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

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
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
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 $G = (V, E)$ 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 $\mathcal{A}$ be a collection of subsets of $E$, where every edge set in $\mathcal{A}$ is called a wiretap set.
A **wiretap network** is a quadruple \((G, s, T, \mathcal{A})\), 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 \(\mathcal{A}\), but no more than one, may be fully accessed by a wiretapper;
- \(\mathcal{A}\) is known by \(s\) and all \(t \in T\) but which wiretap set in \(\mathcal{A}\) 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.
Let $\mathcal{F}$ be an alphabet.

An $\mathcal{F}$-valued secure network code on a wiretap network $(G, s, T, \mathcal{A})$ consists of a set of local encoding mappings $\{\phi_e : e \in E\}$ such that

- if $e \in \text{Out}(s)$,

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

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

  $$\phi_e : \mathcal{F}^{|\text{In}(v)|} \to \mathcal{F}.$$
Definition 1

For a secure network code on the wiretap network \((G, s, T, \mathcal{A})\), \(I(Y_A; M) = 0\) for every wiretap set \(A \in \mathcal{A}\), where \(I(Y_A; M)\) denotes the mutual information between \(Y_A = (Y_e : e \in A)\) and \(M\).
Proposition 1 ([Cai & Yeung]¹)

Let \((G, s, T, \mathcal{A})\) 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 \((G, s, T, \mathcal{A})\) provided that \(|\mathcal{F}| > |\mathcal{A}|\).

The lower bound $|\mathcal{A}|$ 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 $|A| = \text{mincut}(s, A)$.
- The collection of wiretap sets $\mathcal{A}$ is said to be regular, if all wiretap sets in $\mathcal{A}$ are regular.
An Assumption

Assume that all wiretap sets are regular.

- A wiretap set $A$ is said to be regular, if $|A| = \text{mincut}(s, A)$.
- The collection of wiretap sets $\mathcal{A}$ is said to be regular, if all wiretap sets in $\mathcal{A}$ are regular.
- Replace non-regular wiretap sets in $\mathcal{A}$ by their minimum cuts (that are regular) to form $\mathcal{A}'$.
- A secure network code that is secure for $\mathcal{A}'$ is also secure for $\mathcal{A}$. 
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
Let \((G, s, T, \mathcal{A})\) 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 \(\text{CUT}\) that is a minimum cut between \(s\) and \(A\)
- and also between \(s\) and \(A'\).
Proposition 2 ([Guang et al.]²)

The binary relation “∼” 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”,

Proposition 3

Let $A_1, A_2, \ldots, A_m$ be $m$ equivalent edge sets under the equivalence relation “∼”. Then

$$\mincut(s, \bigcup_{i=1}^{m} A_i) = \mincut(s, A_j), \quad \forall j, \ 1 \leq j \leq m.$$
Equivalence Relation “∼”

**Proposition 3**

Let $A_1, A_2, \cdots, A_m$ be $m$ equivalent edge sets under the equivalence relation “∼”. Then

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

- With “∼”, the wiretap sets in $\mathcal{A}$ can be partitioned into equivalence classes.
- All the wiretap sets in an equivalence class have a common minimum cut.
Denote $N(\mathcal{A})$ by the number of the equivalence classes in $\mathcal{A}$.

**Theorem 4**

Let $(G, s, T, A)$ 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 $(G, s, T, A)$ provided that

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

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

---

Example
Let the collection of wiretap sets $\mathcal{A}$ be:

$$
\mathcal{A} = \left\{ \{e_6\}, \{e_7\}, \{e_8\}, \{e_9\}, \{e_{12}\}, \{e_{13}\}, \{e_{14}\}, \{e_{15}\}, \{e_{18}\}, \{e_{19}\}, \{e_{20}\}, \{e_{21}\}, \{e_6, e_{18}\}, \{e_6, e_{19}\}, \{e_7, e_{18}\}, \{e_7, e_{19}\}, \{e_8, e_{11}\}, \{e_8, e_{16}\}, \{e_8, e_{18}\}, \{e_9, e_{10}\}, \{e_9, e_{18}\}, \{e_9, e_{19}\}, \{e_{10}, e_{14}\}, \{e_{10}, e_{15}\}, \{e_{10}, e_{19}\}, \{e_{10}, e_{21}\}, \{e_{11}, e_{14}\}, \{e_{11}, e_{15}\}, \{e_{11}, e_{18}\}, \{e_{11}, e_{20}\}, \{e_{12}, e_{20}\}, \{e_{12}, e_{21}\}, \{e_{13}, e_{17}\}, \{e_{13}, e_{21}\}, \{e_{14}, e_{20}\}, \{e_{14}, e_{21}\}, \{e_{15}, e_{20}\}, \{e_{15}, e_{21}\}, \{e_{18}, e_{20}\}, \{e_{18}, e_{21}\}, \{e_{19}, e_{20}\}, \{e_{19}, e_{21}\}, \{e_1, e_3, e_{16}\}, \{e_1, e_{11}, e_{16}\}, \{e_2, e_{10}, e_{16}\}, \{e_3, e_5, e_{17}\}, \{e_4, e_{10}, e_{17}\}, \{e_5, e_{11}, e_{17}\} \right\}.
$$

