Computing Information Flow Using Symbolic Model-Checking

Rohit Chadha [1] Umang Mathur [2] Stefan Schwoon [3]

[1]University of Missouri
Columbia, Missouri, USA

[2]Indian Institute of Technology - Bombay
Mumbai

[3]LSV, ENS Cachan
France

December 17, 2014

Outline

Introduction

- Quantifying information leakage - Inferring information about inputs by observing public outputs

Introduction

- Quantifying information leakage - Inferring information about inputs by observing public outputs
- No leakage $\implies$ Outputs independent of inputs

Introduction

- Quantifying information leakage - Inferring information about inputs by observing public outputs
- No leakage $\implies$ Outputs independent of inputs
- Full leakage $\implies$ Unique input corresponding to given output

Introduction

- Quantifying information leakage - Inferring information about inputs by observing public outputs
- No leakage $\implies$ Outputs independent of inputs
- Full leakage $\implies$ Unique input corresponding to given output
- Comparing leakage across programs - less leakage is desirable

Measuring Information Leakage

Measuring Information Leakage

Several metrics - min-entropy, Shannon's entropy, etc.,

Measuring Information Leakage

Several metrics - min-entropy, Shannon's entropy, etc.,

1. Min-entropy leakage measures vulnerability of the secret inputs to being guessed correctly in a single attempt of the adversary

$$\mathrm{ME}_{\mathsf{U}}(P) = \log \sum_{o \in \mathcal{O}} \max_{s \in \mathcal{S}} \mu(\mathcal{S} = s \mid \mathcal{O} = o).$$

Measuring Information Leakage

Several metrics - min-entropy, Shannon's entropy, etc.,

1. Min-entropy leakage measures vulnerability of the secret inputs to being guessed correctly in a single attempt of the adversary

$$\mathrm{ME}_{\mathsf{U}}(P) = \log \sum_{o \in \mathcal{O}} \max_{s \in \mathcal{S}} \mu(\mathcal{S} = s \mid \mathcal{O} = o).$$

2. Shannon entropy leakage measures expected number of guesses required to correctly guess the secret input

$$\mathrm{SE}_{\mathsf{U}}(P) = \log |S| - \frac{1}{|S|} \sum_{o \in O} |P^{-1}(o)| \log |P^{-1}(o)|$$

Example

Consider the following example:

Example

Consider the following example:

```
def example (input) :
    output = input % 8
    return output
```

Example

Consider the following example:

```
def example (input) :
    output = input % 8
    return output
```

What would be the information leaked by the above program

Example

Consider the following example:

```
def example (input) :
    output = input % 8
    return output
```

What would be the information leaked by the above program

- using min-entropy ?

Example

Consider the following example:

```
def example (input) :
    output = input % 8
    return output
```

What would be the information leaked by the above program

- using min-entropy ?
- using Shannon entropy ?

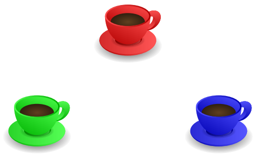
Dining Cryptographers

Dining Cryptographers

- Cryptographers A, B and C: Dine out

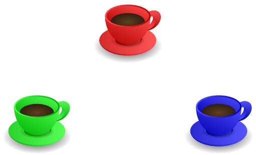
Dining Cryptographers

- Cryptographers A, B and C: Dine out



Dining Cryptographers

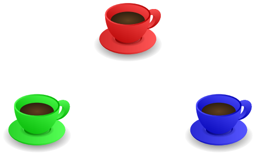
- Cryptographers A, B and C: Dine out



- Payment done by

Dining Cryptographers

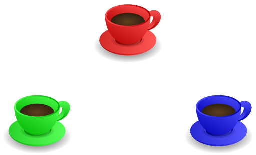
- Cryptographers A, B and C: Dine out



- Payment done by
  - One of A, B or C, or

Dining Cryptographers

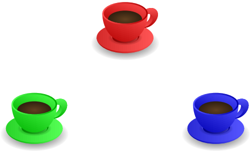
- Cryptographers A, B and C: Dine out



- Payment done by
  - One of A, B or C, or
  - NSA

Dining Cryptographers

- Cryptographers A, B and C: Dine out



- Payment done by
  - One of A, B or C, or
  - NSA
- Determine if the NSA paid or not w/o revealing information about cryptographers

Dining Cryptographers: Protocol

2 stage protocol:

Dining Cryptographers: Protocol

2 stage protocol:

1. Every two cryptographers establish a shared one-bit secret : Toss a coin

Dining Cryptographers: Protocol

2 stage protocol:

