size Reduction for Secure Network Coding:

A Graph Theoretic Approach

Xuan Guang

joint work with Raymond W. Yeung

Institute of Network Coding
The Chinese University of Hong Kong, Hong Kong SAR, China

INC Seminar

Dec. 7, 2016

Outline

Outline

Wiretap Network

- Let $G = (V, E)$ be a finite directed acyclic network with a single source node s and a set of sink nodes $T \subset V \setminus \{s\}$, where
  - V is the set of nodes, and
  - E is the set of edges.
- Parallel edges between two adjacent nodes are allowed.
- An index taken from an alphabet can be transmitted on each edge in E.

Wiretap Network

- Let $G = (V, E)$ be a finite directed acyclic network with a single source node s and a set of sink nodes $T \subset V \setminus \{s\}$, where
  - V is the set of nodes, and
  - E is the set of edges.
- Parallel edges between two adjacent nodes are allowed.
- An index taken from an alphabet can be transmitted on each edge in E.
- Let $\mathscr{A}$ be a collection of subsets of E, where every edge set in $\mathscr{A}$ is called a wiretap set.

Wiretap Network

A wiretap network is a quadruple $(G, s, T, \mathscr{A})$, where

- s generates a random source message M according to an arbitrary distribution on a message set $\mathcal{M}$;
- each $t \in T$ is required to recover the source message M with zero error;
- arbitrary one wiretap set in $\mathscr{A}$, but no more than one, may be fully accessed by a wiretapper;
- $\mathscr{A}$ is known by s and all $t \in T$ but which wiretap set in $\mathscr{A}$ is actually eavesdropped is unknown.

Wiretap Network

- It is necessary to randomize the source message to combat the wiretapper.
- The random key K available at the source node is a random variable that takes values in a set of keys $\mathcal{K}$ according to the uniform distribution.

Secure Network Codes

- Let $\mathcal{F}$ be an alphabet.
- An $\mathcal{F}$-valued secure network code on a wiretap network $(G, s, T, \mathscr{A})$ consists of a set of local encoding mappings $\{\phi_e : e \in E\}$ such that
  - if $e \in \mathrm{Out}(s)$,

$$\phi_e : \ \mathcal{M} \times \mathcal{K} \to \mathcal{F};$$

  - otherwise, i.e., if $e \in \mathrm{Out}(v)$ for a node $v \in V \setminus \{s\}$,

$$\phi_e : \ \mathcal{F}^{|\mathrm{In}(v)|} \to \mathcal{F}.$$

Secure Network Codes

Definition 1

For a secure network code on the wiretap network $(G, s, T, \mathscr{A})$, $I(Y_A; M) = 0$ *for every wiretap set* $A \in \mathscr{A}$, *where* $I(Y_A; M)$ *denotes the mutual information between* $Y_A = (Y_e : e \in A)$ *and* M.

The Required Alphabet Size

Proposition 1 ([Cai & Yeung][1])

Let $(G, s, T, \mathscr{A})$ *be a wiretap network and* $\mathcal{F}$ *be an alphabet with* $|\mathcal{F}| \geq |T|$*, the number of sink nodes in* G*. Then there exists an* $\mathcal{F}$*-valued secure network code over* $(G, s, T, \mathscr{A})$ *provided that* $|\mathcal{F}| > |\mathscr{A}|$*.*

[1] N. Cai and R. W. Yeung, "Secure Network Coding on a Wiretap Network," *IEEE Trans. Inf. Theory*, 2011.

The Required Alphabet Size

- The lower bound $|\mathscr{A}|$ on the required alphabet size is typically too large for implementation in terms of computational complexity and storage requirement.
- Reduction of the required alphabet size is a problem not only of theoretical interest but also of practical importance.

An Assumption

Assume that all wiretap sets are regular.

- A wiretap set A is said to be regular, if $|A| = \mathrm{mincut}(s, A)$.
- The collection of wiretap sets $\mathscr{A}$ is said to be regular, if all wiretap sets in $\mathscr{A}$ are regular.

An Assumption

Assume that all wiretap sets are regular.

- A wiretap set A is said to be regular, if $|A| = \mathrm{mincut}(s, A)$.
- The collection of wiretap sets $\mathscr{A}$ is said to be regular, if all wiretap sets in $\mathscr{A}$ are regular.
- Replace non-regular wiretap sets in $\mathscr{A}$ by their minimum cuts (that are regular) to form $\mathscr{A}'$.
- A secure network code that is secure for $\mathscr{A}'$ is also secure for $\mathscr{A}$.

Outline

Equivalence Relation "$\sim$"

- Let $(G, s, T, \mathscr{A})$ be a wiretap network.
- The binary relation "$\sim$":

  For any two edge sets A and A' in G, we write $A \sim A'$ provided that

  - there exists an edge set CUT that is a minimum cut between s and A and also between s and A'.

Equivalence Relation "$\sim$"

Proposition 2 ([Guang et al.][2])

The binary relation "$\sim$" is an equivalence relation. To be specific, For any three edge sets A, A', and A'' in G:

1. **(Reflexivity)** $A \sim A$;
2. **(Symmetry)** *if* $A \sim A'$ *then* $A' \sim A$;
3. **(Transitivity)** *if* $A \sim A'$ *and* $A' \sim A''$, $A \sim A''$.

[2] X. Guang, J. Lu, and F.-W. Fu, "Small field size for secure network coding", *IEEE Commun. Lett.*, 2015.

Equivalence Relation "$\sim$"

Proposition 3

Let $A_1, A_2, \cdots, A_m$ *be* m *equivalent edge sets under the equivalence relation "*$\sim$*". Then*

$$\text{mincut}(s, \cup_{i=1}^{m} A_i) = \text{mincut}(s, A_j), \quad \forall j,\ 1 \le j \le m.$$

Equivalence Relation "$\sim$"

Proposition 3

Let $A_1, A_2, \cdots, A_m$ *be* m *equivalent edge sets under the equivalence relation "*$\sim$*". Then*

$$\mathrm{mincut}(s, \cup_{i=1}^{m} A_i) = \mathrm{mincut}(s, A_j), \quad \forall j, \ 1 \leq j \leq m.$$

- With "$\sim$", the wiretap sets in $\mathscr{A}$ can be partitioned into equivalence classes.
- All the wiretap sets in an equivalence class have a common minimum cut.