$|\mathcal{A}| = 48.$
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:

\[
\begin{align*}
\text{Cl}_1 &= \left\{ \{e_6\}, \{e_7\} \right\}, \\
\text{Cl}_2 &= \left\{ \{e_8\}, \{e_9\} \right\}, \\
\text{Cl}_3 &= \left\{ \{e_{12}\}, \{e_{13}\} \right\}, \\
\text{Cl}_4 &= \left\{ \{e_{14}\}, \{e_{15}\} \right\}, \\
\text{Cl}_5 &= \left\{ \{e_{18}\}, \{e_{19}\} \right\}, \\
\text{Cl}_6 &= \left\{ \{e_{20}\}, \{e_{21}\} \right\}, \\
\text{Cl}_7 &= \left\{ \{e_{8}, e_{11}\}, \{e_{9}, e_{10}\} \right\}, \\
\text{Cl}_8 &= \left\{ \{e_{10}, e_{19}\}, \{e_{11}, e_{18}\} \right\}, \\
\text{Cl}_9 &= \left\{ \{e_{10}, e_{21}\}, \{e_{11}, e_{20}\} \right\}, \\
\text{Cl}_{10} &= \left\{ \{e_{10}, e_{14}\}, \{e_{10}, e_{15}\}, \{e_{11}, e_{14}\}, \{e_{11}, e_{15}\} \right\}, \\
\text{Cl}_{11} &= \left\{ \{e_{18}, e_{20}\}, \{e_{18}, e_{21}\}, \{e_{19}, e_{20}\}, \{e_{19}, e_{21}\} \right\};
\end{align*}
\]
Example

\[
\begin{align*}
\text{Cl}_{12} &= \left\{ \{e_6, e_{18}\}, \{e_6, e_{19}\}, \{e_7, e_{18}\}, \{e_7, e_{19}\}, \\
&\{e_8, e_{16}\}, \{e_8, e_{18}\}, \{e_9, e_{18}\}, \{e_9, e_{19}\} \right\}, \\
\text{Cl}_{13} &= \left\{ \{e_{12}, e_{20}\}, \{e_{12}, e_{21}\}, \{e_{13}, e_{17}\}, \{e_{13}, e_{21}\}, \\
&\{e_{14}, e_{20}\}, \{e_{14}, e_{21}\}, \{e_{15}, e_{20}\}, \{e_{15}, e_{21}\} \right\}, \\
\text{Cl}_{14} &= \left\{ \{e_1, e_3, e_{16}\}, \{e_1, e_{11}, e_{16}\}, \{e_2, e_{10}, e_{16}\} \right\}, \\
\text{Cl}_{15} &= \left\{ \{e_3, e_5, e_{17}\}, \{e_4, e_{10}, e_{17}\}, \{e_5, e_{11}, e_{17}\} \right\}.
\end{align*}
\]

Then \( N(\mathcal{A}) = 15 \) \((< |\mathcal{A}| = 48)\).
Furthermore, consider $C_{15} = \{ \{e_{18}\}, \{e_{19}\}\}$ and $\{e_1, e_3, e_{16}\}$. 
Definition 2 (Wiretap-Set Domination)

Let $A_1$ and $A_2$ be two wiretap sets in $A$ 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 $A_1 \prec A_2$. 
Definition 3 (Equivalence-Class Domination)