1. Every two cryptographers establish a shared one-bit secret : Toss a coin
2. Each cryptographer publicly announces a bit, which is

Dining Cryptographers: Protocol

2 stage protocol:

1. Every two cryptographers establish a shared one-bit secret : Toss a coin
2. Each cryptographer publicly announces a bit, which is
   - XOR of shared bits, if did not pay

Dining Cryptographers: Protocol

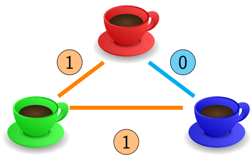
2 stage protocol:

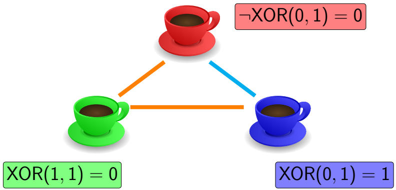
1. Every two cryptographers establish a shared one-bit secret : Toss a coin
2. Each cryptographer publicly announces a bit, which is
   - XOR of shared bits, if did not pay
   - $\neg$ (XOR of shared bits), otherwise

Dining Cryptographers: Protocol

2 stage protocol:

1. Every two cryptographers establish a shared one-bit secret : Toss a coin
2. Each cryptographer publicly announces a bit, which is
   - XOR of shared bits, if did not pay
   - ¬ (XOR of shared bits), otherwise

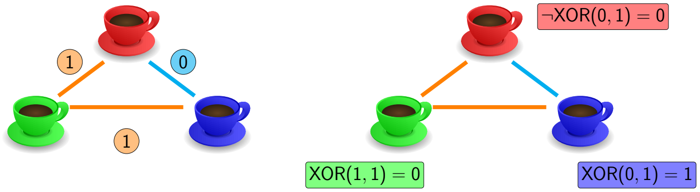




Dining Cryptographers: Protocol

2 stage protocol:

1. Every two cryptographers establish a shared one-bit secret : Toss a coin
2. Each cryptographer publicly announces a bit, which is
   - XOR of shared bits, if did not pay
   - $\neg$ (XOR of shared bits), otherwise

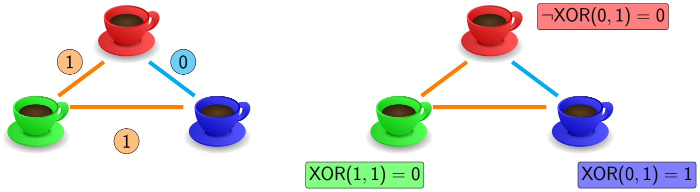


Stage-1 (left) and Stage-2 (right)

Dining Cryptographers: Protocol

2 stage protocol:

1. Every two cryptographers establish a shared one-bit secret : Toss a coin
2. Each cryptographer publicly announces a bit, which is
   - XOR of shared bits, if did not pay
   - ¬ (XOR of shared bits), otherwise



Stage-1 (left) and Stage-2 (right)

$$\text{XOR}(\text{Announcement}_A, \text{Announcement}_B, \text{Announcement}_C) = 0$$

iff

NSA paid for the dinner

Probabilistic Boolean Programs

- Global variablesn $\mathcal{G}$: Input and output

Probabilistic Boolean Programs

- Global variablesn $\mathcal{G}$: Input and output
- Local variables: Internal calculations

Probabilistic Boolean Programs

- Global variablesn $\mathcal{G}$: Input and output
- Local variables: Internal calculations
- Program statements : transform global and local variables

Probabilistic Boolean Programs

- Global variablesn $\mathcal{G}$: Input and output
- Local variables: Internal calculations
- Program statements : transform global and local variables
- For Program P, $F_P : 2^{\mathcal{G}} \rightarrow 2^{\mathcal{G}} \cup \{\perp\}$

Probabilistic Boolean Programs

- Global variablesn $\mathcal{G}$: Input and output
- Local variables: Internal calculations
- Program statements : transform global and local variables
- For Program P, $F_P : 2^{\mathcal{G}} \rightarrow 2^{\mathcal{G}} \cup \{\perp\}$
- $F_P(\bar{g_0}) = \perp$ iff P does not terminate

Probabilistic Boolean Programs

- Global variablesn $\mathcal{G}$: Input and output
- Local variables: Internal calculations
- Program statements : transform global and local variables
- For Program P, $F_P\colon 2^{\mathcal{G}} \to 2^{\mathcal{G}} \cup \{\perp\}$
- $F_P(\bar{g_0}) = \perp$ iff P does not terminate
- Summary - Joint probability distribution μ

