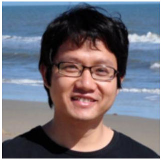
Verifying Absence of Side Channels using Cartesian Hoare Logic


Işıl Dillig


Yu Feng


Jia Chen


M. Heule

THE UNIVERSITY OF TEXAS AT AUSTIN

Resource Side Channel Vulnerabilities

Program has side channel if resource usage varies depending on secret.

Resource Side Channel Vulnerabilities

Program has side channel if resource usage varies depending on secret.

- Formalization similar to **non-interference**.

Resource Side Channel Vulnerabilities

Program has side channel if resource usage varies depending on secret.

- Formalization similar to **non-interference**.
- Let H, L denote the high (secret) and low (non-secret) inputs, and let $R_P(I)$ denote resource usage of P on input I.

Resource Side Channel Vulnerabilities

Program has side channel if resource usage varies depending on secret.

- Formalization similar to **non-interference**.
- Let H, L denote the high (secret) and low (non-secret) inputs, and let $R_P(I)$ denote resource usage of P on input I.

$$\boxed{\exists H_1, H_2, L.\ R_P(H_1, L) \neq R_P(H_2, L)}$$

Bounded non-interference

Problem: Definition is too strong!

Bounded non-interference

Problem: Definition is too strong!

- If difference in resource usage is very small, vulnerability may not be **exploitable**

Bounded non-interference

Problem: Definition is too strong!

- If difference in resource usage is very small, vulnerability may not be **exploitable**
- To relax this definition, we propose **bounded non-interference:**

Bounded non-interference

Problem: Definition is too strong!

- If difference in resource usage is very small, vulnerability may not be **exploitable**
- To relax this definition, we propose **bounded non-interference:**

$$\forall H_1, H_2, L.\ |R_P(H_1, L) - R_P(H_2, L)| < \epsilon$$

Verifying Bounded Non-interference

- Just like non-interference, bounded non-interference is a **2-safety** property

Verifying Bounded Non-interference

- Just like non-interference, bounded non-interference is a **2-safety** property
- In general, **k-safety** properties relate the behavior of k-different runs

Verifying Bounded Non-interference

- Just like non-interference, bounded non-interference is a **2-safety** property
- In general, **k-safety** properties relate the behavior of k-different runs

- To verify bounded non-interference, need technique for verifying k-safety

Self-composition

- Simplest technique to verify k-safety is **self-composition (SC)**
- Given 2-safety property F, self composition sequentially composes two alpha-renamed copies of the program

```
Verify(P, F) {
  P[x1/x];
  P[x2/x];
  assert(F);
}
```

Self-composition

- Simplest technique to verify k-safety is **self-composition (SC)**
- Given 2-safety property F, self composition sequentially composes two alpha-renamed copies of the program

```
Verify(P, F) {
  P[x1/x];
  P[x2/x];
  assert(F);
}
```

Unfortunately, self-composition does not work well in practice

Why Self-composition Doesn't Work

- To see why SC doesn't work, consider this program

```
f(int[] secret,
  int k, int n) {

 i:=0;
 while(i<n){
   if(secret[i])
     foo(); //r+=c
   else
     bar(); //r+=c
  i:=i+k;
 }

}
```

Why Self-composition Doesn't Work

- To see why SC doesn't work, consider this program
- With SC, we obtain this new program

```
 i1:=0;
 while(i1<n){
   if(secret1[i1])
     foo(); //r1+=c
   else
     bar(); //r1+=c
  i1:=i1+k;
 }
i2:=0;
while(i2<n){
   if(secret2[i2])
     foo(); //r2+=c
   else
     bar(); //r2+=c
  i2:=i2+k;
 }
assert(r1 - r2 < b);
```

Why Self-composition Doesn't Work

- To see why SC doesn't work, consider this program
- With SC, we obtain this new program
- Unfortunately, need to know precise values of r1, r2 after each loop to prove assertion, but this is very difficult

```
 i1:=0;
 while(i1<n){
   if(secret1[i1])
     foo(); //r1+=c
   else
     bar(); //r1+=c
  i1:=i1+k;
 }
i2:=0;
while(i2<n){
   if(secret2[i2])
     foo(); //r2+=c
   else
     bar(); //r2+=c
  i2:=i2+k;
 }
assert(r1 - r2 < b);
```

Key idea: Synchronizing loops

- Key idea underlying k-safety verification is to "execute" loops from different executions in lock step!