The Required Alphabet Size

Denote $N(\mathscr{A})$ by the number of the equivalence classes in $\mathscr{A}$.

Theorem 4

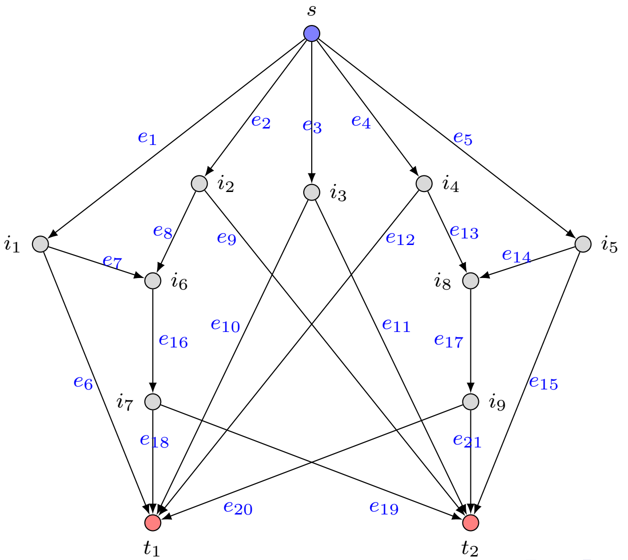
Let $(G, s, T, \mathscr{A})$ be a wiretap network and $\mathcal{F}$ be an alphabet with $|\mathcal{F}| \geq |T|$. Then there exists an $\mathcal{F}$-valued secure network code over $(G, s, T, \mathscr{A})$ provided that

$$|\mathcal{F}| > N(\mathscr{A}).$$

- This lower bound $N(\mathscr{A})$ was originally obtained in [Guang *et al.*][3] for r-wiretap networks, but it also applies for general wiretap networks.

[3] X. Guang, J. Lu, and F.-W. Fu, "Small field size for secure network coding", *IEEE Commun. Lett.*, 2015.

Example



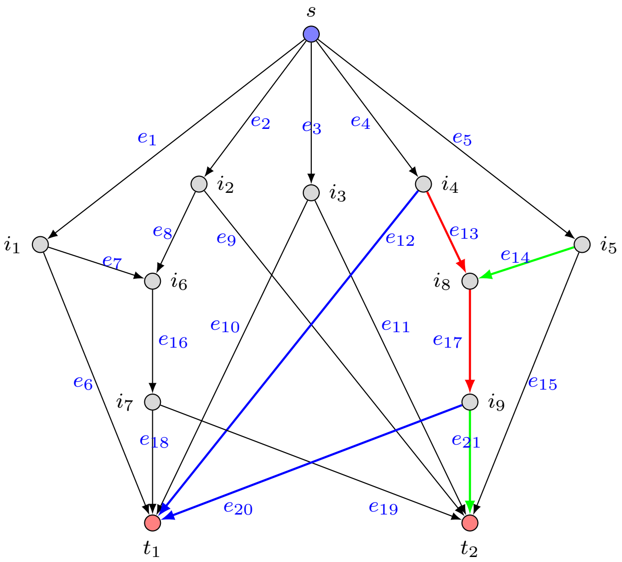
s
e1
e2
e3
e4
e5
i2
i3
i4
i1
i5
e8
e9
e12
e13
e7
e14
i6
i8
e10
e11
e16
e17
e6
e15
i7
i9
e18
e21
e20
e19
t1
t2

- Let the collection of wiretap sets $\mathscr{A}$ be:

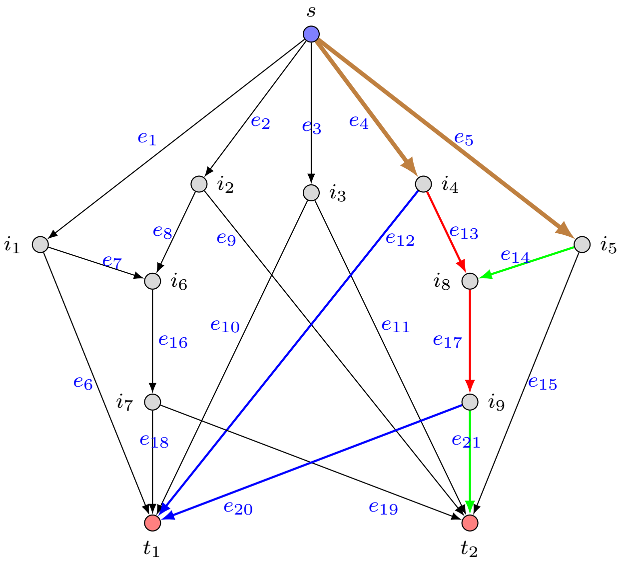
$$
\begin{aligned}
\mathscr{A} = \Big\{ & \{e_6\}, \{e_7\}, \{e_8\}, \{e_9\}, \{e_{12}\}, \{e_{13}\}, \\
& \{e_{14}\}, \{e_{15}\}, \{e_{18}\}, \{e_{19}\}, \{e_{20}\}, \{e_{21}\}, \\
& \{e_6, e_{18}\}, \{e_6, e_{19}\}, \{e_7, e_{18}\}, \{e_7, e_{19}\}, \{e_8, e_{11}\}, \\
& \{e_8, e_{16}\}, \{e_8, e_{18}\}, \{e_9, e_{10}\}, \{e_9, e_{18}\}, \{e_9, e_{19}\}, \\
& \{e_{10}, e_{14}\}, \{e_{10}, e_{15}\}, \{e_{10}, e_{19}\}, \{e_{10}, e_{21}\}, \{e_{11}, e_{14}\}, \\
& \{e_{11}, e_{15}\}, \{e_{11}, e_{18}\}, \{e_{11}, e_{20}\}, \{e_{12}, e_{20}\}, \{e_{12}, e_{21}\}, \\
& \{e_{13}, e_{17}\}, \{e_{13}, e_{21}\}, \{e_{14}, e_{20}\}, \{e_{14}, e_{21}\}, \{e_{15}, e_{20}\}, \\
& \{e_{15}, e_{21}\}, \{e_{18}, e_{20}\}, \{e_{18}, e_{21}\}, \{e_{19}, e_{20}\}, \{e_{19}, e_{21}\}, \\
& \{e_1, e_3, e_{16}\}, \{e_1, e_{11}, e_{16}\}, \{e_2, e_{10}, e_{16}\}, \\
& \{e_3, e_5, e_{17}\}, \{e_4, e_{10}, e_{17}\}, \{e_5, e_{11}, e_{17}\} \Big\}.
\end{aligned}
$$

- $|\mathscr{A}| = 48$.

e.g., consider three wiretap sets $\{e_{12}, e_{20}\}$, $\{e_{13}, e_{17}\}$, $\{e_{14}, e_{21}\}$.



e.g., consider three wiretap sets $\{e_{12}, e_{20}\}$, $\{e_{13}, e_{17}\}$, $\{e_{14}, e_{21}\}$.



