Security and Reliability of the Internet Of Things (IoT): A Smart Meter Case Study

Karthik Pattabiraman

Farid Molazem Tabrizi, Maryam Raiyat, Abraham Chan, Ivan Beschastnikh

University of British Columbia (UBC)

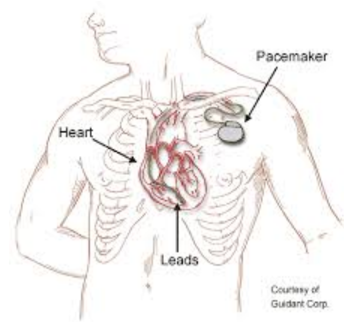
My Research

- **Building fault-tolerant and secure software systems**

- **Application-level fault and attack tolerance**
  - Software resilience techniques [SC’16][DSN’16][DSN’15][DSN’14A][DSN14B]
  - Web applications’ reliability [ICSE’16][ICSE’15][ICSE’14A][ICSE’14B]
  - IoT Security [ACSAC’16][EDCC’15][HASE’14]

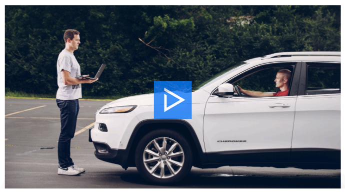
- **This talk**
  - IoT Security and Reliability: Smart Meter Case Study

IoT Systems are Everywhere

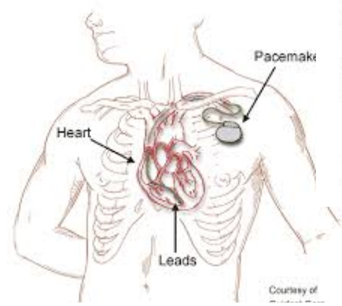


IoT Security and Reliability

HACKERS REMOTELY KILL A JEEP ON THE HIGHWAY—WITH ME IN IT




Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses



Smart meters can be hacked to cut power bills

By Mark Ward
Technology correspondent, BBC News

16 October 2014 | Technology

Share

Smart meters could help people do a better job of managing power use

Smart meters widely used in Spain can be hacked to under-report energy use, security researchers have found.

Nest Thermostat Glitch Leaves Users in the Cold

Disruptions
By NICK BILTON JAN. 13, 2016

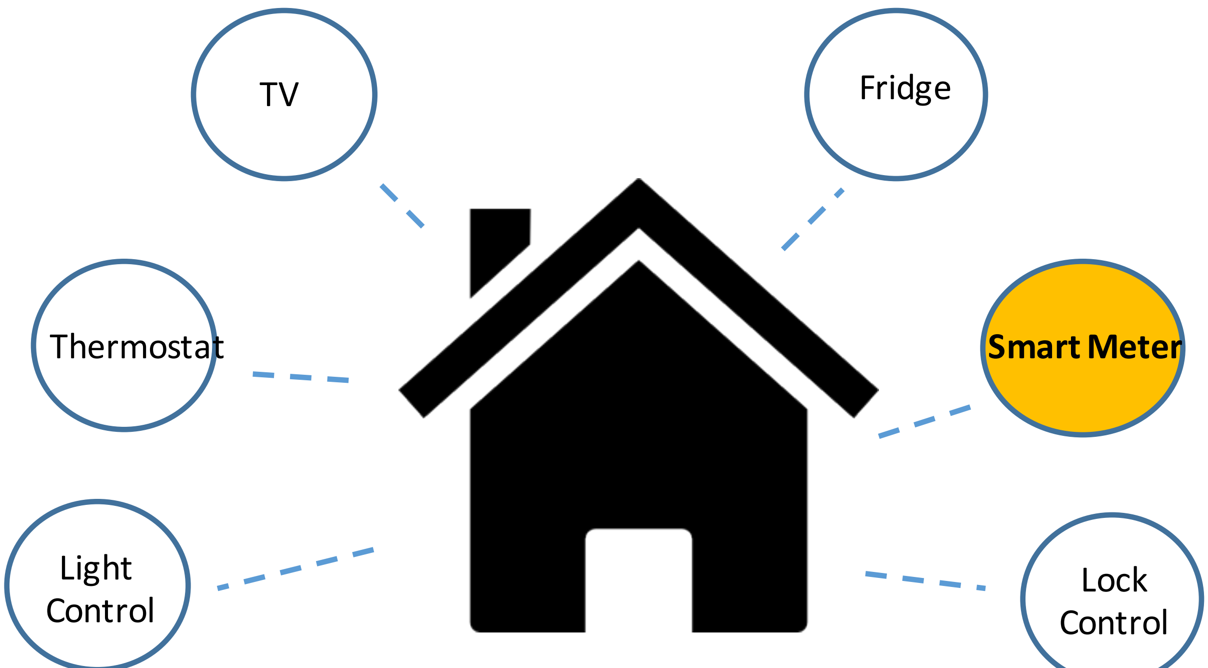
The Nest Learning Thermostat is dead to me, literally. Last week, my once-beloved "smart" thermostat suffered from a mysterious software bug that drained its battery and sent our home into a chill in the middle of the night.

Although I had set the thermostat to 70 degrees overnight, my wife and I were woken by a crying baby at 4 a.m. The thermometer in his room read 64 degrees, and the Nest

IoT Security and Reliability: Challenges

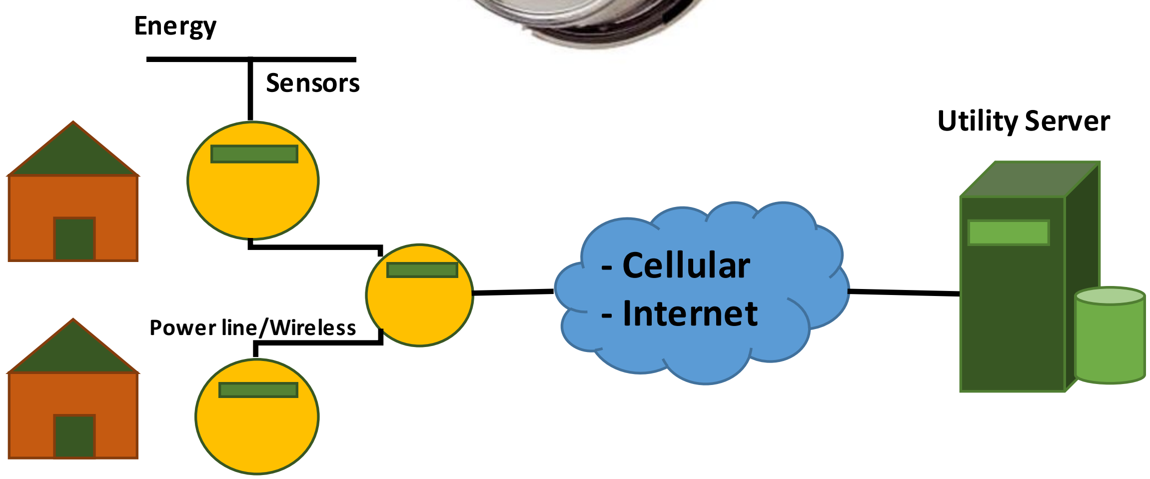
- **IoT devices are resource constrained**
  - Low memory and computing capacity
  - Sometimes energy constrained
- **Large scale of deployment**
  - Worms can spread quickly in the network
  - Need scalable solutions with low false positives
- **Autonomous operation**
  - Need for human intervention should be minimal or none
  - Must be capable of operating continuously for a long time