Key idea: Synchronizing loops

- Key idea underlying k-safety verification is to "execute" loops from different executions in lock step!
- In a nutshell, this corresponds to executing the following program:

Key idea: Synchronizing loops

- Key idea underlying k-safety verification is to "execute" loops from different executions in lock step!

- In a nutshell, this corresponds to executing the following program:

```
  i1:=0;i2:=0;
  while(i1<n && i2 <n){
    if(secret1[i1]) foo(); //r1+=c
    else bar(); //r1+=c
    if(secret2[i2]) foo(); //r2+=c
    else bar(); //r2+=c
    i1:=i1+k;
    i2:=i2+k;
  }
assert(r1 - r2 < b);
```

Key idea: Synchronizing loops

- Key idea underlying k-safety verification is to "execute" loops from different executions in lock step!

- In a nutshell, this corresponds to executing the following program:

```
i1:=0;i2:=0;
while(i1<n && i2 <n){
  if(secret1[i1]) foo(); //r1+=c
  else bar(); //r1+=c
  if(secret2[i2]) foo(); //r2+=c
  else bar(); //r2+=c
  i1:=i1+k;
  i2:=i2+k;
}
assert(r1 - r2 < b);
```

Easy to infer r1=r2!

Cartesian Hoare Logic (CHL)

- Our PLDI'16 paper introduces **Cartesian Hoare Logic (CHL)** for verifying k-safety

$$\text{(Expand)}\quad \frac{\vdash \langle\Phi\rangle\ \boxed{\circledast} S^n\ \langle\Psi\rangle}{\vdash \|\Phi\|\ S\ \|\Psi\|}$$

$$\text{(Lift)}\quad \frac{\vdash \{\Phi\}\ S\ \{\Psi\}}{\vdash \langle\Phi\rangle\ S\ \langle\Psi\rangle}$$

$$(\flat\text{−intro 1})\quad \frac{\vdash \langle\Phi\rangle\ S;\flat \circledast \chi\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ S \circledast \chi\ \langle\Psi\rangle}$$

$$(\flat\text{−intro 2})\quad \frac{\vdash \langle\Phi\rangle\ \chi \circledast \flat\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ \chi\ \langle\Psi\rangle}$$

$$(\flat\text{−elim})\quad \frac{\vdash \langle\Phi\rangle\ \chi\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ \chi \circledast \flat\ \langle\Psi\rangle}$$

$$\text{(Assoc)}\quad \frac{\vdash \langle\Phi\rangle\ \chi_1 \circledast (\chi_2 \circledast \chi_3)\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ (\chi_1 \circledast \chi_2) \circledast \chi_3\ \langle\Psi\rangle}$$

$$\text{(Comm)}\quad \frac{\vdash \langle\Phi\rangle\ \chi_2 \circledast \chi_1\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ \chi_1 \circledast \chi_2\ \langle\Psi\rangle}$$

$$\text{(Step)}\quad \frac{\vdash \{\Phi\}\ S_1\ \{\Phi'\} \quad \vdash \langle\Phi'\rangle\ S_2 \circledast \chi\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ S_1; S_2 \circledast \chi\ \langle\Psi\rangle}$$

$$\text{(Havoc)}\quad \frac{V = \text{vars}(\chi)}{\vdash \langle\Phi\rangle\ \chi\ \langle\exists V.\Phi\rangle}$$

$$\text{(Consq)}\quad \frac{\Phi \Rightarrow \Phi' \quad \Psi' \Rightarrow \Psi \quad \vdash \langle\Phi'\rangle\ \chi\ \langle\Psi'\rangle}{\vdash \langle\Phi\rangle\ \chi\ \langle\Psi\rangle}$$

