RUB

RUHR-UNIVERSITÄT BOCHUM

The Subset-Sum Problem

cryptanalysis employing a probabilistic approach

@ KU Leuven, 22.09.2016 – 13.30

Matthias Minihold
ECRYPT-NET Early Stage Researcher
Cryptology and IT-Security, Ruhr-Universität Bochum

hgi
Horst Görtz Institut
für IT-Sicherheit

1 The Subset-Sum problem

- Motivation
- Historical Remarks
- Easy and Hard Instances
- Evolution of Algorithms
- Technique 1 - Meet in the Middle
- Technique 2 - Enlarge Number Set
- Applications

Outline

1. The Subset-Sum problem
   - Motivation
   - Historical Remarks
   - Easy and Hard Instances
   - Evolution of Algorithms
   - Technique 1 - Meet in the Middle
   - Technique 2 - Enlarge Number Set
   - Applications

hgi
Horst Görtz Institut
für IT-Sicherheit

Motivation — informal

Motivation — informal

Informally: Pack a knapsack with items to meet a weight constraint.

Motivation — informal

Informally: Pack a knapsack with items to meet a weight constraint.
Suppose you are at the airport:

- The luggage may weight $\leq S$ [kg] at check-in.

Motivation — informal

Informally: Pack a knapsack with items to meet a weight constraint.
Suppose you are at the airport:

- The luggage may weight $\leq S$ [kg] at check-in.
- Not squandering, the bag WILL weight S [kg].

Motivation — informal

Informally: Pack a knapsack with items to meet a weight constraint.
Suppose you are at the airport:

- The luggage may weight $\leq S$ [kg] at check-in.
- Not squandering, the bag WILL weight S [kg].
- n equally beautiful items with weights $a_1, a_2, \ldots a_n$.

Motivation — informal

Informally: Pack a knapsack with items to meet a weight constraint.
Suppose you are at the airport:

- The luggage may weight $\leq S$ [kg] at check-in.
- Not squandering, the bag WILL weight S [kg].
- n equally beautiful items with weights $a_1, a_2, \ldots a_n$.

Motivation — informal

Informally: Pack a knapsack with items to meet a weight constraint.
Suppose you are at the airport:

- The luggage may weight $\leq S$ [kg] at check-in.
- Not squandering, the bag WILL weight S [kg].
- n equally beautiful items with weights $a_1, a_2, \ldots a_n$.

Definition 1 (Subset-Sum)

$$\text{Given } n, S, a_1, a_2, \ldots a_n \in \mathbb{N}, \text{ find } I \subseteq [n] : \sum_{i \in I} a_i = S. \tag{1}$$

Outline

Historical Remarks

- Names: (0-1-)Knapsack, Subset-Sum problem

Historical Remarks

- Names: (0-1-)Knapsack, Subset-Sum problem
- First studied in 1897

Historical Remarks

- Names: (0-1-)Knapsack, Subset-Sum problem
- First studied in 1897
- Early recognized to be $\mathcal{NP}$-$\texttt{complete}$ by reduction to 3-SAT:

Historical Remarks

- Names: (0-1-)Knapsack, Subset-Sum problem
- First studied in 1897
- Early recognized to be $\mathcal{NP}$-`complete` by reduction to 3-SAT:
  - Satisfiability with at most 3 literals per clause ($\hat{=}$ 3-SAT)

Historical Remarks

- Names: (0-1-)Knapsack, Subset-Sum problem
- First studied in 1897
- Early recognized to be $\mathcal{NP}$-complete by reduction to 3-SAT:
  - Satisfiability with at most 3 literals per clause ($\triangleq$ 3-SAT)
  - $\Rightarrow_R$ Chromatic number ($=$ Graph Coloring)

Historical Remarks

- Names: (0-1-)Knapsack, Subset-Sum problem
- First studied in 1897
- Early recognized to be $\mathcal{NP}$-`complete` by reduction to 3-SAT:
  - Satisfiability with at most 3 literals per clause ($\triangleq$ 3-SAT)
  - $\Rightarrow_R$ Chromatic number ($=$ Graph Coloring)
  - $\Rightarrow_R$ Exact cover

Historical Remarks

- Names: (0-1-)Knapsack, Subset-Sum problem
- First studied in 1897
- Early recognized to be $\mathcal{NP}$-`complete` by reduction to 3-SAT:
    - Satisfiability with at most 3 literals per clause ($\hat{=}$ 3-SAT)
    - $\Rightarrow_R$ Chromatic number ($=$ Graph Coloring)
    - $\Rightarrow_R$ Exact cover
    - $\Rightarrow_R$ Knapsack