Example

- The equivalence classes of wiretap sets are:

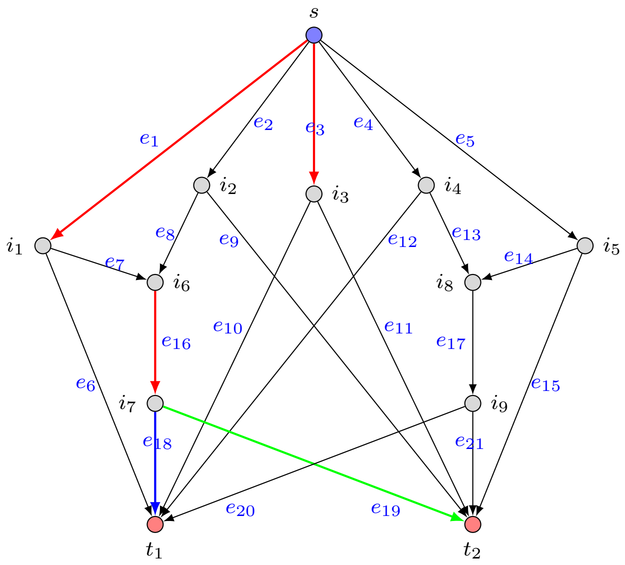
$$\begin{aligned}
&\mathrm{Cl}_1 = \Big\{\{e_6\},\{e_7\}\Big\}, \quad \mathrm{Cl}_2 = \Big\{\{e_8\},\{e_9\}\Big\},\\
&\mathrm{Cl}_3 = \Big\{\{e_{12}\},\{e_{13}\}\Big\}, \ \mathrm{Cl}_4 = \Big\{\{e_{14}\},\{e_{15}\}\Big\},\\
&\mathrm{Cl}_5 = \Big\{\{e_{18}\},\{e_{19}\}\Big\}, \ \mathrm{Cl}_6 = \Big\{\{e_{20}\},\{e_{21}\}\Big\},\\
&\mathrm{Cl}_7 = \Big\{\{e_8,e_{11}\},\{e_9,e_{10}\}\Big\},\\
&\mathrm{Cl}_8 = \Big\{\{e_{10},e_{19}\},\{e_{11},e_{18}\}\Big\},\\
&\mathrm{Cl}_9 = \Big\{\{e_{10},e_{21}\},\{e_{11},e_{20}\}\Big\},\\
&\mathrm{Cl}_{10} = \Big\{\{e_{10},e_{14}\},\{e_{10},e_{15}\},\{e_{11},e_{14}\},\{e_{11},e_{15}\}\Big\},\\
&\mathrm{Cl}_{11} = \Big\{\{e_{18},e_{20}\},\{e_{18},e_{21}\},\{e_{19},e_{20}\},\{e_{19},e_{21}\}\Big\};
\end{aligned}$$

Example

$$\begin{aligned}
\mathrm{Cl}_{12} =& \Big\{ \{e_6, e_{18}\}, \{e_6, e_{19}\}, \{e_7, e_{18}\}, \{e_7, e_{19}\}, \\
& \{e_8, e_{16}\}, \{e_8, e_{18}\}, \{e_9, e_{18}\}, \{e_9, e_{19}\} \Big\}, \\
\mathrm{Cl}_{13} =& \Big\{ \{e_{12}, e_{20}\}, \{e_{12}, e_{21}\}, \{e_{13}, e_{17}\}, \{e_{13}, e_{21}\}, \\
& \{e_{14}, e_{20}\}, \{e_{14}, e_{21}\}, \{e_{15}, e_{20}\}, \{e_{15}, e_{21}\} \Big\}, \\
\mathrm{Cl}_{14} =& \Big\{ \{e_1, e_3, e_{16}\}, \{e_1, e_{11}, e_{16}\}, \{e_2, e_{10}, e_{16}\} \Big\}, \\
\mathrm{Cl}_{15} =& \Big\{ \{e_3, e_5, e_{17}\}, \{e_4, e_{10}, e_{17}\}, \{e_5, e_{11}, e_{17}\} \Big\}.
\end{aligned}$$

- Then $N(\mathscr{A}) = 15$ $(< |\mathscr{A}| = 48)$.

Furthermore, consider $\mathrm{Cl}_5 = \{\{e_{18}\}, \{e_{19}\}\}$ and $\{e_1, e_3, e_{16}\}$.



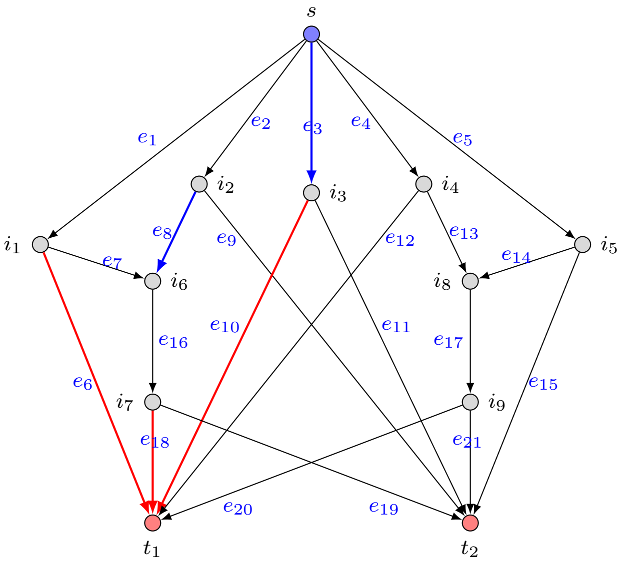
Wiretap-Set Domination

Definition 2 (Wiretap-Set Domination)

Let A_1 and A_2 be two wiretap sets in $\mathscr{A}$ with $|A_1| < |A_2|$.
We say that A_1 is dominated by A_2, denoted by $A_1 \prec A_2$, if there exists a minimum cut between s and A_2 that also separates A_1 from s. In other words, upon deleting the edges in the minimum cut between s and A_2, s and A_1 are also disconnected.