$$\text{(Seq)}\quad \frac{\begin{array}{c}\vdash \langle\Phi\rangle\ (S_1 \circledast \ldots \circledast S_n)\ \langle\Phi'\rangle \\ \vdash \langle\Phi'\rangle\ (R_1 \circledast \ldots \circledast R_n) \circledast \chi\ \langle\Psi\rangle\end{array}}{\vdash \langle\Phi\rangle\ (S_1; R_1 \circledast \ldots \circledast S_n; R_n) \circledast \chi\ \langle\Psi\rangle}$$

$$\text{(If)}\quad \frac{\begin{array}{c}\vdash \langle\Phi \wedge c\rangle\ S_1; S \circledast \chi\ \langle\Psi\rangle \\ \vdash \langle\Phi \wedge \neg c\rangle\ S_2; S \circledast \chi\ \langle\Psi\rangle\end{array}}{\vdash \langle\Phi\rangle\ (S_1 \oplus_c S_2; S) \circledast \chi\ \langle\Psi\rangle}$$

$$\text{(Loop)}\quad \frac{\begin{array}{c}\Phi \Rightarrow \mathbb{I} \\ \vdash \{\mathbb{I}\}\ [(c_1, S_1, \flat), \ldots, (c_n, S_n, \flat)]^*\ \{\mathbb{I}\} \\ \vdash \langle \text{post}_i(\mathbb{I}) \wedge \neg c_i\rangle\ S_i' \circledast \chi\ \langle\Psi\rangle \quad (\forall i \in [1, n])\end{array}}{\vdash \langle\Phi\rangle\ [(c_1, S_1, S_1'), \ldots, (c_n, S_n, S_n')]^* \circledast \chi\ \langle\Psi\rangle}$$

Cartesian Hoare Logic (CHL)

- Our PLDI'16 paper introduces **Cartesian Hoare Logic (CHL)** for verifying k-safety
- CHL proof rules allow synchronization between loops

$$\text{(Expand)} \quad \frac{\vdash \langle\Phi\rangle \boxast S^n \langle\Psi\rangle}{\vdash \|\Phi\|\ S\ \|\Psi\|}$$

$$\text{(Lift)} \quad \frac{\vdash \{\Phi\}\ S\ \{\Psi\}}{\vdash \langle\Phi\rangle\ S\ \langle\Psi\rangle}$$

$$(\flat\text{−intro 1}) \quad \frac{\vdash \langle\Phi\rangle\ S;\flat \circledast \chi\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ S \circledast \chi\ \langle\Psi\rangle}$$

$$(\flat\text{−intro 2}) \quad \frac{\vdash \langle\Phi\rangle\ \chi \circledast \flat\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ \chi\ \langle\Psi\rangle}$$

$$(\flat\text{−elim}) \quad \frac{\vdash \langle\Phi\rangle\ \chi\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ \chi \circledast \flat\ \langle\Psi\rangle}$$

$$\text{(Assoc)} \quad \frac{\vdash \langle\Phi\rangle\ \chi_1 \circledast (\chi_2 \circledast \chi_3)\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ (\chi_1 \circledast \chi_2) \circledast \chi_3\ \langle\Psi\rangle}$$

$$\text{(Comm)} \quad \frac{\vdash \langle\Phi\rangle\ \chi_2 \circledast \chi_1\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ \chi_1 \circledast \chi_2\ \langle\Psi\rangle}$$

$$\text{(Step)} \quad \frac{\vdash \{\Phi\}\ S_1\ \{\Phi'\} \quad \vdash \langle\Phi'\rangle\ S_2 \circledast \chi\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ S_1; S_2 \circledast \chi\ \langle\Psi\rangle}$$

$$\text{(Havoc)} \quad \frac{V = \mathrm{vars}(\chi)}{\vdash \langle\Phi\rangle\ \chi\ \langle\exists V.\Phi\rangle}$$