Historical Remarks

- Names: (0-1-)Knapsack, Subset-Sum problem
- First studied in 1897
- Early recognized to be $\mathcal{NP}$-`complete` by reduction to 3-SAT:
  - Satisfiability with at most 3 literals per clause ($\hat{=}$ 3-SAT)
  - $\Rightarrow_R$ Chromatic number ($=$ Graph Coloring)
  - $\Rightarrow_R$ Exact cover
  - $\Rightarrow_R$ Knapsack
  - worst-case instances are computationally intractable

RUB

Outline

hgi
Horst Görtz Institut
für IT-Sicherheit

Easy and Hard Instances

Given an instance by $n, S, a_1, a_2, \ldots a_n \in \mathbb{N}$ with density $d := \frac{n}{\log(\max_i a_i)}$.

Easy and Hard Instances

Given an instance by $n, S, a_1, a_2, \ldots a_n \in \mathbb{N}$ with density $d := \frac{n}{\log(\max_i a_i)}$.

Easy instances:

- (d<0.9408) Solve Shortest Vector Problem in $\mathcal{L}(B)$.

Easy and Hard Instances

Given an instance by $n, S, a_1, a_2, \ldots a_n \in \mathbb{N}$ with density $d := \frac{n}{\log(\max_i a_i)}$.
Easy instances:

- (d<0.9408) Solve Shortest Vector Problem in $\mathcal{L}(B)$.
- (d>1) Solve Dynamic Programming Problem

Easy and Hard Instances

Given an instance by $n, S, a_1, a_2, \ldots a_n \in \mathbb{N}$ with density $d := \frac{n}{\log(\max_i a_i)}$.

Easy instances:

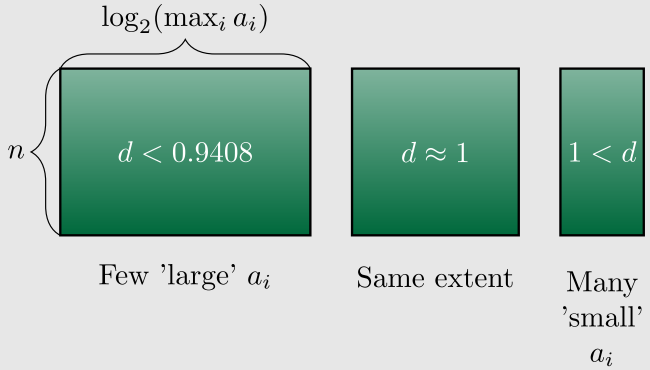
- (d<0.9408) Solve Shortest Vector Problem in $\mathcal{L}(B)$.
- (d>1) Solve Dynamic Programming Problem

Easy and Hard Instances

Given an instance by $n, S, a_1, a_2, \ldots a_n \in \mathbb{N}$ with density $d := \frac{n}{\log(\max_i a_i)}$.

Easy instances:

- (d<0.9408) Solve Shortest Vector Problem in $\mathcal{L}(B)$.
- (d>1) Solve Dynamic Programming Problem



Easy vs. Hard instances

Hard instances:

- Element size: $\log_2 a_i \approx n$ for $i = 1, 2, \ldots, n$

Easy vs. Hard instances

Hard instances:

- Element size: $\log_2 a_i \approx n$ for $i = 1, 2, \ldots, n$
- Characterized by $0.9408 < d = \frac{n}{\log(\max_i a_i)} < 1$

Easy vs. Hard instances

Hard instances:

- Element size: $\log_2 a_i \approx n$ for $i = 1, 2, \ldots, n$
- Characterized by $0.9408 < d = \frac{n}{\log(\max_i a_i)} < 1$
- Balanced solution: $|I| \approx \frac{n}{2}$

Easy vs. Hard instances

Hard instances:

- Element size: $\log_2 a_i \approx n$ for $i = 1, 2, \ldots, n$
- Characterized by $0.9408 < d = \frac{n}{\log(\max_i a_i)} < 1$
- Balanced solution: $|I| \approx \frac{n}{2}$
- How to generate hard instances parametrized by n:

Easy vs. Hard instances

Hard instances:

- Element size: $\log_2 a_i \approx n$ for $i = 1, 2, \ldots, n$
- Characterized by $0.9408 < d = \frac{n}{\log(\max_i a_i)} < 1$
- Balanced solution: $|I| \approx \frac{n}{2}$
- How to generate hard instances parametrized by n:
    1. Choose $a \in_R \{0, 1, \ldots, 2^n - 1\}^n, x \in_R \{0, 1\}^n$ uniformly.