- Note that $A_1 \prec A_2$ does not mean that A_2 is at the "upstream" of A_1.

Let $A_1 = \{e_3, e_8\}$ and $A_2 = \{e_6, e_{10}, e_{18}\}$, and $A_1 \prec A_2$.



Equivalence-Class Domination

Definition 3 (Equivalence-Class Domination)

For two distinct equivalence classes Cl_1 and Cl_2, if there exists a common minimum cut of the wiretap sets in Cl_2 that separates all the wiretap sets in Cl_1 from s, we say that Cl_1 is dominated by Cl_2, denoted by $\mathrm{Cl}_1 \prec \mathrm{Cl}_2$.

Equivalence-Class Domination

Theorem 5

$\mathrm{Cl}(A_1) \prec \mathrm{Cl}(A_2)$ *if and only if* $A_1 \prec A_2$.

Equivalence-Class Domination

Theorem 6

The equivalence-class domination relation "$\prec$" amongst the equivalence classes in $\mathscr{A}$ is a strict partial order. Specifically, let Cl_1, Cl_2, and Cl_3 be three arbitrary equivalence classes, and then

1. **(Irreflexivity)** $\mathrm{Cl}_1 \nprec \mathrm{Cl}_1$;
2. **(Transitivity)** *if* $\mathrm{Cl}_1 \prec \mathrm{Cl}_2$ *and* $\mathrm{Cl}_2 \prec \mathrm{Cl}_3$, *then* $\mathrm{Cl}_1 \prec \mathrm{Cl}_3$;
3. **(Asymmetry)** *if* $\mathrm{Cl}_1 \prec \mathrm{Cl}_2$, *then* $\mathrm{Cl}_2 \nprec \mathrm{Cl}_1$.

Maximal Equivalence Class

- Now, the set of all the equivalence classes in $\mathscr{A}$ can be considered as a strictly partially ordered set.
- Thus, we can define its maximal equivalence classes.

Definition 4 (Maximal Equivalence Class)

For a collection of wiretap sets $\mathscr{A}$, an equivalence class Cl is a maximal equivalence class if there exists no other equivalence class Cl' such that $\mathrm{Cl}' \succ \mathrm{Cl}$.
Denote by $N_{\max}(\mathscr{A})$ the number of the maximal equivalence classes with respect to $\mathscr{A}$.

The Required Alphabet Size

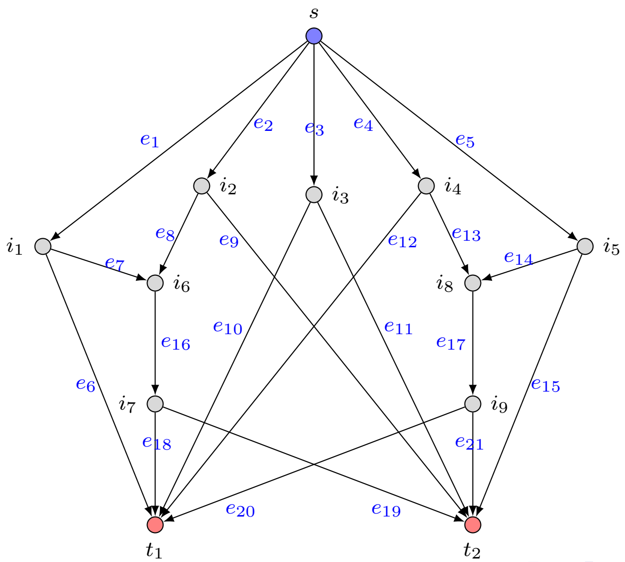
Theorem 7

Let $(G, s, T, \mathscr{A})$ *be a wiretap network and* $\mathcal{F}$ *be an alphabet with* $|\mathcal{F}| \geq |T|$*. Then there exists an* $\mathcal{F}$*-valued secure network code on* $(G, s, T, \mathscr{A})$ *provided that the alphabet size*

$$|\mathcal{F}| > N_{\max}(\mathscr{A}).$$

- $N_{\max}(\mathscr{A}) \leq N(\mathscr{A}) \leq |\mathscr{A}|$.

Example (Cont.)



Example (Cont.)

The Hasse diagram of all 15 equivalence classes, ordered by the equivalence-class domination relation "$\prec$".

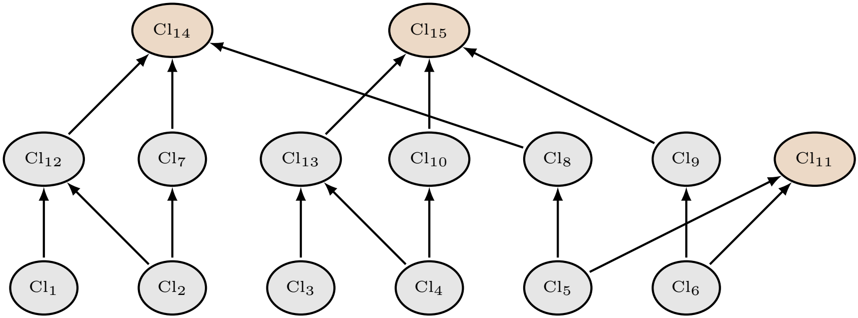


Figure: Cl_{11}, Cl_{14}, and Cl_{15} are all of the maximal equivalence classes.

Example (Cont.)

The alphabet size $\|\mathcal{F}\|$	
Lower Bound I: $\|\mathscr{A}\|$	48
Lower Bound II: $N(\mathscr{A})$	15
Lower Bound III: $N_{\max}(\mathscr{A})$	3

- The improvement of $N_{\max}(\mathscr{A})$ over $N(\mathscr{A})$ can be unbounded.