For two distinct equivalence classes $\text{Cl}_1$ and $\text{Cl}_2$, if there exists a common minimum cut of the wiretap sets in $\text{Cl}_2$ that separates all the wiretap sets in $\text{Cl}_1$ from $s$, we say that $\text{Cl}_1$ is dominated by $\text{Cl}_2$, denoted by $\text{Cl}_1 \prec \text{Cl}_2$. 
Theorem 5

\[ \text{Cl}(A_1) \prec \text{Cl}(A_2) \text{ if and only if } A_1 \prec A_2. \]
The equivalence-class domination relation “≺” amongst the equivalence classes in 𝒜 is a strict partial order. Specifically, let \( \text{Cl}_1, \text{Cl}_2, \text{and} \text{Cl}_3 \) be three arbitrary equivalence classes, and then

1. **(Irreflexivity)** \( \text{Cl}_1 \not≺ \text{Cl}_1; \)

2. **(Transitivity)** if \( \text{Cl}_1 \prec \text{Cl}_2 \) and \( \text{Cl}_2 \prec \text{Cl}_3 \), then \( \text{Cl}_1 \prec \text{Cl}_3; \)

3. **(Asymmetry)** if \( \text{Cl}_1 \prec \text{Cl}_2 \), then \( \text{Cl}_2 \not≺ \text{Cl}_1. \)
Maximal Equivalence Class

- Now, the set of all the equivalence classes in $A$ 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 $A$, an equivalence class $Cl$ is a maximal equivalence class if there exists no other equivalence class $Cl'$ such that $Cl' \succ Cl$. Denote by $N_{\text{max}}(A)$ the number of the maximal equivalence classes with respect to $A$. 
Theorem 7

Let \((G, s, T, \mathcal{A})\) 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 \((G, s, T, \mathcal{A})\) provided that the alphabet size

\[|\mathcal{F}| > N_{\text{max}}(\mathcal{A}).\]

\[N_{\text{max}}(\mathcal{A}) \leq N(\mathcal{A}) \leq |\mathcal{A}|.\]
The Hasse diagram of all 15 equivalence classes, ordered by the equivalence-class domination relation “≺”.

Figure: $\text{Cl}_{11}$, $\text{Cl}_{14}$, and $\text{Cl}_{15}$ are all of the maximal equivalence classes.
The alphabet size $|\mathcal{F}|$

| Lower Bound I: $|\mathcal{A}|$ | 48 |
| Lower Bound II: $N(\mathcal{A})$ | 15 |
| Lower Bound III: $N_{\text{max}}(\mathcal{A})$ | 3 |

- The improvement of $N_{\text{max}}(\mathcal{A})$ over $N(\mathcal{A})$ can be unbounded.
$N_{\text{max}}(\mathcal{A})$ is graph-theoretical.

$N_{\text{max}}(\mathcal{A})$ only depends on the topology of the network $G$ and the collection $\mathcal{A}$ of wiretap sets.
New Problem Proposed

- $N_{\text{max}}(\mathcal{A})$ is graph-theoretical.

- $N_{\text{max}}(\mathcal{A})$ only depends on the topology of the network $G$ and the collection $\mathcal{A}$ of wiretap sets.

- In general, computing the value of $N_{\text{max}}(\mathcal{A})$, or characterizing the corresponding Hasse diagram, is nontrivial.

- Even in the simple example, its value is not obvious.
New Problem Proposed

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

**How to efficiently compute $N_{\text{max}}(\mathcal{A})$?**
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
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.
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 $A \in \mathcal{A}$.
Theorem 9

In a wiretap network \((G, s, T, A)\), let \(C_l\) be an arbitrary equivalence class of the wiretap sets. Then

1. all the wiretap sets in \(C_l\) have the same primary minimum cut, which hence is called the primary minimum cut of the equivalence class \(C_l\), and

2. for every equivalence class \(C_l'\) with \(C_l' \prec C_l\), the primary minimum cut of \(C_l\) separates all the wiretap sets in \(C_l'\) from \(s\).
To compute $N_{\text{max}}(\mathcal{A})$, 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 $N_{\text{max}}(\mathcal{A})$:

1. Define a set $\mathcal{B}$, and initialize $\mathcal{B}$ to the empty set.

2. Arbitrarily choose a wiretap set $A \in \mathcal{A}$ that has the largest cardinality in $\mathcal{A}$. Find the primary minimum cut between $s$ and $A$, and call it $\text{CUT}$.

3. Partition the edge set $E$ into two disjoint subsets: $E_{\text{CUT}}$ and $E_{\text{CUT}}^c \triangleq E \setminus E_{\text{CUT}}$, where $E_{\text{CUT}}$ is the set of the edges reachable from the source node $s$ upon deleting the edges in $\text{CUT}$.