Probabilistic Boolean Programs

- Global variablesn $\mathcal{G}$: Input and output
- Local variables: Internal calculations
- Program statements : transform global and local variables
- For Program P, $F_P \colon 2^{\mathcal{G}} \to 2^{\mathcal{G}} \cup \{\perp\}$
- $F_P(\bar{g_0}) = \perp$ iff P does not terminate
- Summary - Joint probability distribution μ

Algebraic Decision Diagrams

- Set of variables $\mathcal{V}$

Algebraic Decision Diagrams

- Set of variables $\mathcal{V}$
- Algebraic set M ($M = [0, 1]$ for probabilistic statements, $M = \{0, 1\}$ implies BDDs)

Algebraic Decision Diagrams

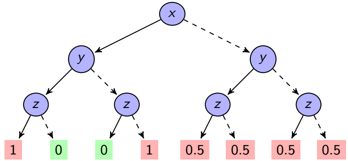
- Set of variables $\mathcal{V}$
- Algebraic set M ($M = [0, 1]$ for probabilistic statements, $M = \{0, 1\}$ implies BDDs)
- ADD : $2^{\mathcal{V}} \to M$

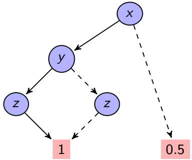
Algebraic Decision Diagrams

- Set of variables $\mathcal{V}$
- Algebraic set M ($M = [0, 1]$ for probabilistic statements, $M = \{0, 1\}$ implies BDDs)
- ADD : $2^{\mathcal{V}} \rightarrow M$
- Efficient reduced representations, similar to BDDs

Algebraic Decision Diagrams

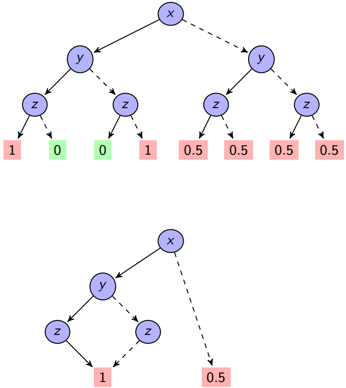
- Set of variables $\mathcal{V}$
- Algebraic set M ($M = [0,1]$ for probabilistic statements, $M = \{0,1\}$ implies BDDs)
- ADD : $2^{\mathcal{V}} \rightarrow M$
- Efficient reduced representations, similar to BDDs





Algebraic Decision Diagrams

- Set of variables $\mathcal{V}$
- Algebraic set M ($M = [0, 1]$ for probabilistic statements, $M = \{0, 1\}$ implies BDDs)
- ADD : $2^{\mathcal{V}} \to M$
- Efficient reduced representations, similar to BDDs



ADD (up) and its reduced form (bottom)

Computing Summaries: Fixed Point Iteration

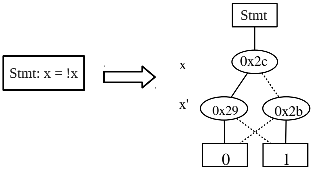
- Program statement $l \rightarrow \mu_l$

Computing Summaries: Fixed Point Iteration

- Program statement $l \rightarrow \mu_l$
- Can be represented efficiently as MTBBDs

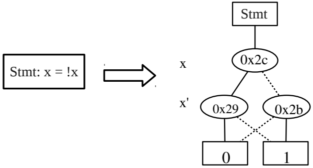
Computing Summaries: Fixed Point Iteration

- Program statement $I \to \mu_I$
- Can be represented efficiently as MTBBDs



Computing Summaries: Fixed Point Iteration

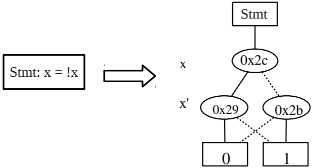
- Program statement $I \to \mu_I$
- Can be represented efficiently as MTBBDs



- Compose statements

Computing Summaries: Fixed Point Iteration

- Program statement $I \rightarrow \mu_I$
- Can be represented efficiently as MTBBDs



- Compose statements
- Arrive at a fixed point (Summary μ)

Min Entropy : Symbolic Algorithm

For a program P, with

Min Entropy : Symbolic Algorithm

For a program P, with

- input set S (uniform distribution),

Min Entropy : Symbolic Algorithm

For a program P, with

- input set S (uniform distribution),
- output set O, and,

Min Entropy : Symbolic Algorithm

For a program P, with

- input set S (uniform distribution),
- output set O, and,
- joint probability distribution μ,