IoT Example: Smart Meters

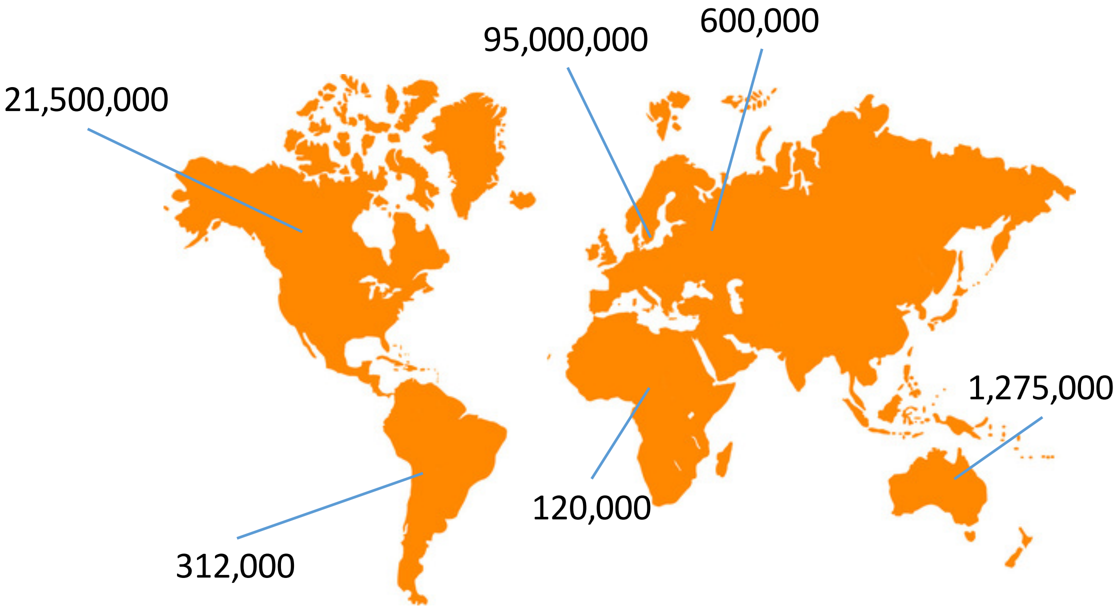


Smart Meter


123456
iCON
20 000 004


Energy
Sensors
Utility Server
- Cellular
- Internet
Power line/Wireless

Global Status of Smart Meters



2009: 76 million

2010: 118 million

2012: 1 billion

Smart Meter Security

- **Smart meter Attacks**
  - No need for physical presence
  - Hard to detect by inspection or testing
  - Attacks can be large-scale

Analog Meter

Smart Meter

Smart Meter Security is a concern

USA TODAY

NEWS SPORTS LIFE MONEY TECH TRAVEL OPINION 83° CROSSWORDS YOUR TAKE INVESTIGATIONS

Search

2015 could be year of first smart-home hacks

John Shinal, Special for USA TODAY 6:47 p.m. EST December 30, 2014

204

FBI frets about dumb security in smart meters

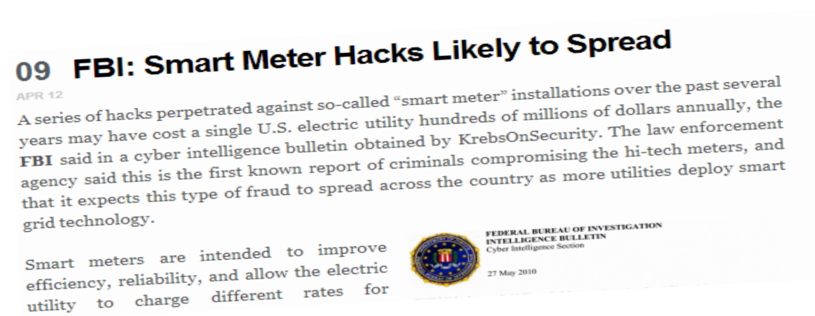
Lax security costs utilities plenty

By **Richard Chirgwin • Get more from this author**

Posted in Security, 9th April 2012 23:00 GMT

Free whitepaper – EMA advanced performance analytics report

The FBI is seeing increasing hacks on electricity smart meters, with most attac
consumers get power without paying for it.



09 FBI: Smart Meter Hacks Likely to Spread

APR 12

A series of hacks perpetrated against so-called "smart meter" installations over the past several years may have cost a single U.S. electric utility hundreds of millions of dollars annually, the **FBI** said in a cyber intelligence bulletin obtained by KrebsOnSecurity. The law enforcement agency said this is the first known report of criminals compromising the hi-tech meters, and that it expects this type of fraud to spread across the country as more utilities deploy smart grid technology.

Smart meters are intended to improve efficiency, reliability, and allow the electric utility to charge different rates for

FEDERAL BUREAU OF INVESTIGATION
INTELLIGENCE BULLETIN
Cyber Intelligence Section

27 May 2010

Outline

- Motivation and Goals
- Host-based Intrusion Detection System (IDS) for smart meters [EDCC’15 – Distinguished Paper Award][HASE’14]
- Model checking to find design vulnerabilities in smart meters [ACSAC’16]
- Ongoing Work and Conclusions

IDS: Goal

- **Goal:** Make IoT embedded devices secure
  - Build a host-based intrusion detection system