4. Remove all the wiretap sets in $\mathcal{A}$ that are subsets of $E_{\text{CUT}}^c$ and add the primary minimum cut $\text{CUT}$ to $\mathcal{B}$.

5. Repeat Steps 2) to 4) until $\mathcal{A}$ is empty and output $\mathcal{B}$, where $N_{\text{max}}(\mathcal{A}) = |\mathcal{B}|$. 
Algorithm for computing $N(\mathcal{A})$:

1. Define a set $\mathcal{B}$, and initialize $\mathcal{B}$ to the empty set.

2. Arbitrarily choose a wiretap set $A \in \mathcal{A}$ that has the largest cardinality in $\mathcal{A}$. Find the primary minimum cut between $s$ and $A$, and call it CUT.

3. Partition the edge set $E$ into two disjoint subsets: $E_{\text{CUT}}$ and $E_{\text{CUT}}^c \triangleq E \setminus E_{\text{CUT}}$, where $E_{\text{CUT}}$ 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 $\mathcal{A}$ that are subsets of $E_{\text{CUT}}^c$. Add the primary minimum cut CUT to $\mathcal{B}$.

5. Repeat Steps 2) to 4) until $\mathcal{A}$ is empty and output $\mathcal{B}$, where $N(\mathcal{A}) = |\mathcal{B}|$. 
Example (Cont.)
Example (Cont.)
Algorithm modified for computing $N_{\text{max}}(\mathcal{A})$ without regular assumption:

1. Define a set $\mathcal{B}$, and initialize $\mathcal{B}$ to the empty set.

2. Arbitrarily choose a wiretap set $A \in \mathcal{A}$ (\(\mathcal{A}\) is not necessarily regular) that has the largest cardinality the largest minimum cut capacity in $\mathcal{A}$. Find the primary minimum cut between $s$ and $A$, and call it $\text{CUT}$.

3. Partition the edge set $E$ into two disjoint subsets: $E_{\text{CUT}}$ and $E_{\text{CUT}}^c \triangleq E \setminus E_{\text{CUT}}$, where $E_{\text{CUT}}$ is the set of the edges reachable from the source node $s$ upon deleting the edges in $\text{CUT}$.

4. Remove all the wiretap sets in $\mathcal{A}$ that are subsets of $E_{\text{CUT}}^c$ and add the primary minimum cut $\text{CUT}$ to $\mathcal{B}$.

5. Repeat Steps 2) to 4) until $\mathcal{A}$ is empty and output $\mathcal{B}$, where $N_{\text{max}}(\mathcal{A}) = |\mathcal{B}|$. 
However,

- This modification requires pre-computing the minimum cut capacity of every wiretap set in $\mathcal{A}$.

- This will significantly increase the computational complexity of the algorithm when $|\mathcal{A}|$ is large.
Algorithm (Without Regular Assumption)

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

1. Define a set $\mathcal{B}$, and initialize $\mathcal{B}$ to the empty set.

2. Arbitrarily choose a wiretap set $A \in \mathcal{A}$ ($\mathcal{A}$ is not necessary regular) that has the largest cardinality in $\mathcal{A}$. Find the primary minimum cut between $s$ and $A$, and call it CUT.

3. Partition the edge set $E$ into two disjoint subsets: $E_{\text{CUT}}$ and $E_{\text{CUT}}^c \triangleq E \setminus E_{\text{CUT}}$, where $E_{\text{CUT}}$ 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 $\mathcal{A}$ and all the wiretap or edge sets in $\mathcal{A} \cup \mathcal{B}$ that are subsets of $E_{\text{CUT}}^c$. Add the primary minimum cut CUT to $\mathcal{B}$.

5. Repeat Steps 2) to 4) until $\mathcal{A}$ is empty and output $\mathcal{B}$, where $N_{\text{max}}(\mathcal{A}) = |\mathcal{B}|$. 

In Step 4), \( \text{CUT}_A \) is added to \( \mathcal{B} \).

- If \( A \) has the largest minimum cut capacity in \( \mathcal{A} \) (e.g. \( \text{Cl}_{14} \)), then \( \text{CUT}_A \) will stay in \( \mathcal{B} \) until the algorithm terminates.