Easy vs. Hard instances

Hard instances:

- Element size: $\log_2 a_i \approx n$ for $i = 1, 2, \ldots, n$
- Characterized by $0.9408 < d = \frac{n}{\log(\max_i a_i)} < 1$
- Balanced solution: $|I| \approx \frac{n}{2}$
- How to generate hard instances parametrized by n:
  1. Choose $a \in_R \{0, 1, \ldots, 2^n - 1\}^n, x \in_R \{0, 1\}^n$ uniformly.
  2. Output $(a = (a_1, a_2, \ldots a_n), S = a \cdot x = \sum_{i=1}^n x_i a_i)$.

Easy vs. Hard instances

Hard instances:

- Element size: $\log_2 a_i \approx n$ for $i = 1, 2, \ldots, n$
- Characterized by $0.9408 < d = \frac{n}{\log(\max_i a_i)} < 1$
- Balanced solution: $|I| \approx \frac{n}{2}$
- How to generate hard instances parametrized by n:
  1. Choose $a \in_R \{0, 1, \ldots, 2^n - 1\}^n, x \in_R \{0, 1\}^n$ uniformly.
  2. Output $(a = (a_1, a_2, \ldots a_n), S = a \cdot x = \sum_{i=1}^n x_i a_i)$.
  3. Finding x given (a, S) is (assumed to be) computationally hard.

Outline

1. The Subset-Sum problem
   - Motivation
   - Historical Remarks
   - Easy and Hard Instances
   - Evolution of Algorithms
   - Technique 1 - Meet in the Middle
   - Technique 2 - Enlarge Number Set
   - Applications

hgi Horst Görtz Institut für IT-Sicherheit

Evolution of Algorithms (giving exact solutions)

Algorithm (year)	Time	Space

Table: Expected time and space requirements of algorithms solving Eq. (1).

Evolution of Algorithms (giving exact solutions)

Algorithm (year)	Time	Space
Exhaustive Search	$2^{1.000n} \approx (2.000)^n$	$2^{0.000n} \approx (1.000)^n$

Table: Expected time and space requirements of algorithms solving Eq. (1).

Evolution of Algorithms (giving exact solutions)

Algorithm (year)	Time	Space
Exhaustive Search	$2^{1.000n} \approx (2.000)^n$	$2^{0.000n} \approx (1.000)^n$
$>$ Idea: *'meet-in-the-middle'*		

Table: Expected time and space requirements of algorithms solving Eq. (1).

Evolution of Algorithms (giving exact solutions)

Algorithm (year)	Time	Space
Exhaustive Search	$2^{1.000n} \approx (2.000)^n$	$2^{0.000n} \approx (1.000)^n$
$>$ Idea: *'meet-in-the-middle'*		
Horowitz-Sahni (1974)	$2^{0.500n} \approx (1.414)^n$	$2^{0.500n} \approx (1.414)^n$

Table: Expected time and space requirements of algorithms solving Eq. (1).

Evolution of Algorithms (giving exact solutions)

Algorithm (year)	**Time**	**Space**
Exhaustive Search	$2^{1.000n} \approx (2.000)^n$	$2^{0.000n} \approx (1.000)^n$
$>$ Idea: *'meet-in-the-middle'*		
Horowitz-Sahni (1974)	$2^{0.500n} \approx (1.414)^n$	$2^{0.500n} \approx (1.414)^n$
$>$ Idea: *'4-list approach'*		

Table: Expected time and space requirements of algorithms solving Eq. (1).

Evolution of Algorithms (giving exact solutions)

Algorithm (year)	Time	Space
Exhaustive Search	$2^{1.000n} \approx (2.000)^n$	$2^{0.000n} \approx (1.000)^n$
$>$ Idea: *'meet-in-the-middle'*		
Horowitz-Sahni (1974)	$2^{0.500n} \approx (1.414)^n$	$2^{0.500n} \approx (1.414)^n$
$>$ Idea: *'4-list approach'*		
Schroeppel-Shamir (1979)	$2^{0.500n} \approx (1.414)^n$	$2^{0.250n} \approx (1.189)^n$

Table: Expected time and space requirements of algorithms solving Eq. (1).