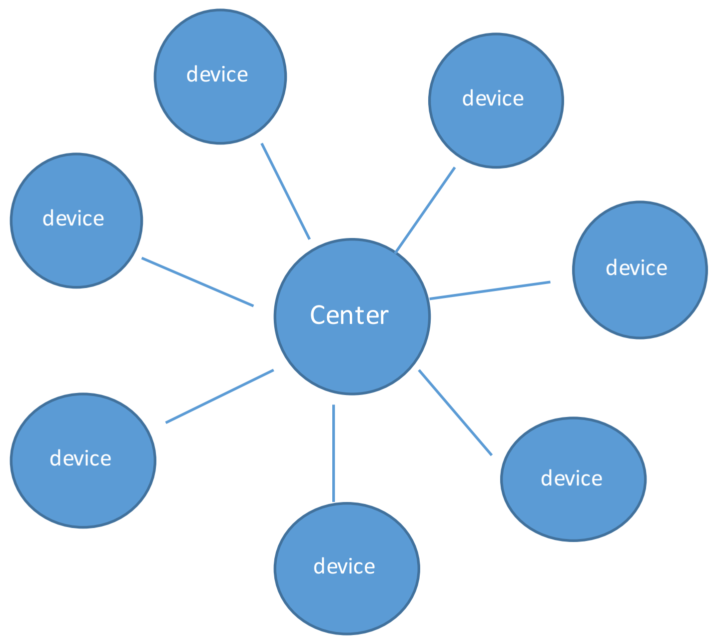
- **Important constraints**

  - Small embedded devices => Low memory capacity

  - Large scale => No false positives

  - Low cost => Automated, no special hardware etc.

IDS Challenge: False Positives

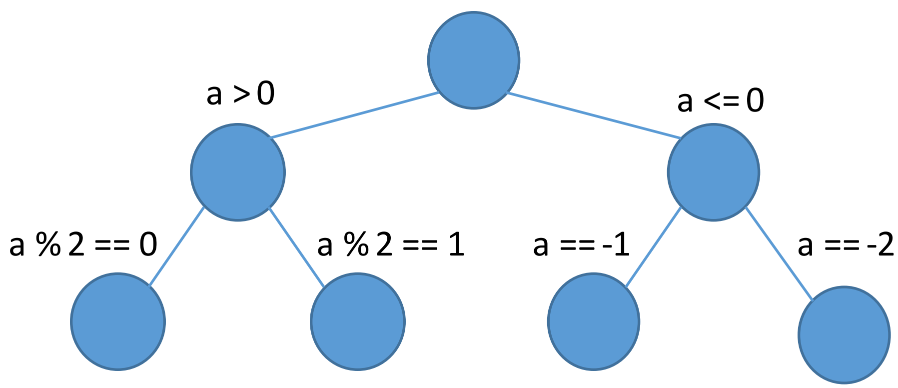


IDS Challenge: Memory Constraints

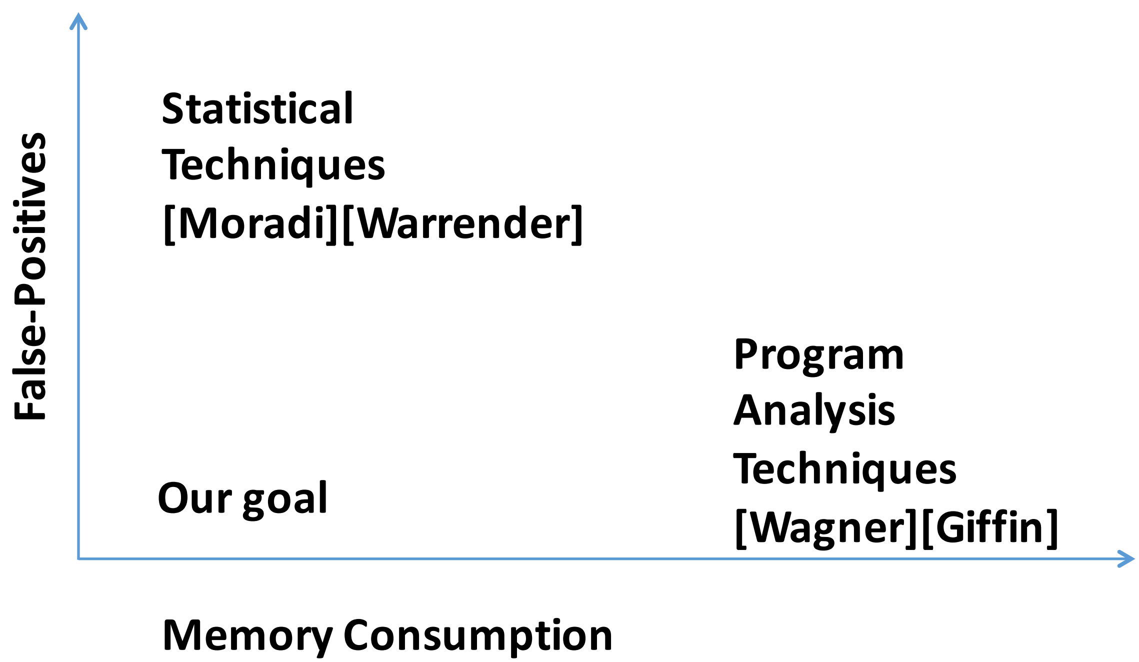
```
{
  a = receive();
  if (a > 0)
    foo(a);
  else
    bar(a);
}
```

```
void foo(int a) {
  if (a % 2 == 0)
    even(a);
  else
    odd(a);
}
```

```
void bar(int a) {
  if (a == -1)
    error1();
  else if (a == -2)
    error2();
}
```