- If \( A \) does not have the largest minimum cut capacity in \( \mathcal{A} \),
  1. if \( A \) belongs to a maximal equivalence class (e.g. \( \text{Cl}_{11} \)), then \( \text{CUT}_A \) will stay in \( \mathcal{B} \) until the algorithm terminates;
  2. otherwise (e.g. \( \text{Cl}_{10} \)), \( \text{CUT}_A \) will eventually be replaced by a primary minimum cut of a maximal equivalence class \( \text{Cl} \) with \( \text{Cl} \succ \text{Cl}(A) \).

- Algorithm II computes the minimum cut capacity at most \( N(\mathcal{A}) \) times (instead of exactly \( |\mathcal{A}| \) times).
Algorithm 1: Algorithm for Computing $N_{\text{max}}(\mathcal{A})$

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

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

begin
1. Set $\mathcal{B} = \emptyset$;
2. while $\mathcal{A} \neq \emptyset$ do
   3. choose a wiretap set $A$ of the largest cardinality in $\mathcal{A}$;
   4. find the primary minimum cut $\text{CUT}$ of $A$;
   5. partition $E$ into two parts $E_{\text{CUT}}$ and $E_{\text{CUT}}^c = E \setminus E_{\text{CUT}}$;
   6. for each $B \in \mathcal{A} \cup \mathcal{B}$ do
      7. if $B \subseteq E_{\text{CUT}}^c$ then
         8. remove $B$ from $\mathcal{A}$.
      end
   end
   9. add $\text{CUT}$ to $\mathcal{B}$.
end
10. Return $\mathcal{B}$.
\hspace{1cm} // Note that $|\mathcal{B}| = N_{\text{max}}(\mathcal{A})$. 
Line 5 in Algorithm II can be implemented efficiently by slightly modifying existing search algorithms on directed graphs. 

- The complexity is in $O(|E_{\text{CUT}}|)$ time.
Algorithm for Edge Partition

**Algorithm 2: Search Algorithm**

```plaintext
begin
1    Unmark all nodes in $V$;
2    mark source node $s$;
3    pred($s$) := 0;  // pred($i$) refers to a predecessor node of node $i$.
4    set the edge-set SET = $\emptyset$;
5    set the node-set LIST = \{s\};
6    while LIST $\neq$ $\emptyset$ do
7        select a node $i$ in LIST;
8        if node $i$ is incident to an edge $(i, j)$ such that node $j$ is unmarked
9            then
10               mark node $j$;
11               pred($j$) := $i$;
12               add node $j$ to LIST;
13               add all parallel edges leading from $i$ to $j$ to SET;
14            else
15               delete node $i$ from LIST;
16        end
17    end
18    Return the edge-set SET.
end
```
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 $n = \min \text{cut}(s, t)$ edge-disjoint paths $P_1, P_2, \ldots, P_n$ from $s$ to $t$ such that for every edge $e$, $f(e) = \{1, e \in P_i \text{ for some } 1 \leq i \leq n; 0, \text{ otherwise.}\}$
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 $n(=\text{mincut}(s,t))$ edge-disjoint paths $P_1, P_2, \cdots, P_n$ from $s$ to $t$ such that for every edge $e$,

$$f(e) = \begin{cases} 
1, & e \in P_i \text{ for some } 1 \leq i \leq n; \\
0, & \text{otherwise}. 
\end{cases}$$
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
1. Set \( S = \{s\} \);
2. for each node \( i \in S \) do
   3. 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
      4. replace \( S \) by \( S \cup \{j\} \).
   end
end
5. Return \( \text{CUT} = \{e : \text{tail}(e) \in S \text{ and } \text{head}(e) \in V \setminus S\} \).
end
Example for Algorithm 3
Example for Algorithm 3
Example for Algorithm 3
Example for Algorithm 3

\[s \quad i_1 \quad 1 \quad i_2 \quad 1 \quad i_3 \quad 1 \quad i_4 \]
\[i_5 \quad 1 \quad i_6 \quad 0 \quad i_7 \quad 0 \]
\[i_8 \quad 1 \quad i_9 \quad 1 \quad i_{10} \quad 1 \quad i_{11} \]
\[t \]

Xuan Guang (CUHK)  Alphabet Size Reduction for SNC  Dec. 7, 2016  48 / 50
Example for Algorithm 3
Example for Algorithm 3
Example for Algorithm 3
Theorem 10

The output edge set \( \text{CUT} \) 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
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!