Evolution of Algorithms (giving exact solutions)

Algorithm (year)	Time	Space
Exhaustive Search	$2^{1.000n} \approx (2.000)^n$	$2^{0.000n} \approx (1.000)^n$
$>$ Idea: *'meet-in-the-middle'*		
Horowitz-Sahni (1974)	$2^{0.500n} \approx (1.414)^n$	$2^{0.500n} \approx (1.414)^n$
$>$ Idea: *'4-list approach'*		
Schroeppel-Shamir (1979)	$2^{0.500n} \approx (1.414)^n$	$2^{0.250n} \approx (1.189)^n$
$>$ Idea: *'multiple representations'*		

Table: Expected time and space requirements of algorithms solving Eq. (1).

Evolution of Algorithms (giving exact solutions)

Algorithm (year)	Time	Space
Exhaustive Search	$2^{1.000n} \approx (2.000)^n$	$2^{0.000n} \approx (1.000)^n$
$>$ Idea: *'meet-in-the-middle'*		
Horowitz-Sahni (1974)	$2^{0.500n} \approx (1.414)^n$	$2^{0.500n} \approx (1.414)^n$
$>$ Idea: *'4-list approach'*		
Schroeppel-Shamir (1979)	$2^{0.500n} \approx (1.414)^n$	$2^{0.250n} \approx (1.189)^n$
$>$ Idea: *'multiple representations'*		
Howgrave-Graham-Joux (2010)	$2^{0.337n} \approx (1.263)^n$	$2^{0.311n} \approx (1.241)^n$
$>$ Idea: *'enlarged number-set'* $\{-1, 0, 1\}$		

Table: Expected time and space requirements of algorithms solving Eq. (1).

Evolution of Algorithms (giving exact solutions)

Algorithm (year)	**Time**	**Space**
Exhaustive Search	$2^{1.000n} \approx (2.000)^n$	$2^{0.000n} \approx (1.000)^n$
$>$ Idea: *'meet-in-the-middle'*		
Horowitz-Sahni (1974)	$2^{0.500n} \approx (1.414)^n$	$2^{0.500n} \approx (1.414)^n$
$>$ Idea: *'4-list approach'*		
Schroeppel-Shamir (1979)	$2^{0.500n} \approx (1.414)^n$	$2^{0.250n} \approx (1.189)^n$
$>$ Idea: *'multiple representations'*		
Howgrave-Graham-Joux (2010)	$2^{0.337n} \approx (1.263)^n$	$2^{0.311n} \approx (1.241)^n$
$>$ Idea: *'enlarged number-set'* $\{-1, 0, 1\}$		
Becker-Coron-Joux (2011)	$2^{0.291n} \approx (1.223)^n$	$2^{0.291n} \approx (1.223)^n$

Table: Expected time and space requirements of algorithms solving Eq. (1).

Evolution of Algorithms (giving exact solutions)

Algorithm (year)	Time	Space
Exhaustive Search	$2^{1.000n} \approx (2.000)^n$	$2^{0.000n} \approx (1.000)^n$
$>$ Idea: *'meet-in-the-middle'*		
Horowitz-Sahni (1974)	$2^{0.500n} \approx (1.414)^n$	$2^{0.500n} \approx (1.414)^n$
$>$ Idea: *'4-list approach'*		
Schroeppel-Shamir (1979)	$2^{0.500n} \approx (1.414)^n$	$2^{0.250n} \approx (1.189)^n$
$>$ Idea: *'multiple representations'*		
Howgrave-Graham-Joux (2010)	$2^{0.337n} \approx (1.263)^n$	$2^{0.311n} \approx (1.241)^n$
$>$ Idea: *'enlarged number-set'* $\{-1, 0, 1\}$		
Becker-Coron-Joux (2011)	$2^{0.291n} \approx (1.223)^n$	$2^{0.291n} \approx (1.223)^n$
$>$ Idea: *'quantum algorithm'*		

Table: Expected time and space requirements of algorithms solving Eq. (1).