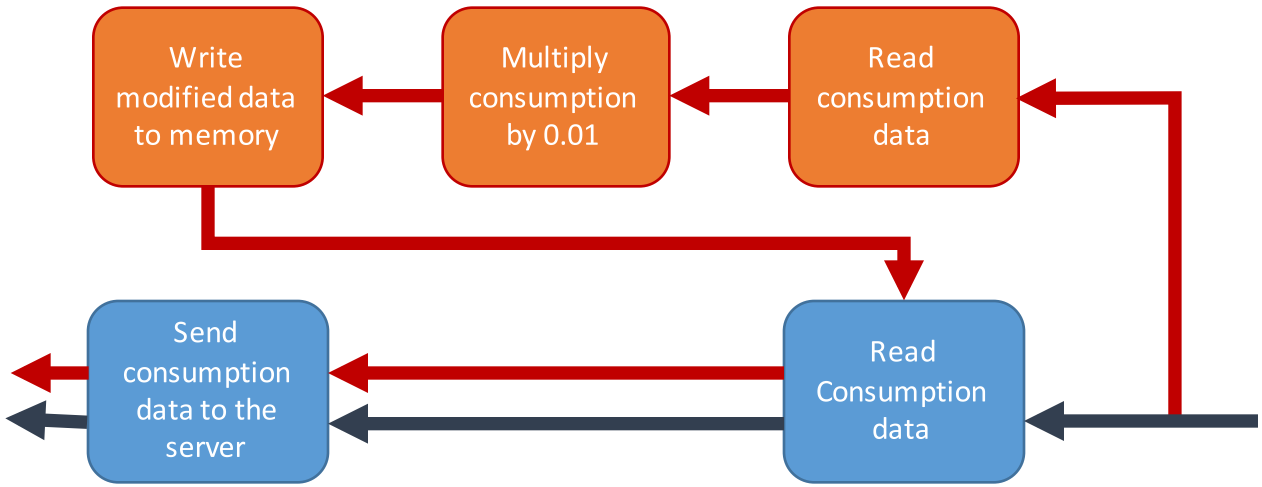


IDS Existing Solutions



IDS Threat model

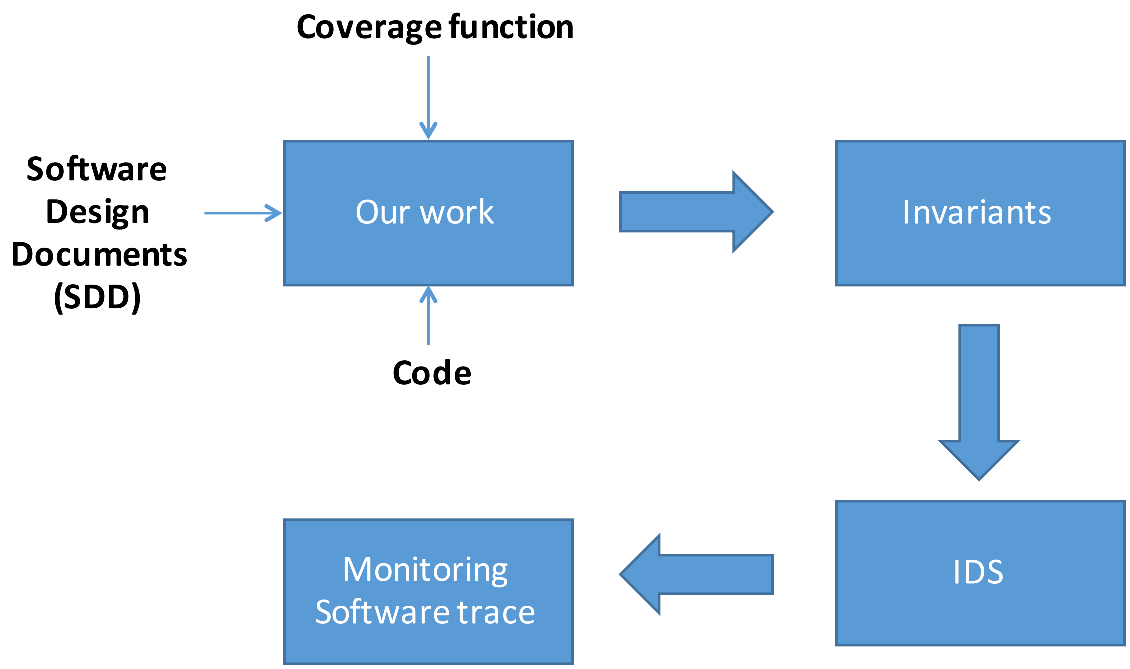
- Adversary: Wants to change the execution of the software (in subtle ways) to avoid detection. Do not consider privacy or confidentiality.



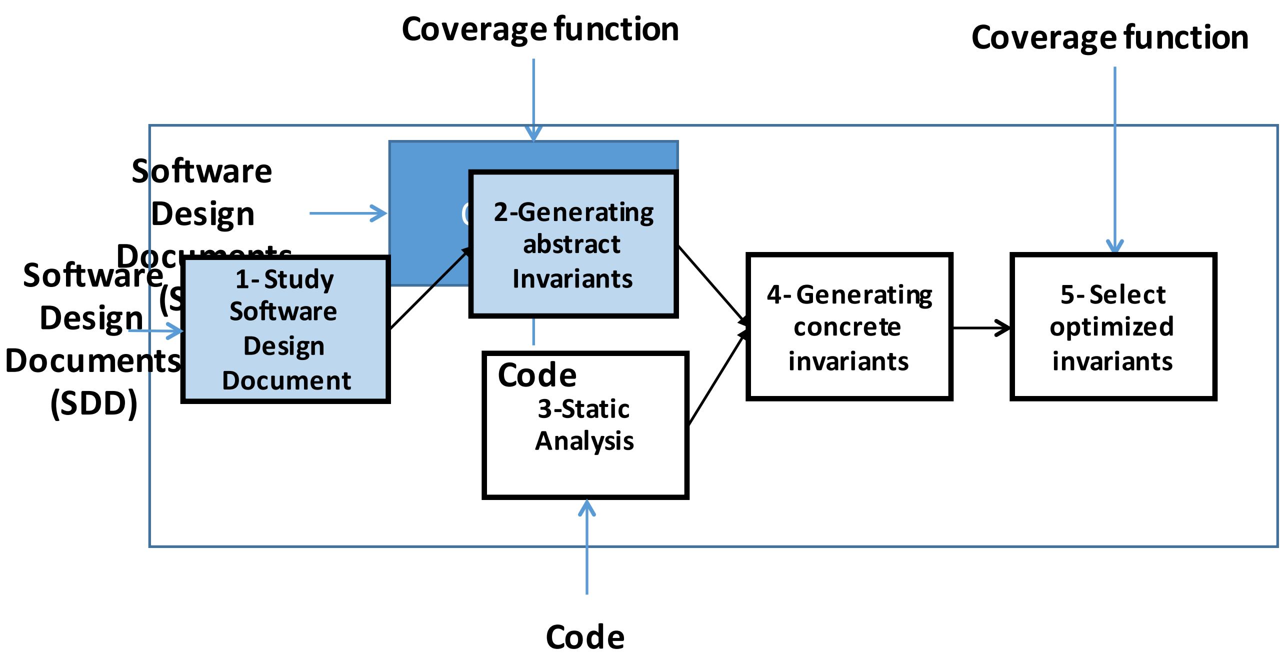
IDS: Main Idea

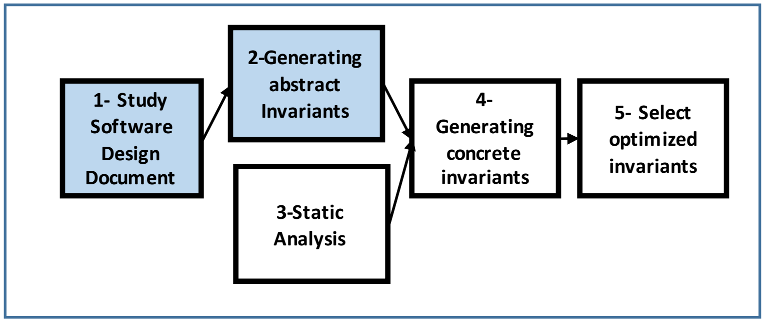
- Quantify security to detect only the most critical attacks, subject to memory constraints