Min Entropy : Symbolic Algorithm

For a program P, with

- input set S (uniform distribution),
- output set O, and,
- joint probability distribution μ,

the min-entropy leakage $\mathrm{ME_U}(P)$ is

$$\mathrm{ME_U}(P) = \log \sum_{o \in O} \max_{s \in S} \mu(\mathcal{S} = s \mid \mathcal{O} = o).$$

Min Entropy : Symbolic Algorithm

For a program P, with

- input set S (uniform distribution),
- output set O, and,
- joint probability distribution μ,

the min-entropy leakage $\mathrm{ME}_{\mathrm{U}}(P)$ is

$$\mathrm{ME}_{\mathrm{U}}(P) = \log \sum_{o \in O} \max_{s \in S} \mu(\mathcal{S} = s \mid \mathcal{O} = o).$$

Algorithm 6: Symbolic computation of min-entropy leakage of a probabilistic program

Input: $\mathcal{G}, \mathcal{G}'$ and T_P the summary of P.

Output: $\mathrm{ME}_{\mathrm{U}}(P)$

begin

  $T_{\mathsf{out},P} \longleftarrow \mathsf{abstract}(\max, \mathcal{G}, T_P)$

  $\mathsf{sum}_{\mathsf{out}} \longleftarrow \mathsf{val}(\mathsf{abstract}(+, \mathcal{G}', T_{\mathsf{out},P}))$

  $T_{\mathsf{term},P} \longleftarrow \mathsf{abstract}(+, \mathcal{G}', T_P)$

  $\mathsf{sum}_{\mathsf{out}} \longleftarrow \mathsf{sum}_{\mathsf{out}} + (1 - \mathsf{val}(\mathsf{abstract}(\min, \mathcal{G}, T_{\mathsf{term},P})));$

  return $\log sum_{out}$

Shannon Entropy : Symbolic Algorithm

$$\mathrm{SE}_{\mathsf{U}}(P) = \log |S| - \frac{1}{|S|} \sum_{o \in O} |P^{-1}(o)| \log |P^{-1}(o)|$$

Shannon Entropy : Symbolic Algorithm

$$\mathrm{SE}_{\mathrm{U}}(P) = \log|S| - \frac{1}{|S|}\sum_{o\in O}|P^{-1}(o)|\log|P^{-1}(o)|$$

Algorithm 8: Symbolic computation of Shannon entropy leakage of a probabilistic program

Input: $\mathcal{G}, \mathcal{G}'$ and T_P the summary of P.
Output: $\mathrm{SE}_{\mathrm{U}}(P)$

Let n be the number of variables in $\mathcal{G}$.

begin

- $T_{\text{norm-eq-size},P} \longleftarrow \text{divide}(\text{abstract}(+, \mathcal{G}, T_P), 2^n)$
- $\text{val}_{\text{out}} \longleftarrow (\text{- val}(\text{abstract}(\star, \mathcal{G}', T_{\text{norm-eq-size},P})))$
- $T_{\text{term},P} \longleftarrow \text{abstract}(+, \mathcal{G}', T_P)$
- $\text{prob}_{\text{out,non-term}} \longleftarrow (1 - \frac{\text{val}(\text{abstract}(+, \mathcal{G}, T_{\text{term},P}))}{2^n})$
- $\text{val}_{\text{out,non-term}} \longleftarrow (\text{- prob}_{\text{out,non-term}} \log \text{prob}_{\text{out,non-term}})$
- $T_{\text{norm-}\star\text{out},P} \longleftarrow \text{divide}(\text{abstract}(\star, \mathcal{G}', T_P), 2^n)$
- $\text{val}_{\text{cond}} \longleftarrow (\text{-val}(\text{abstract}(+, \mathcal{G}, T_{\star\text{out},P})))$
- $T_{\text{non-term},P} \longleftarrow \text{subtract}(1, T_{\text{term},P})$
- $\text{val}_{\text{cond,non-term}} \longleftarrow (\text{-}\frac{\text{val}(\text{abstract}(\star, \mathcal{G}, T_{\text{non-term-prob},P}))}{2^n})$
- **return** $(val_{out} + val_{out,non\text{-}term} - val_{cond} - val_{cond,non\text{-}term})$

Moped-QLeak

- Tool `Moped-QLeak`: extends `Moped`

Moped-QLeak

- Tool `Moped-QLeak`: extends `Moped`
- Source - C/C++

Moped-QLeak

- Tool `Moped-QLeak`: extends `Moped`
- Source - C/C++
- Input language *Remopla* - arrays, integers, struct's, etc.,