New Problem Proposed

- $N_{\max}(\mathscr{A})$ is graph-theoretical.
- $N_{\max}(\mathscr{A})$ only depends on the topology of the network G and the collection $\mathscr{A}$ of wiretap sets.

New Problem Proposed

- $N_{\max}(\mathscr{A})$ is graph-theoretical.
- $N_{\max}(\mathscr{A})$ only depends on the topology of the network G and the collection $\mathscr{A}$ of wiretap sets.
- In general, computing the value of $N_{\max}(\mathscr{A})$, or characterizing the corresponding Hasse diagram, is nontrivial.
- Even in the simple example, its value is not obvious.

New Problem Proposed

- $N_{\max}(\mathscr{A})$ is graph-theoretical.
- $N_{\max}(\mathscr{A})$ only depends on the topology of the network G and the collection $\mathscr{A}$ of wiretap sets.
- In general, computing the value of $N_{\max}(\mathscr{A})$, or characterizing the corresponding Hasse diagram, is nontrivial.
- Even in the simple example, its value is not obvious.
- **How to efficiently compute $N_{\max}(\mathscr{A})$?**

Outline

Primary Minimum Cut

Definition 5 (Primary Minimum Cut)

A minimum cut between the source node s and a sink node t in G is primary, if it separates s and all the minimum cuts between s and t.
In other words, a primary minimum cut between s and t is a common minimum cut of all the minimum cuts between s and t.

- The notion of primary minimum cut is crucial to the development of our algorithm.

Existence and Uniqueness of Primary Minimum Cut

Theorem 8

The primary minimum cut is well-defined, that is, the primary minimum cut between the source node s and a sink node t exists and is unique.

- The concept of the primary minimum cut between the source node s and a sink node t can be extended to between s and a wiretap set $A \in \mathscr{A}$.

Cornerstone

Theorem 9

In a wiretap network $(G, s, T, \mathscr{A})$, *let* Cl *be an arbitrary equivalence class of the wiretap sets. Then*

1. *all the wiretap sets in* Cl *have the same primary minimum cut, which hence is called the primary minimum cut of the equivalence class* Cl, *and*
2. *for every equivalence class* Cl' *with* $\mathrm{Cl}' \prec \mathrm{Cl}$, *the primary minimum cut of* Cl *separates all the wiretap sets in* Cl' *from* s.

Cornerstone

- To compute $N_{\max}(\mathscr{A})$, it suffices to compute the primary minimum cuts of all the maximal equivalence classes.
- With this, we bypass the complicated operations for determining the equivalence classes of wiretap sets and the domination relation among them.
- This is the key to the efficiency of the algorithm.

Algorithm for computing $N_{\max}(\mathscr{A})$:

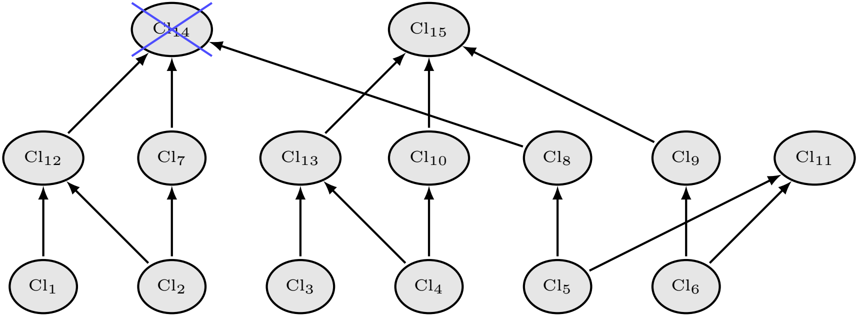
1. Define a set $\mathscr{B}$, and initialize $\mathscr{B}$ to the empty set.
2. Arbitrarily choose a wiretap set $A \in \mathscr{A}$ that has the largest cardinality in $\mathscr{A}$. Find the primary minimum cut between s and A, and call it CUT.
3. Partition the edge set E into two disjoint subsets: E_{CUT} and $E^c_{\mathrm{CUT}} \triangleq E \setminus E_{\mathrm{CUT}}$, where E_{CUT} is the set of the edges reachable from the source node s upon deleting the edges in CUT.
4. Remove all the wiretap sets in $\mathscr{A}$ that are subsets of E^c_{CUT} and add the primary minimum cut CUT to $\mathscr{B}$.
5. Repeat Steps 2) to 4) until $\mathscr{A}$ is empty and output $\mathscr{B}$, where $N_{\max}(\mathscr{A}) = |\mathscr{B}|$.

Algorithm

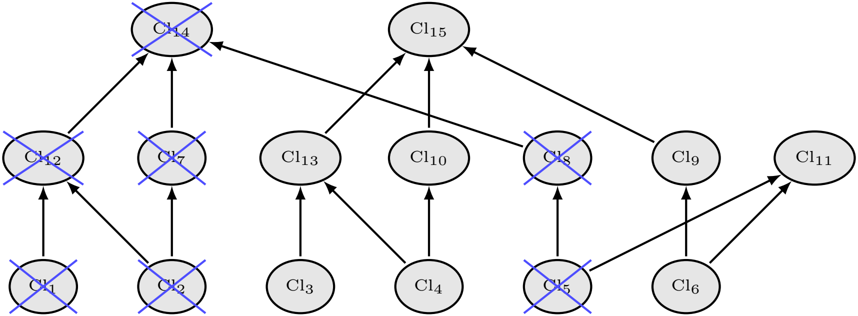
Algorithm for computing $N(\mathscr{A})$:

1. Define a set $\mathscr{B}$, and initialize $\mathscr{B}$ to the empty set.
2. Arbitrarily choose a wiretap set $A \in \mathscr{A}$ that has the largest cardinality in $\mathscr{A}$. Find the primary minimum cut between s and A, and call it CUT.
3. Partition the edge set E into two disjoint subsets: E_{CUT} and $E_{\mathrm{CUT}}^c \triangleq E \setminus E_{\mathrm{CUT}}$, where E_{CUT} is the set of the edges reachable from the source node s upon deleting the edges in CUT.
4. Remove all the wiretap sets of the same cardinality as A in $\mathscr{A}$ that are subsets of E_{CUT}^c. Add the primary minimum cut CUT to $\mathscr{B}$.
5. Repeat Steps 2) to 4) until $\mathscr{A}$ is empty and output $\mathscr{B}$, where $N(\mathscr{A}) = |\mathscr{B}|$.