IDS Approach: Overview



IDS Approach: Details

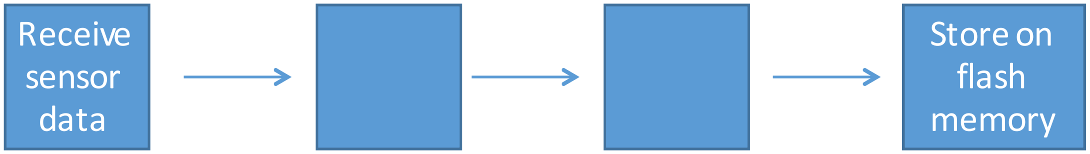




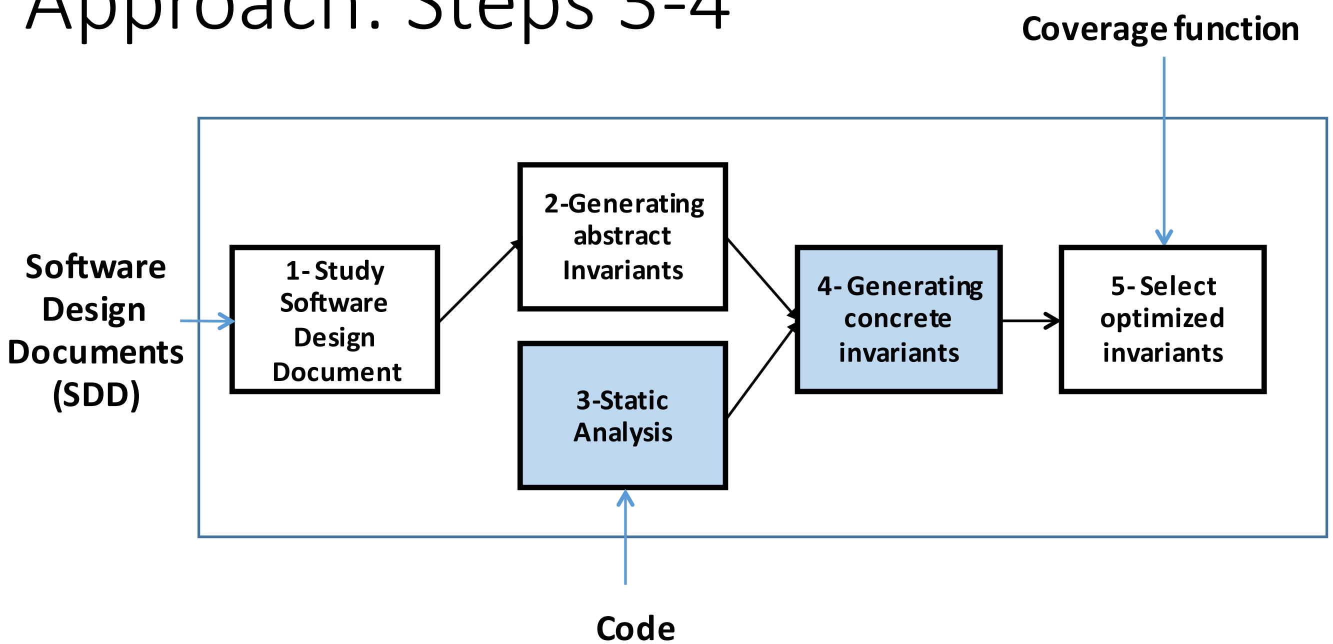
- Storage/Retrieval integrity

Sensor data must eventually be stored on flash memory

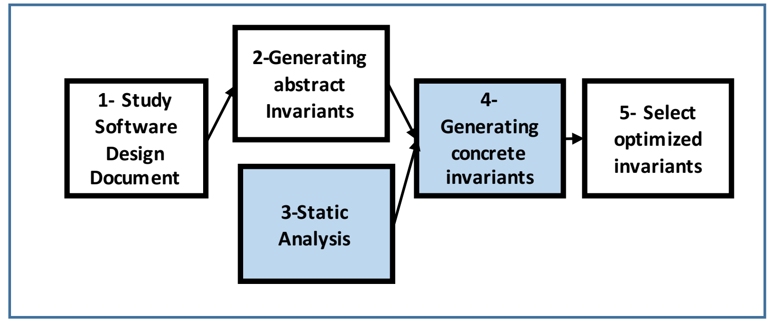
$\square(getting\ sensorData \Longrightarrow (\lozenge\ store\ on\ flash))$



IDS Approach: Steps 3-4



Abstract invariants → Concrete invariants (contain system calls)




```
{
    ....
    data = socket.receive();
    ....
}
```

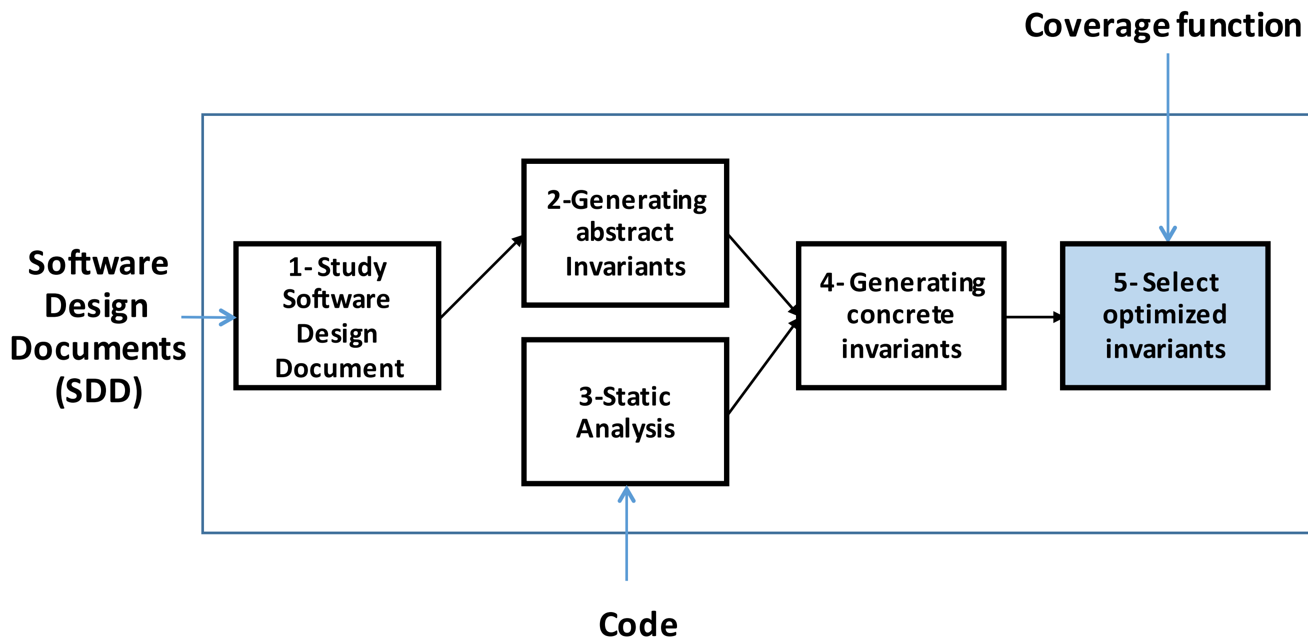
```
{
    ....
    write(f, data);
    ....
}
```

```
...
recv(4, 0x47cf68, 8192, 0)
...
write(1, 0x47cf68, 4) = 4
...
```