Evolution of Algorithms (giving exact solutions)

Algorithm (year)	**Time**	**Space**
Exhaustive Search	$2^{1.000n} \approx (2.000)^n$	$2^{0.000n} \approx (1.000)^n$
$>$ Idea: *'meet-in-the-middle'*		
Horowitz-Sahni (1974)	$2^{0.500n} \approx (1.414)^n$	$2^{0.500n} \approx (1.414)^n$
$>$ Idea: *'4-list approach'*		
Schroeppel-Shamir (1979)	$2^{0.500n} \approx (1.414)^n$	$2^{0.250n} \approx (1.189)^n$
$>$ Idea: *'multiple representations'*		
Howgrave-Graham-Joux (2010)	$2^{0.337n} \approx (1.263)^n$	$2^{0.311n} \approx (1.241)^n$
$>$ Idea: *'enlarged number-set'* $\{-1, 0, 1\}$		
Becker-Coron-Joux (2011)	$2^{0.291n} \approx (1.223)^n$	$2^{0.291n} \approx (1.223)^n$
$>$ Idea: *'quantum algorithm'*		
Bernstein-Jeffery-Lange-Meurer (2013)	$2^{0.241n} \approx (1.182)^n$	$2^{0.241n} \approx (1.182)^n$

Table: Expected time and space requirements of algorithms solving Eq. (1).

- Asymptotically best exact algorithm is a quantum algorithm

Evolution of Algorithms (giving exact solutions)

Algorithm (year)	Time	Space
Exhaustive Search	$2^{1.000n} \approx (2.000)^n$	$2^{0.000n} \approx (1.000)^n$
$>$ Idea: *'meet-in-the-middle'*		
Horowitz-Sahni (1974)	$2^{0.500n} \approx (1.414)^n$	$2^{0.500n} \approx (1.414)^n$
$>$ Idea: *'4-list approach'*		
Schroeppel-Shamir (1979)	$2^{0.500n} \approx (1.414)^n$	$2^{0.250n} \approx (1.189)^n$
$>$ Idea: *'multiple representations'*		
Howgrave-Graham-Joux (2010)	$2^{0.337n} \approx (1.263)^n$	$2^{0.311n} \approx (1.241)^n$
$>$ Idea: *'enlarged number-set'* $\{-1, 0, 1\}$		
Becker-Coron-Joux (2011)	$2^{0.291n} \approx (1.223)^n$	$2^{0.291n} \approx (1.223)^n$
$>$ Idea: *'quantum algorithm'*		
Bernstein-Jeffery-Lange-Meurer (2013)	$2^{0.241n} \approx (1.182)^n$	$2^{0.241n} \approx (1.182)^n$

Table: Expected time and space requirements of algorithms solving Eq. (1).

- Asymptotically best exact algorithm is a quantum algorithm
- Lower bound on running time is unkown

Outline

1 The Subset-Sum problem
- Motivation
- Historical Remarks
- Easy and Hard Instances
- Evolution of Algorithms
- Technique 1 - Meet in the Middle
- Technique 2 - Enlarge Number Set
- Applications

Technique 1 - Meet in the Middle

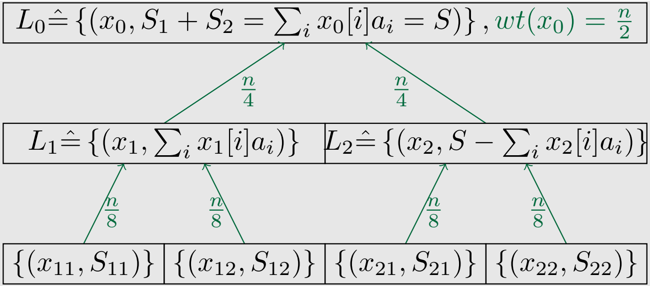


Figure: Schröppel-Shamir: Combining disjoint sub-problems of smaller weight.

Technique 1 - Meet in the Middle

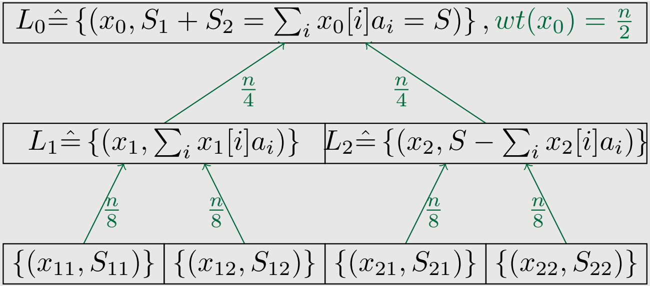


Figure: Schröppel-Shamir: Combining disjoint sub-problems of smaller weight.