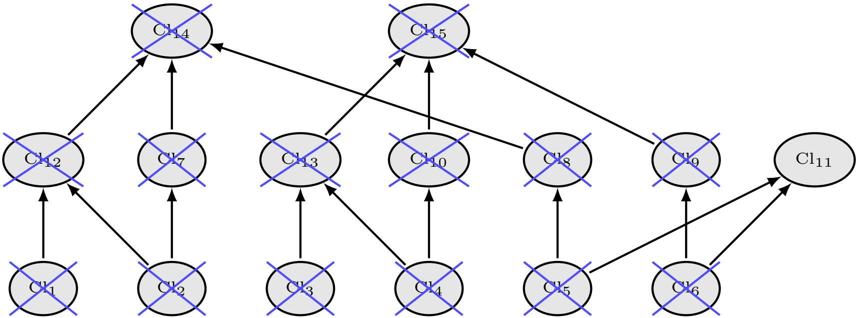
Example (Cont.)



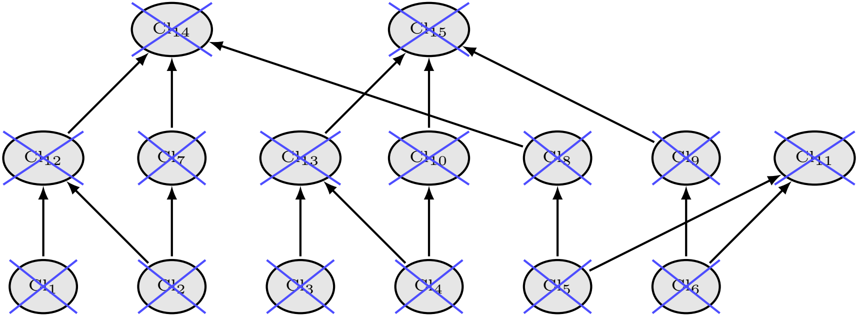
Example (Cont.)



Example (Cont.)



Example (Cont.)



Algorithm (Without Regular Assumption)

Algorithm modified for computing $N_{\max}(\mathscr{A})$ without regular assumption:

1. Define a set $\mathscr{B}$, and initialize $\mathscr{B}$ to the empty set.
2. Arbitrarily choose a wiretap set $A \in \mathscr{A}$ ($\mathscr{A}$ is not necessary regular) that has ~~**the largest cardinality**~~ **the largest minimum cut capacity** in $\mathscr{A}$. Find the primary minimum cut between s and A, and call it CUT.
3. Partition the edge set E into two disjoint subsets: E_{CUT} and $E^c_{\mathrm{CUT}} \triangleq E \setminus E_{\mathrm{CUT}}$, where E_{CUT} is the set of the edges reachable from the source node s upon deleting the edges in CUT.
4. Remove all the wiretap sets in $\mathscr{A}$ that are subsets of E^c_{CUT} and add the primary minimum cut CUT to $\mathscr{B}$.
5. Repeat Steps 2) to 4) until $\mathscr{A}$ is empty and output $\mathscr{B}$, where $N_{\max}(\mathscr{A}) = |\mathscr{B}|$.

Algorithm (Without Regular Assumption)

However,

- This modification requires pre-computing the minimum cut capacity of every wiretap set in $\mathscr{A}$.
- This will significantly increase the computational complexity of the algorithm when $|\mathscr{A}|$ is large.

Algorithm (Without Regular Assumption)

Algorithm II modified for computing $N_{\max}(\mathscr{A})$ without regular assumption:

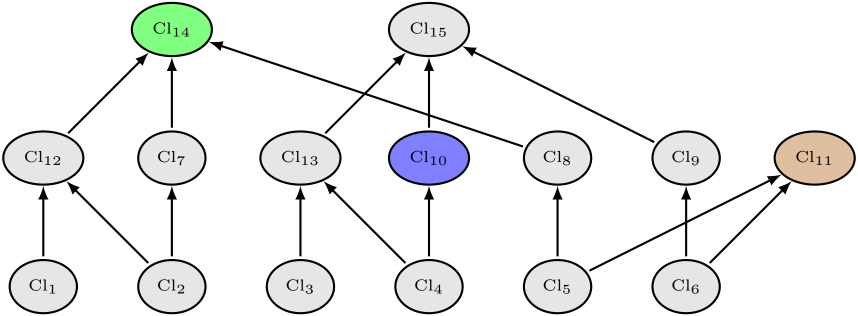
1. Define a set $\mathscr{B}$, and initialize $\mathscr{B}$ to the empty set.
2. Arbitrarily choose a wiretap set $A \in \mathscr{A}$ ($\mathscr{A}$ is not necessary regular) that has the largest cardinality in $\mathscr{A}$. Find the primary minimum cut between s and A, and call it CUT.
3. Partition the edge set E into two disjoint subsets: E_{CUT} and $E^c_{\mathrm{CUT}} \triangleq E \setminus E_{\mathrm{CUT}}$, where E_{CUT} is the set of the edges reachable from the source node s upon deleting the edges in CUT.
4. Remove ~~**all the wiretap sets in $\mathscr{A}$**~~ **all the wiretap or edge sets in $\mathscr{A} \cup \mathscr{B}$** that are subsets of E^c_{CUT}. Add the primary minimum cut CUT to $\mathscr{B}$.
5. Repeat Steps 2) to 4) until $\mathscr{A}$ is empty and output $\mathscr{B}$, where $N_{\max}(\mathscr{A}) = |\mathscr{B}|$.

Verification of Modified Algorithm II

In Step 4), CUT_A is added to $\mathscr{B}$.

- If A has the largest minimum cut capacity in $\mathscr{A}$ (e.g. Cl_{14}), then CUT_A will stay in $\mathscr{B}$ until the algorithm terminates.
- If A does not have the largest minimum cut capacity in $\mathscr{A}$,
  1. if A belongs to a maximal equivalence class (e.g. Cl_{11}), then CUT_A will stay in $\mathscr{B}$ until the algorithm terminates;
  2. otherwise (e.g. Cl_{10}), CUT_A will eventually be replaced by a primary minimum cut of a maximal equivalence class Cl with $\mathrm{Cl} \succ \mathrm{Cl}(A)$.
- Algorithm II computes the minimum cut capacity **at most** $N(\mathscr{A})$ times (instead of exactly $|\mathscr{A}|$ times).