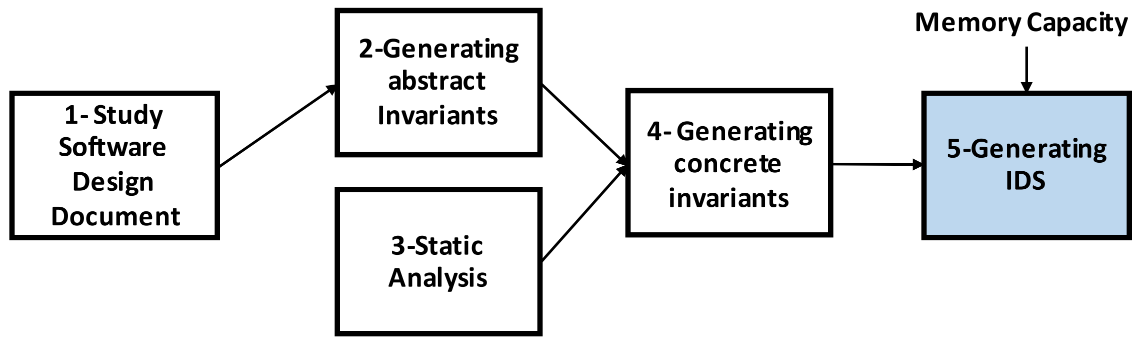
$$\square(getting\ sensorData(data) \Longrightarrow (\lozenge\ store\ on\ flash(data)))$$

$$\square(receive(d) \Longrightarrow (\lozenge\ write(d)))$$

IDS Approach: Step 5



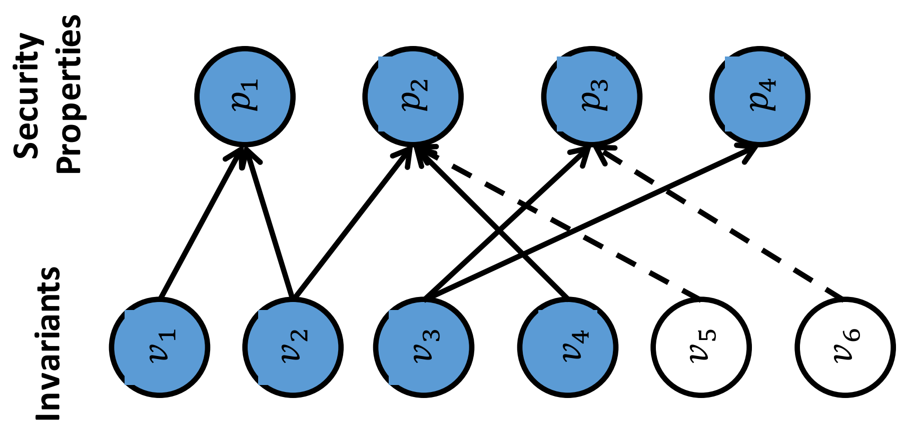
IDS Approach: Building the IDS



Formulate building the IDS as an optimization problem, where we maximize coverage subject to cost constraints

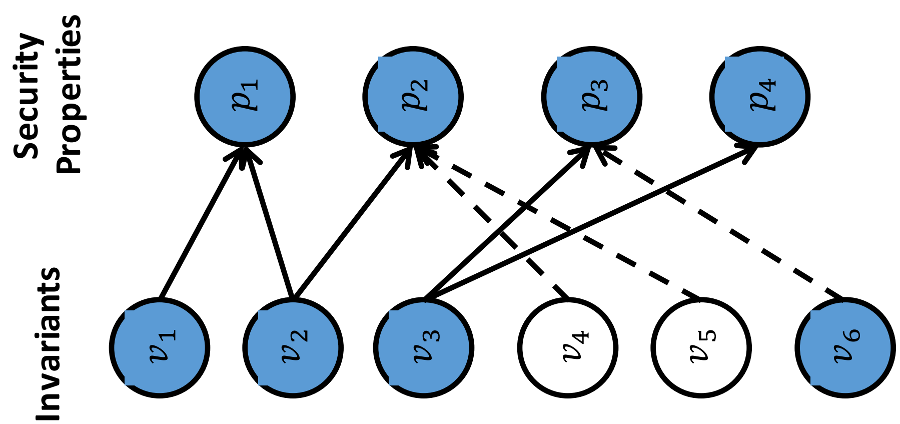
IDS Coverage: MaxMin Coverage

MaxMin Coverage IDS: Maximize minimum coverage
i.e., distribute coverage among all properties



IDS Coverage: MaxProperty IDS

MaxProperty IDS:
Maximize security properties that are fully covered



IDS: Building the IDS

Select the invariants from the graph according to the coverage function

↓

Automatically convert it to Buchi Automaton

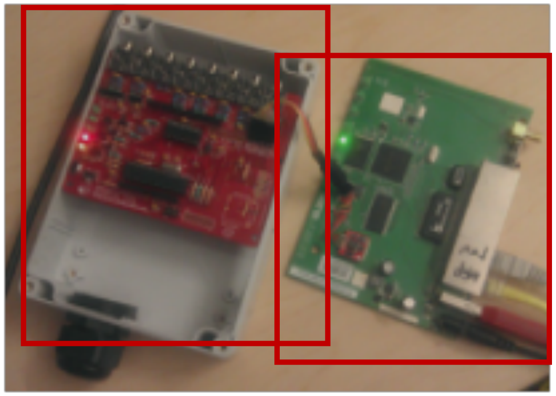
↓

Monitor the invariants at runtime

IDS Evaluation: Testbed

- Testbed: Smart Meter

- Meter:
  - Arduino board
    - ATMEGA 32x series microcontroller
    - Sensors
  - Gateway board
    - Broadcom BCM 3302 240MHz CPU
    - 16 MB RAM
    - **4 MB available for IDS**
    - OpenWRT Linux
  - IDS runs on the Gateway board