- Identify subsets $I \subseteq [n]$ with vectors $x \in \{0,1\}^n : i \in I \Leftrightarrow x[i] = 1$

Technique 1 - Meet in the Middle

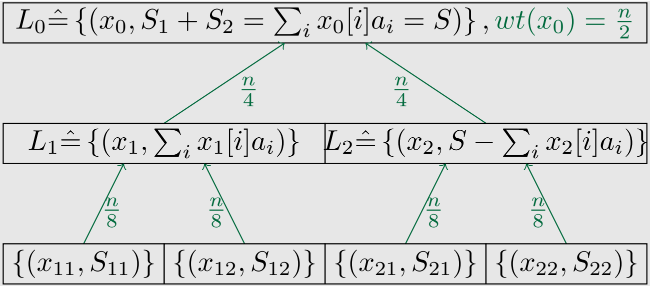


Figure: Schröppel-Shamir: Combining disjoint sub-problems of smaller weight.

- Identify subsets $I \subseteq [n]$ with vectors $x \in \{0,1\}^n : i \in I \Leftrightarrow x[i] = 1$
- Construct collisions based on the birthday-paradox

Technique 1 - Meet in the Middle

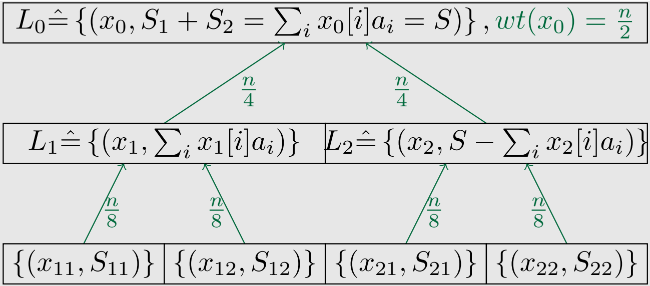


Figure: Schröppel-Shamir: Combining disjoint sub-problems of smaller weight.

- Identify subsets $I \subseteq [n]$ with vectors $x \in \{0,1\}^n : i \in I \Leftrightarrow x[i] = 1$
- Construct collisions based on the birthday-paradox
- Construct lists L_1, L_2 of (vector, sum)-pairs in smaller dimension

Technique 1 - Meet in the Middle

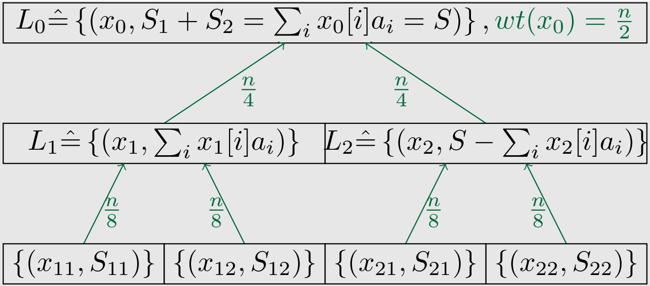


Figure: Schröppel-Shamir: Combining disjoint sub-problems of smaller weight.

- Identify subsets $I \subseteq [n]$ with vectors $x \in \{0, 1\}^n : i \in I \Leftrightarrow x[i] = 1$
- Construct collisions based on the birthday-paradox
- Construct lists L_1, L_2 of (vector, sum)-pairs in smaller dimension
- Merge L_1, L_2 to solutions $(x_0, S) \in L_0$ of the Subset-Sum problem.

Outline

Horst Görtz Institut für IT-Sicherheit

Technique 2 - Enlarge Number Set

$$\boxed{L_0 \hat{=} \left\{(x_0, \sum_i x_0[i]a_i = S_1 + S_2 = S)\right\}, wt(x_0) = \frac{n}{2}}$$

$$=$$

$$\boxed{L_1 \hat{=} \left\{(x_1, \sum_i x_1[i]a_i)\right\}, wt(x_1) = \frac{n}{4}}$$

$$+$$

$$\boxed{L_2 \hat{=} \left\{(x_2, S - \sum_i x_2[i]a_i)\right\}, wt(x_2) = \frac{n}{4}}$$

Figure: Adding length *n* solutions of sub-problems enlarges the number-set.