Example (Cont.)



Algorithm II (Without Regular Assumption)

Algorithm 1: Algorithm for Computing $N_{\max}(\mathscr{A})$

Input: The wiretap network $(G, s, T, \mathscr{A})$, where $G = (V, E)$.

Output: $N_{\max}(\mathscr{A})$, the number of maximal equivalence classes with respect to $(G, s, T, \mathscr{A})$.

begin
  Set $\mathscr{B} = \emptyset$;
  while $\mathscr{A} \neq \emptyset$ **do**
    choose a wiretap set A of the largest cardinality in $\mathscr{A}$;
    find the primary minimum cut CUT of A;
    partition E into two parts E_{CUT} and $E^c_{\mathrm{CUT}} = E \setminus E_{\mathrm{CUT}}$;
    for each $B \in \mathscr{A} \cup \mathscr{B}$ **do**
      if $B \subseteq E^c_{\mathrm{CUT}}$ **then**
        remove B from $\mathscr{A}$.
      end
    end
    add CUT to $\mathscr{B}$.
  end
  Return $\mathscr{B}$. // Note that $|\mathscr{B}| = N_{\max}(\mathscr{A})$.
end

Line 5: Edge Partition

- **Line 5 in Algorithm II** can be implemented efficiently by slightly modifying existing search algorithms on directed graphs.
- The complexity is in $\mathcal{O}(|E_{\mathrm{CUT}}|)$ time.

Algorithm for Edge Partition

Algorithm 2: Search Algorithm

```
begin
    Unmark all nodes in V;
    mark source node s;
    pred(s) := 0;    // pred(i) refers to a predecessor node of node i.
    set the edge-set SET = ∅;
    set the node-set LIST = {s};
    while LIST ≠ ∅ do
        select a node i in LIST;
        if node i is incident to an edge (i, j) such that node j is unmarked
        then
            mark node j;
            pred(j) := i;
            add node j to LIST;
            add all parallel edges leading from i to j to SET;
        else
            delete node i from LIST;
        end
    end
    Return the edge-set SET.
end
```

Line 4: Finding Primary Minimum Cut

- Instead of the primary minimum cut between s and an edge set A, we consider the primary minimum cut between s and a sink node t.

Line 4: Finding Primary Minimum Cut

- Instead of the primary minimum cut between s and an edge set A, we consider the primary minimum cut between s and a sink node t.
- Let f be a maximal flow from s to t. Then f can be decomposed into $n(=\mathrm{mincut}(s,t))$ edge-disjoint paths $P_1, P_2, \cdots, P_n$ from s to t such that for every edge e,

$$f(e) = \begin{cases} 1, & e \in P_i \text{ for some } 1 \le i \le n; \\ 0, & \text{otherwise.} \end{cases}$$

Algorithm for Finding Primary Minimum Cut

Algorithm 3: Algorithm for Finding the Primary Minimum Cut

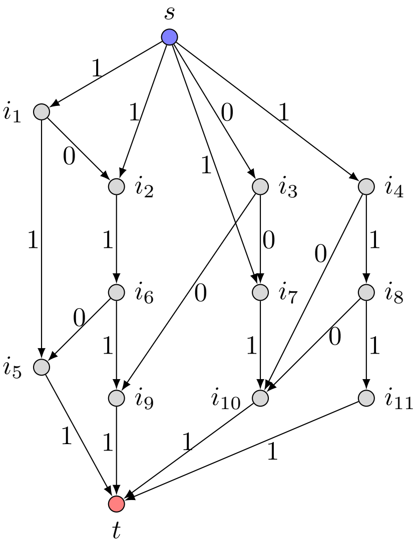
Input: An acyclic network $G = (V, E)$ with a maximal flow f from the source node s to a sink node t.

Output: The primary minimum cut between s and t.

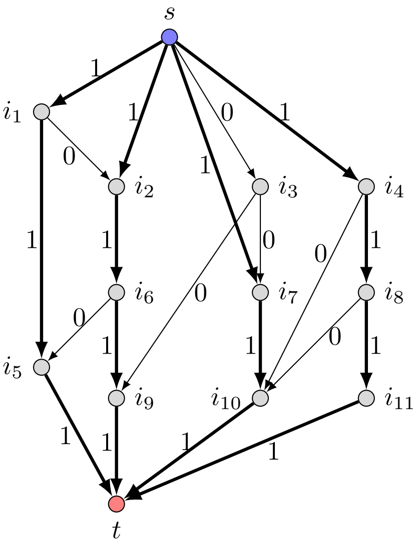
begin
- Set $S = \{s\}$;
- **for** each node $i \in S$ **do**
  - **if** **$\exists$ a node $j \in V \setminus S$ s.t. either $\exists$ a forward edge e from i to j s.t. $f(e) = 0$ or $\exists$ a reverse edge e from j to i s.t. $f(e) = 1$** **then**
    - replace S by $S \cup \{j\}$.
  - **end**
- **end**
- Return $\mathrm{CUT} = \{e : \ \mathrm{tail}(e) \in S$ and $\mathrm{head}(e) \in V \setminus S\}$.