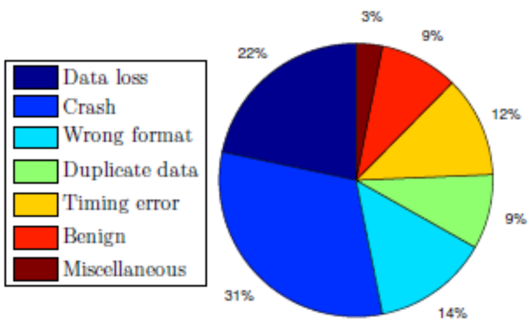


IDS Evaluation: Fault injection

- Flipping branches (surreptiously)

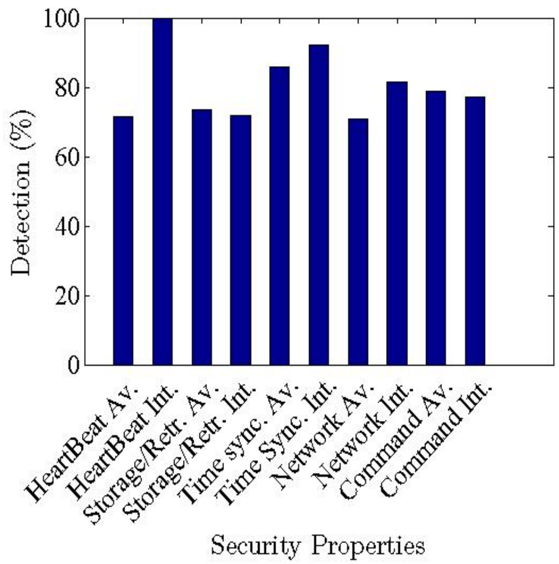
```
if (data_file ~= nil) then
  big_string = data_file:read("*all")
  ...
end
```

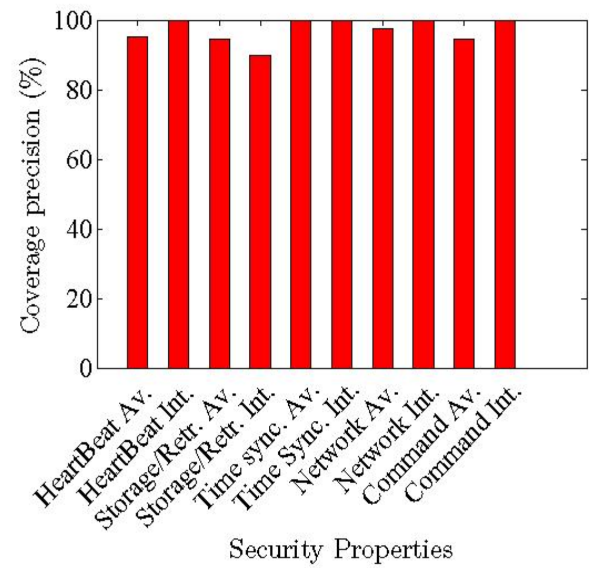
```
if (data_file == nil) then
  big_string = data_file:read("*all")
  ...
end
```



IDS Results (MaxMin IDS: 2 MB memory)

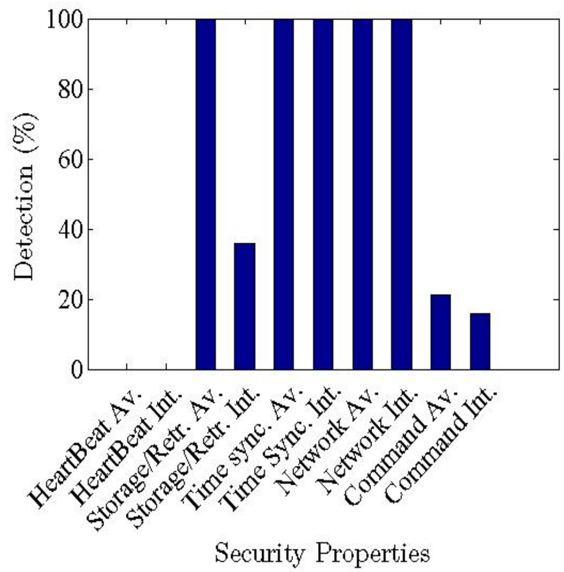
- How good is the coverage of the IDS (left)?
- How good the graph-based optimization is reflected at run-time (right)?

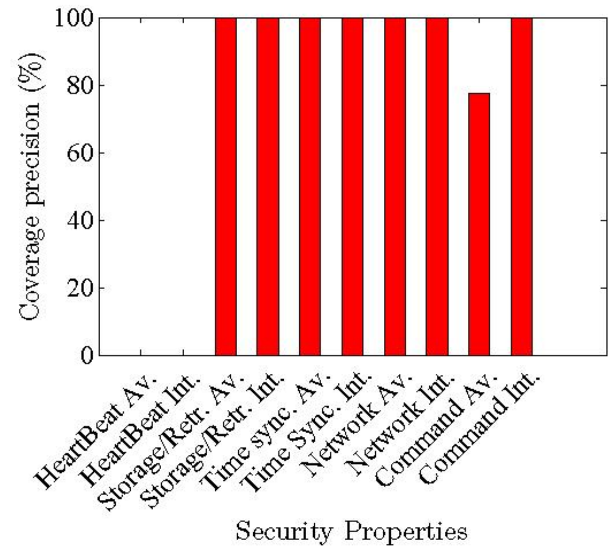




IDS Results (MaxProperty IDS: 2 MB memory)

- How good is the coverage of the IDS (left)?
- How good the graph-based optimization is reflected at run-time (right)?



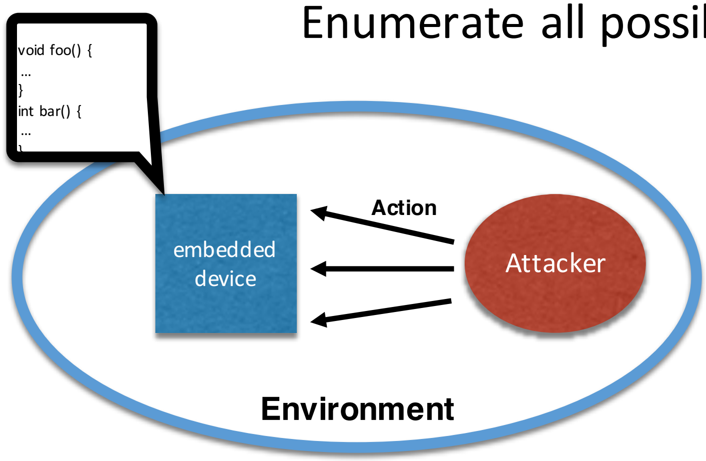


Outline

- Motivation and Goals
- Host-based Intrusion Detection System (IDS) for smart meters [EDCC'15 – Distinguished Paper Award][HASE'14]
- Model checking to find design vulnerabilities in smart meters [ACSAC'16]
- Ongoing Work and Conclusions