$$\text{(Consq)} \quad \frac{\Phi \Rightarrow \Phi' \quad \Psi' \Rightarrow \Psi \quad \vdash \langle\Phi'\rangle\ \chi\ \langle\Psi'\rangle}{\vdash \langle\Phi\rangle\ \chi\ \langle\Psi\rangle}$$

$$\text{(Seq)} \quad \frac{\begin{array}{c}\vdash \langle\Phi\rangle\ (S_1 \circledast \ldots \circledast S_n)\ \langle\Phi'\rangle \\ \vdash \langle\Phi'\rangle\ (R_1 \circledast \ldots \circledast R_n) \circledast \chi\ \langle\Psi\rangle\end{array}}{\vdash \langle\Phi\rangle\ (S_1; R_1 \circledast \ldots \circledast S_n; R_n) \circledast \chi\ \langle\Psi\rangle}$$

$$\text{(If)} \quad \frac{\begin{array}{c}\vdash \langle\Phi \land c\rangle\ S_1; S \circledast \chi\ \langle\Psi\rangle \\ \vdash \langle\Phi \land \neg c\rangle\ S_2; S \circledast \chi\ \langle\Psi\rangle\end{array}}{\vdash \langle\Phi\rangle\ (S_1 \oplus_c S_2; S) \circledast \chi\ \langle\Psi\rangle}$$

$$\text{(Loop)} \quad \frac{\begin{array}{c}\Phi \Rightarrow \mathbb{I} \\ \vdash \{\mathbb{I}\}\ [(c_1, S_1, \flat), \ldots, (c_n, S_n, \flat)]^*\ \{\mathbb{I}\} \\ \vdash \langle \mathrm{post}_i(\mathbb{I}) \land \neg c_i\rangle\ S'_i \circledast \chi\ \langle\Psi\rangle \quad (\forall i \in [1, n])\end{array}}{\vdash \langle\Phi\rangle\ [(c_1, S_1, S'_1), \ldots, (c_n, S_n, S'_n)]^* \circledast \chi\ \langle\Psi\rangle}$$

Cartesian Hoare Logic (CHL)

- Our PLDI'16 paper introduces **Cartesian Hoare Logic (CHL)** for verifying k-safety
- CHL proof rules allow synchronization between loops
- Easy to automate reasoning in CHL

$$\text{(Expand)} \quad \frac{\vdash \langle \Phi \rangle \boxed{\circledast} S^n \langle \Psi \rangle}{\vdash \| \Phi \| \; S \; \| \Psi \|}$$

$$\text{(Lift)} \quad \frac{\vdash \{\Phi\} \; S \; \{\Psi\}}{\vdash \langle \Phi \rangle \; S \; \langle \Psi \rangle}$$

$$(\flat\text{−intro 1}) \quad \frac{\vdash \langle \Phi \rangle \; S; \flat \circledast \chi \; \langle \Psi \rangle}{\vdash \langle \Phi \rangle \; S \circledast \chi \; \langle \Psi \rangle}$$

$$(\flat\text{−intro 2}) \quad \frac{\vdash \langle \Phi \rangle \; \chi \circledast \flat \; \langle \Psi \rangle}{\vdash \langle \Phi \rangle \; \chi \; \langle \Psi \rangle}$$

$$(\flat\text{−elim}) \quad \frac{\vdash \langle \Phi \rangle \; \chi \; \langle \Psi \rangle}{\vdash \langle \Phi \rangle \; \chi \circledast \flat \; \langle \Psi \rangle}$$

$$\text{(Assoc)} \quad \frac{\vdash \langle \Phi \rangle \; \chi_1 \circledast (\chi_2 \circledast \chi_3) \; \langle \Psi \rangle}{\vdash \langle \Phi \rangle \; (\chi_1 \circledast \chi_2) \circledast \chi_3 \; \langle \Psi \rangle}$$

$$\text{(Comm)} \quad \frac{\vdash \langle \Phi \rangle \; \chi_2 \circledast \chi_1 \; \langle \Psi \rangle}{\vdash \langle \Phi \rangle \; \chi_1 \circledast \chi_2 \; \langle \Psi \rangle}$$