Technique 2 - Enlarge Number Set

$$\boxed{L_0 \hat{=} \left\{ \left(x_0, \sum_i x_0[i]a_i = S_1 + S_2 = S \right) \right\}, wt(x_0) = \frac{n}{2}}$$

$$=$$

$$\boxed{L_1 \hat{=} \left\{ \left(x_1, \sum_i x_1[i]a_i \right) \right\}, wt(x_1) = \frac{n}{4}}$$

$$+$$

$$\boxed{L_2 \hat{=} \left\{ \left(x_2, S - \sum_i x_2[i]a_i \right) \right\}, wt(x_2) = \frac{n}{4}}$$

Figure: Adding length *n* solutions of sub-problems enlarges the number-set.

- Merge non-disjoint partial solutions in L_1, L_2 filtering 'inconsistent'

Technique 2 - Enlarge Number Set

$$\boxed{L_0 \hat{=} \left\{\left(x_0, \sum_i x_0[i]a_i = S_1 + S_2 = S\right)\right\}, wt(x_0) = \frac{n}{2}}$$

$$=$$

$$\boxed{L_1 \hat{=} \left\{\left(x_1, \sum_i x_1[i]a_i\right)\right\}, wt(x_1) = \frac{n}{4}}$$

$$+$$

$$\boxed{L_2 \hat{=} \left\{\left(x_2, S - \sum_i x_2[i]a_i\right)\right\}, wt(x_2) = \frac{n}{4}}$$

Figure: Adding length n solutions of sub-problems enlarges the number-set.

- Merge non-disjoint partial solutions in L_1, L_2 filtering 'inconsistent'
- Enlarge intermediate 'number-set': $x_{11}[i] + x_{12}[i] =: x_1[i] \notin \{0, 1\}$.

Technique 2 - Enlarge Number Set

$$\boxed{L_0 \hat{=} \left\{(x_0, \sum_i x_0[i]a_i = S_1 + S_2 = S)\right\}, wt(x_0) = \frac{n}{2}}$$

$$=$$

$$\boxed{L_1 \hat{=} \left\{(x_1, \sum_i x_1[i]a_i)\right\}, wt(x_1) = \frac{n}{4}}$$

$$+$$

$$\boxed{L_2 \hat{=} \left\{(x_2, S - \sum_i x_2[i]a_i)\right\}, wt(x_2) = \frac{n}{4}}$$

Figure: Adding length n solutions of sub-problems enlarges the number-set.

- Merge non-disjoint partial solutions in L_1, L_2 filtering 'inconsistent'
- Enlarge intermediate 'number-set': $x_{11}[i] + x_{12}[i] =: x_1[i] \notin \{0, 1\}$.
- Speed-ups if $x_1[i], x_2[i] \sim \mathcal{D}(\{-1, 0, 1\})$ w.r.t. distribution $\mathcal{D}$.

Outline

Applications

- These cryptanalytic methods are meta-techniques

Applications

- These cryptanalytic methods are meta-techniques
- Applicable to lattice- or code-based $\mathcal{NP}$-`complete` problems

Applications

- These cryptanalytic methods are meta-techniques
- Applicable to lattice- or code-based $\mathcal{NP}$-`complete` problems

Example 2 (Outsourcing a database to the cloud [1])

Applications

- These cryptanalytic methods are meta-techniques
- Applicable to lattice- or code-based $\mathcal{NP}$-`complete` problems

Example 2 (Outsourcing a database to the cloud [1])

Setup: A preprocessed length n text T is stored in the cloud.

Applications

- These cryptanalytic methods are meta-techniques
- Applicable to lattice- or code-based $\mathcal{NP}$-`complete` problems

Example 2 (Outsourcing a database to the cloud [1])

Setup: A preprocessed length n text T is stored in the cloud.
Given: A pattern P of length $m \leq n$, find all locations of P in T.

Applications

- These cryptanalytic methods are meta-techniques
- Applicable to lattice- or code-based $\mathcal{NP}$-`complete` problems

Example 2 (Outsourcing a database to the cloud [1])

Setup: A preprocessed length n text T is stored in the cloud.
Given: A pattern P of length $m \leq n$, find all locations of P in T.
Goal: Cloud learns no information about text T and pattern P.

S. Faust, C. Hazay, and D. Venturi.
Outsourced pattern matching.
Cryptology ePrint Archive, Report 2014/662, 2014.
http://eprint.iacr.org/2014/662.

RUB

RUHR-UNIVERSITÄT BOCHUM

QUESTIONS?

Thank you for your attention!

hgi
Horst Görtz Institut
für IT-Sicherheit