end

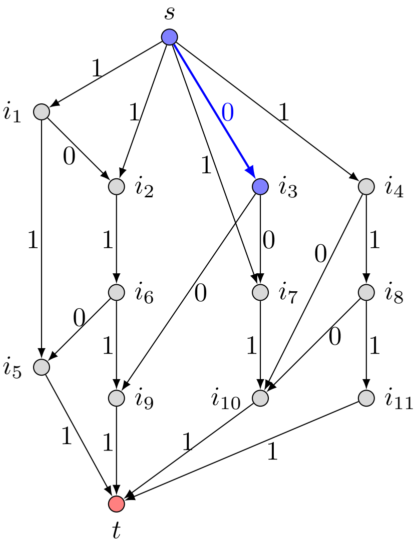
Example for Algorithm 3



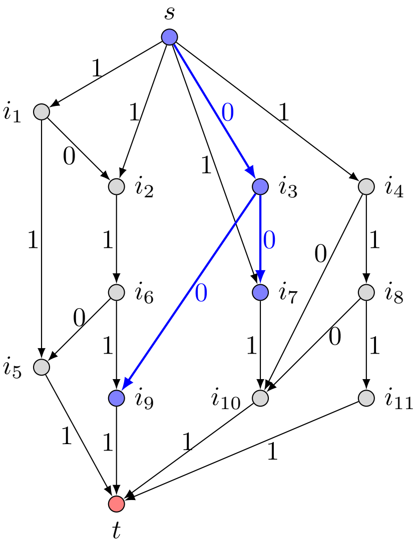
Example for Algorithm 3



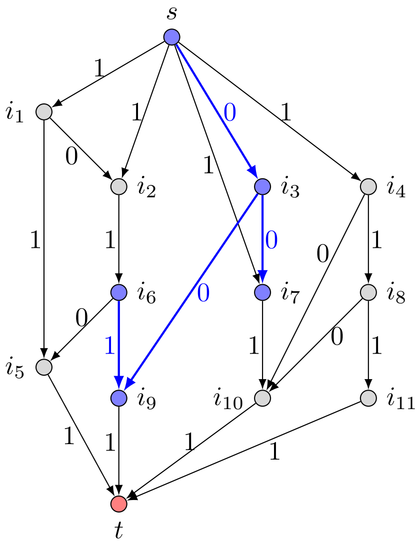
Example for Algorithm 3



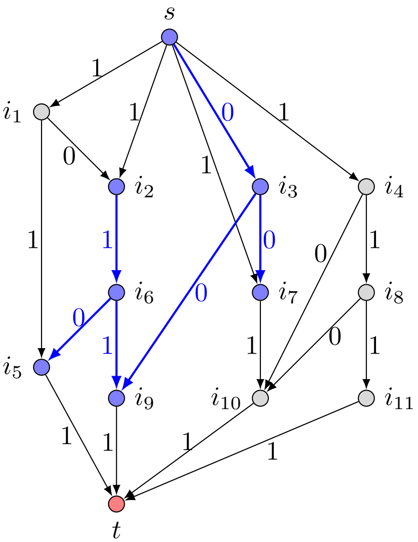
Example for Algorithm 3



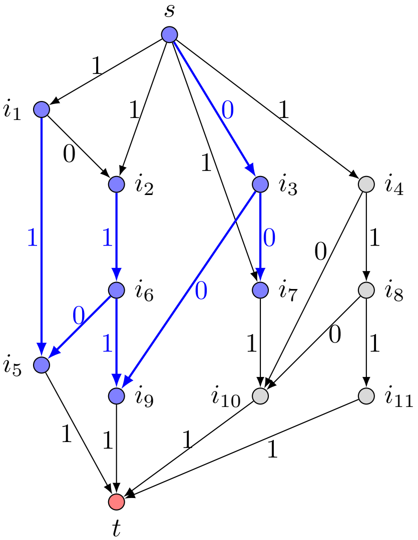
Example for Algorithm 3



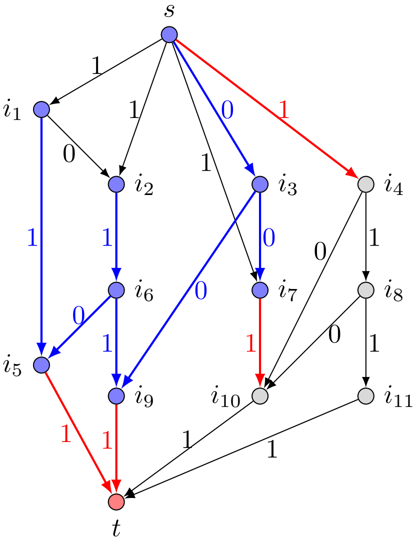
Example for Algorithm 3



Example for Algorithm 3



Example for Algorithm 3



Algorithm for Finding Primary Minimum Cut

Theorem 10

The output edge set CUT *of Algorithm 3 is the primary minimum cut between* s *and* t*.*

- The complexity of Algorithm 3 does not exceed $\mathcal{O}(|E|)$ time.

Outline

1. Preliminaries
   - Secure Network Coding
   - Alphabet Size Problem
2. A New Lower Bound on Required Alphabet Size
3. Efficient Algorithm for Computing the Lower Bound
   - Primary Minimum Cut
   - Algorithm
4. Conclusion Remarks

Concluding Remarks

- Our lower bound is independent of constructions of secure network codes.
- Our lower bound is applicable to both linear and non-linear secure network codes.

Concluding Remarks

- Our lower bound is independent of constructions of secure network codes.
- Our lower bound is applicable to both linear and non-linear secure network codes.
- Many proofs are non-trivial, involving some new techniques.

Concluding Remarks

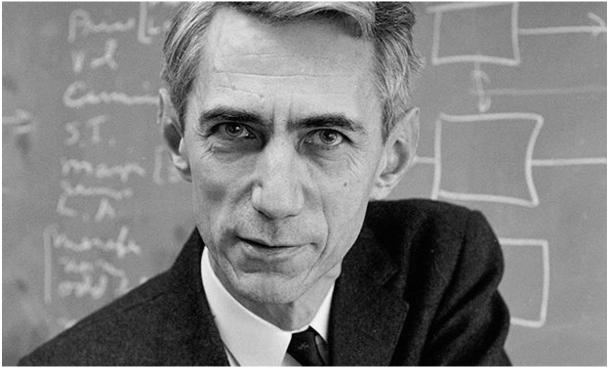
- Our lower bound is independent of constructions of secure network codes.
- Our lower bound is applicable to both linear and non-linear secure network codes.
- Many proofs are non-trivial, involving some new techniques.
- Whether the graph theoretic approach can help solve other alphabet size problems, such as in network error correction coding.

Concluding Remarks

- Our lower bound is independent of constructions of secure network codes.
- Our lower bound is applicable to both linear and non-linear secure network codes.
- Many proofs are non-trivial, involving some new techniques.
- Whether the graph theoretic approach can help solve other alphabet size problems, such as in network error correction coding.
- The concepts and results are of fundamental interest in graph theory and we expect that they will find applications in graph theory and beyond.

Happy Shannon's Centenary!!!

Happy Shannon's Centenary!!!



Thanks for your attention!