$$\text{(Step)} \quad \frac{\vdash \{\Phi\} \; S_1 \; \{\Phi'\} \quad \vdash \langle \Phi' \rangle \; S_2 \circledast \chi \; \langle \Psi \rangle}{\vdash \langle \Phi \rangle \; S_1; S_2 \circledast \chi \; \langle \Psi \rangle}$$

$$\text{(Havoc)} \quad \frac{V = \text{vars}(\chi)}{\vdash \langle \Phi \rangle \; \chi \; \langle \exists V. \Phi \rangle}$$

$$\text{(Consq)} \quad \frac{\Phi \Rightarrow \Phi' \quad \Psi' \Rightarrow \Psi \quad \vdash \langle \Phi' \rangle \; \chi \; \langle \Psi' \rangle}{\vdash \langle \Phi \rangle \; \chi \; \langle \Psi \rangle}$$

$$\text{(Seq)} \quad \frac{\begin{array}{c} \vdash \langle \Phi \rangle \; (S_1 \circledast \ldots \circledast S_n) \; \langle \Phi' \rangle \\ \vdash \langle \Phi' \rangle \; (R_1 \circledast \ldots \circledast R_n) \circledast \chi \; \langle \Psi \rangle \end{array}}{\vdash \langle \Phi \rangle \; (S_1; R_1 \circledast \ldots \circledast S_n; R_n) \circledast \chi \; \langle \Psi \rangle}$$

$$\text{(If)} \quad \frac{\begin{array}{c} \vdash \langle \Phi \land c \rangle \; S_1; S \circledast \chi \; \langle \Psi \rangle \\ \vdash \langle \Phi \land \neg c \rangle \; S_2; S \circledast \chi \; \langle \Psi \rangle \end{array}}{\vdash \langle \Phi \rangle \; (S_1 \oplus_c S_2; S) \circledast \chi \; \langle \Psi \rangle}$$

$$\text{(Loop)} \quad \frac{\begin{array}{c} \Phi \Rightarrow \mathbb{I} \\ \vdash \{\mathbb{I}\} \; [(c_1, S_1, \flat), \ldots, (c_n, S_n, \flat)]^* \; \{\mathbb{I}\} \\ \vdash \langle \text{post}_i(\mathbb{I}) \land \neg c_i \rangle \; S_i' \circledast \chi \; \langle \Psi \rangle \quad (\forall i \in [1, n]) \end{array}}{\vdash \langle \Phi \rangle \; [(c_1, S_1, S_1'), \ldots, (c_n, S_n, S_n')]^* \circledast \chi \; \langle \Psi \rangle}$$

Cartesian Hoare Logic (CHL)

- Our PLDI'16 paper introduces **Cartesian Hoare Logic (CHL)** for verifying k-safety
- CHL proof rules allow synchronization between loops
- Easy to automate reasoning in CHL
- More scalable and precise than prior techniques

$$\text{(Expand)} \qquad \frac{\vdash \langle\Phi\rangle \boxast S^n \langle\Psi\rangle}{\vdash \|\Phi\|\ S\ \|\Psi\|}$$

$$\text{(Lift)} \qquad \frac{\vdash \{\Phi\}\ S\ \{\Psi\}}{\vdash \langle\Phi\rangle\ S\ \langle\Psi\rangle}$$

$$(\flat\text{−intro 1}) \qquad \frac{\vdash \langle\Phi\rangle\ S; \flat \circledast \chi\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ S \circledast \chi\ \langle\Psi\rangle}$$

$$(\flat\text{−intro 2}) \qquad \frac{\vdash \langle\Phi\rangle\ \chi \circledast \flat\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ \chi\ \langle\Psi\rangle}$$

$$(\flat\text{−elim}) \qquad \frac{\vdash \langle\Phi\rangle\ \chi\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ \chi \circledast \flat\ \langle\Psi\rangle}$$

$$\text{(Assoc)} \qquad \frac{\vdash \langle\Phi\rangle\ \chi_1 \circledast (\chi_2 \circledast \chi_3)\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ (\chi_1 \circledast \chi_2) \circledast \chi_3\ \langle\Psi\rangle}$$