Moped-QLeak

- Tool `Moped-QLeak`: extends `Moped`
- Source - C/C++
- Input language *Remopla* - arrays, integers, struct's, etc.,

```
define N 32

define DEFAULT_INT_BITS N

unsigned int var1;
bool g;

module void f(unsigned int v, bool z){

    bool k;
    pchoice
    :: 0.2 -> label2: k = g && z;
    :: 0.8 -> var1 = var1 + v;
    choicep
}

module void main(){

    var1 = 53;
    pchoice
    :: 0.3 -> label1: g = true;
    :: 0.7 ->  f(var1, !g);
    choicep
}
```

Moped-QLeak

Modifications/Optimizations made:

Moped-QLeak

Modifications/Optimizations made:

- Algebraic operations

Moped-QLeak

Modifications/Optimizations made:

- Algebraic operations
- Variable orderings - manual

Moped-QLeak

Modifications/Optimizations made:

- Algebraic operations
- Variable orderings - manual

Salient features:

Moped-QLeak

Modifications/Optimizations made:

- Algebraic operations
- Variable orderings - manual

Salient features:

- Handles large number of bits (30 bits)

Moped-QLeak

Modifications/Optimizations made:

- Algebraic operations
- Variable orderings - manual

Salient features:

- Handles large number of bits (30 bits)
- Time taken in miliseconds

Moped-QLeak

Modifications/Optimizations made:

- Algebraic operations
- Variable orderings - manual

Salient features:

- Handles large number of bits (30 bits)
- Time taken in miliseconds
- Consistently outperforms **sqifc** (Malacaria et. al)

Moped-QLeak

Example	Order	ME	SE	Time	Data types
Illustrative Example	I	3	2.03966e-05	0.215	bool
Electronic Purse	D	2	2	0.009	5 bit integers (Restricted)
Mix and Duplicate	S	16	16	0.041	bool
Binary Search	I	16	16	9.307	bool
Sanity Check	I	4	1.168e-7	0.060	bool
Implicit Flow	D	2.8074	1.757e-07	0.016	30 bit integers
Implicit Flow	D	2.8074	0.003	0.010	15 bit integers
Implicit Flow	D	2.8074	4.67189e-08	0.190	bool
Masked Copy	I	16	16	0.038	bool
Sum Query	D	4.80735	4.35132	0.034	5 bit integers (Restricted)

Related Work

- (Köpf et. al.,) : iteratively refine equivalence classes (deterministic only)

Related Work

- (Köpf et. al.,) : iteratively refine equivalence classes (deterministic only)
- (Klebanov et. al.,) : program to SMT formula, count outputs (deterministic, loop free only)

Related Work

- (Köpf et. al.,) : iteratively refine equivalence classes (deterministic only)
- (Klebanov et. al.,) : program to SMT formula, count outputs (deterministic, loop free only)
- (Biondi et. al.,) : forward symbolic execution - performance comparable to **sqifc**

Related Work

- (Köpf et. al.,) : iteratively refine equivalence classes (deterministic only)
- (Klebanov et. al.,) : program to SMT formula, count outputs (deterministic, loop free only)
- (Biondi et. al.,) : forward symbolic execution - performance comparable to **sqifc**

Tool demonstration

Conclusions and Future Work

- Symbolic algorithms for measuring information leakage

Conclusions and Future Work

- Symbolic algorithms for measuring information leakage
- Interagble in any BDD based reachability analysis tool

Conclusions and Future Work

- Symbolic algorithms for measuring information leakage
- Interagble in any BDD based reachability analysis tool
- Summary calculation is the overhead - BDD size (algebraic operations) and variable orderings

Conclusions and Future Work

- Symbolic algorithms for measuring information leakage
- Interagble in any BDD based reachability analysis tool
- Summary calculation is the overhead - BDD size (algebraic operations) and variable orderings
- Future work:

Conclusions and Future Work

- Symbolic algorithms for measuring information leakage
- Interagble in any BDD based reachability analysis tool
- Summary calculation is the overhead - BDD size (algebraic operations) and variable orderings
- Future work:
  - Recursive algorithms

Conclusions and Future Work

- Symbolic algorithms for measuring information leakage
- Interagble in any BDD based reachability analysis tool
- Summary calculation is the overhead - BDD size (algebraic operations) and variable orderings
- Future work:
  - Recursive algorithms
  - Other symbolic model-checking frameworks - CEGAR

Conclusions and Future Work

- Symbolic algorithms for measuring information leakage
- Interagble in any BDD based reachability analysis tool
- Summary calculation is the overhead - BDD size (algebraic operations) and variable orderings
- Future work:
  - Recursive algorithms
  - Other symbolic model-checking frameworks - CEGAR

Thank You !