$$\text{(Comm)} \qquad \frac{\vdash \langle\Phi\rangle\ \chi_2 \circledast \chi_1\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ \chi_1 \circledast \chi_2\ \langle\Psi\rangle}$$

$$\text{(Step)} \qquad \frac{\vdash \{\Phi\}\ S_1\ \{\Phi'\} \quad \vdash \langle\Phi'\rangle\ S_2 \circledast \chi\ \langle\Psi\rangle}{\vdash \langle\Phi\rangle\ S_1; S_2 \circledast \chi\ \langle\Psi\rangle}$$

$$\text{(Havoc)} \qquad \frac{V = \mathrm{vars}(\chi)}{\vdash \langle\Phi\rangle\ \chi\ \langle\exists V.\Phi\rangle}$$

$$\text{(Consq)} \qquad \frac{\Phi \Rightarrow \Phi' \quad \Psi' \Rightarrow \Psi \quad \vdash \langle\Phi'\rangle\ \chi\ \langle\Psi'\rangle}{\vdash \langle\Phi\rangle\ \chi\ \langle\Psi\rangle}$$

$$\text{(Seq)} \qquad \frac{\begin{array}{c}\vdash \langle\Phi\rangle\ (S_1 \circledast \ldots \circledast S_n)\ \langle\Phi'\rangle \\ \vdash \langle\Phi'\rangle\ (R_1 \circledast \ldots \circledast R_n) \circledast \chi\ \langle\Psi\rangle\end{array}}{\vdash \langle\Phi\rangle\ (S_1; R_1 \circledast \ldots \circledast S_n; R_n) \circledast \chi\ \langle\Psi\rangle}$$

$$\text{(If)} \qquad \frac{\begin{array}{c}\vdash \langle\Phi \wedge c\rangle\ S_1; S \circledast \chi\ \langle\Psi\rangle \\ \vdash \langle\Phi \wedge \neg c\rangle\ S_2; S \circledast \chi\ \langle\Psi\rangle\end{array}}{\vdash \langle\Phi\rangle\ (S_1 \oplus_c S_2; S) \circledast \chi\ \langle\Psi\rangle}$$

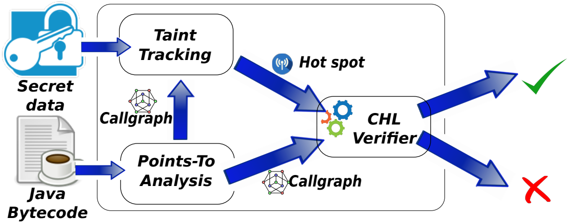
$$\text{(Loop)} \qquad \frac{\begin{array}{c}\Phi \Rightarrow \mathbb{I} \\ \vdash \{\mathbb{I}\}\ [(c_1, S_1, \flat), \ldots, (c_n, S_n, \flat)]^*\ \{\mathbb{I}\} \\ \vdash \langle \mathrm{post}_i(\mathbb{I}) \wedge \neg c_i\rangle\ S'_i \circledast \chi\ \langle\Psi\rangle \quad (\forall i \in [1, n])\end{array}}{\vdash \langle\Phi\rangle\ [(c_1, S_1, S'_1), \ldots, (c_n, S_n, S'_n)]^* \circledast \chi\ \langle\Psi\rangle}$$

Verifying Bounded Non-interference

- Our tool for verifying bounded non-interference performs reasoning in CHL

Verifying Bounded Non-interference

- Our tool for verifying bounded non-interference performs reasoning in CHL
- However, combines CHL verifier with taint analysis for better scalability



Results

- Using this technique, we solved several benchmarks from engagements (e.g., GabFeed, SnapBuddy, TourPlanner)
- Furthermore, we were able to find vulnerabilities in real Java applications, such as Tomcat, Jetty, JBoss, and SpringSecurity
- In addition, we also used technique to analyze other k-safety properties (e.g. transitivity, associativity, symmetry etc.)

Demo Time!