Model Checking: Problem

Enumerate all possible attacks

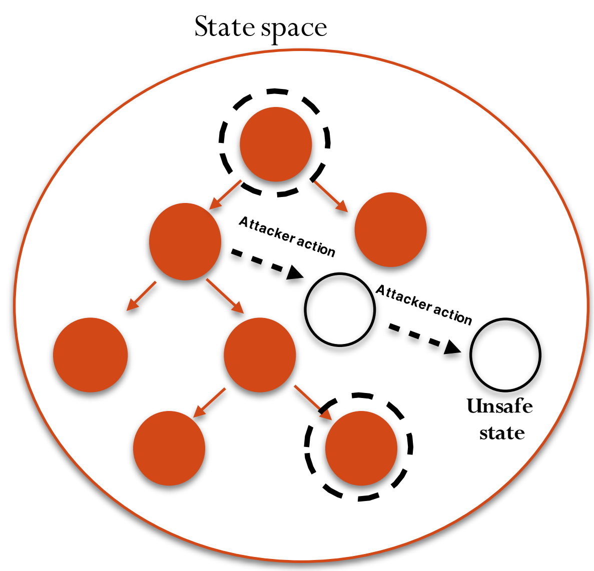


Model Checking: Challenge

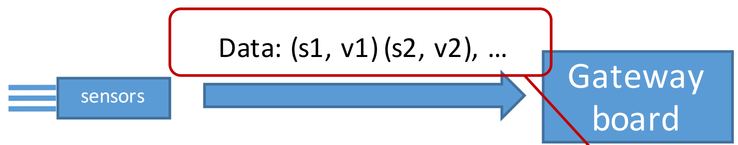
- Formal analysis requires well-defined properties (e.g. TCP/IP)
  - Unclear in IoT devices
- The state space may be very large
  - Require the *right* level of abstraction
    - High-level enough to avoid state space explosion
    - Low-level enough to be translatable to device code

Model Checking: Our approach

- Key Idea: Each class of embedded devices performs similar operations
  - We can abstract the operations
    - Create an abstract model
      - Formalize the model (using Maude)
      - Formalize attacker actions
      - Define unsafe states
      - Run model checking to find attacker actions leading to unsafe states



Model Checking: Formal model



SENSOR-STATES

```
1. mod SENSOR-STATES is
2. op getSensorDataList : —> SensorDataList.
3. var dataList : SensorDataList.
4. var r n : Nat.
5. rl [r1]:  getSensorDataList —> sensorDataElement(0,0).
6. crl [r2]: sensorDataElement(r,n) —> sensorDataElement(r,n)
             sensorDataElement(r+1, 0)   if r < maxSensorNumber.
7. crl [r3]: sensorDataElement(r,n) —>  sensorDataElement(r,n+1)
             if n < maxSensorData.
8. endm
```

Defines the operation of receiving data from sensors SensorDataList is a list of tuples, each called sensorDataElement

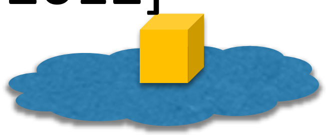
Defines necessary variables for defining the operations

Base of recursion

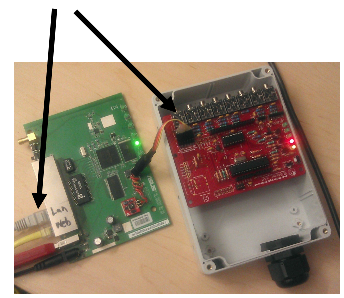
Recursively defining the rule to extend one sensorDataElement, to up to maxSensorNumber elements. Each tuple is: [value, sensor channel number].

Model Checking: Threat model

Root access to a node in grid network [Mo et al. 2012]



- Actions
  - Drop messages
  - Replay messages
  - Reboot meter

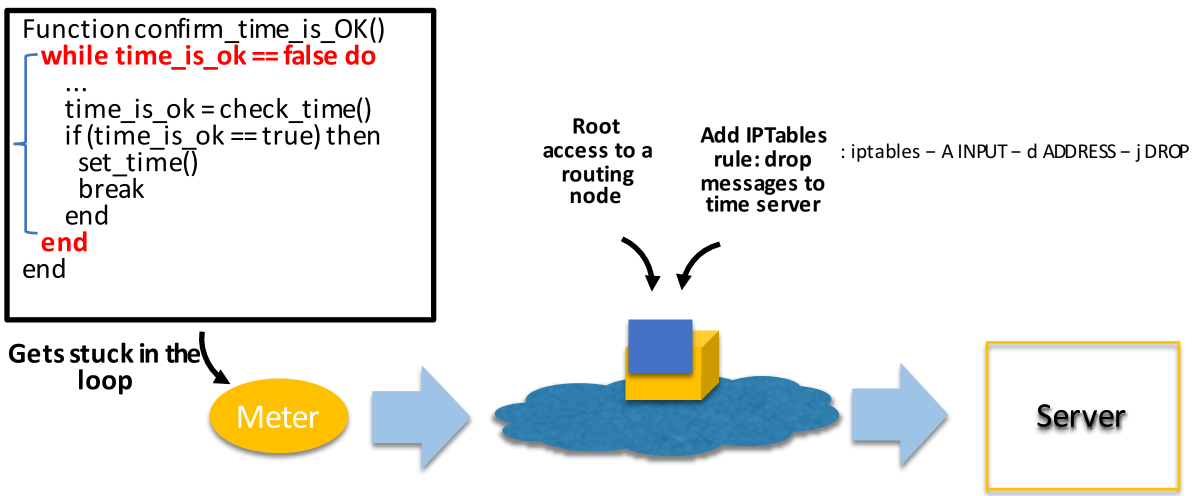


Read/Write access to communication interfaces [McLaughlin et al. 2010]

Model Checking: Results

- For each attacker action: query for paths to unsafe states, e.g.,
  - *search sensor(N1, M1) sensor(N2, M2) sensor(N3, M3)* $\Rightarrow$ *sensor(N1, M1) sensor(N2, M2)*
  - Checks if any data may be lost via dropping messages
  - Found many attacks: Many map to the same execution path

Model Checking: Attacks example 1



Model Checking: Attack example 2

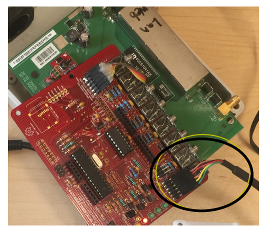
Sensor board

Data

Request

Communication board

Normal behavior



Find serial communication configuration (a handful common configs, a couple of hundreds total configs

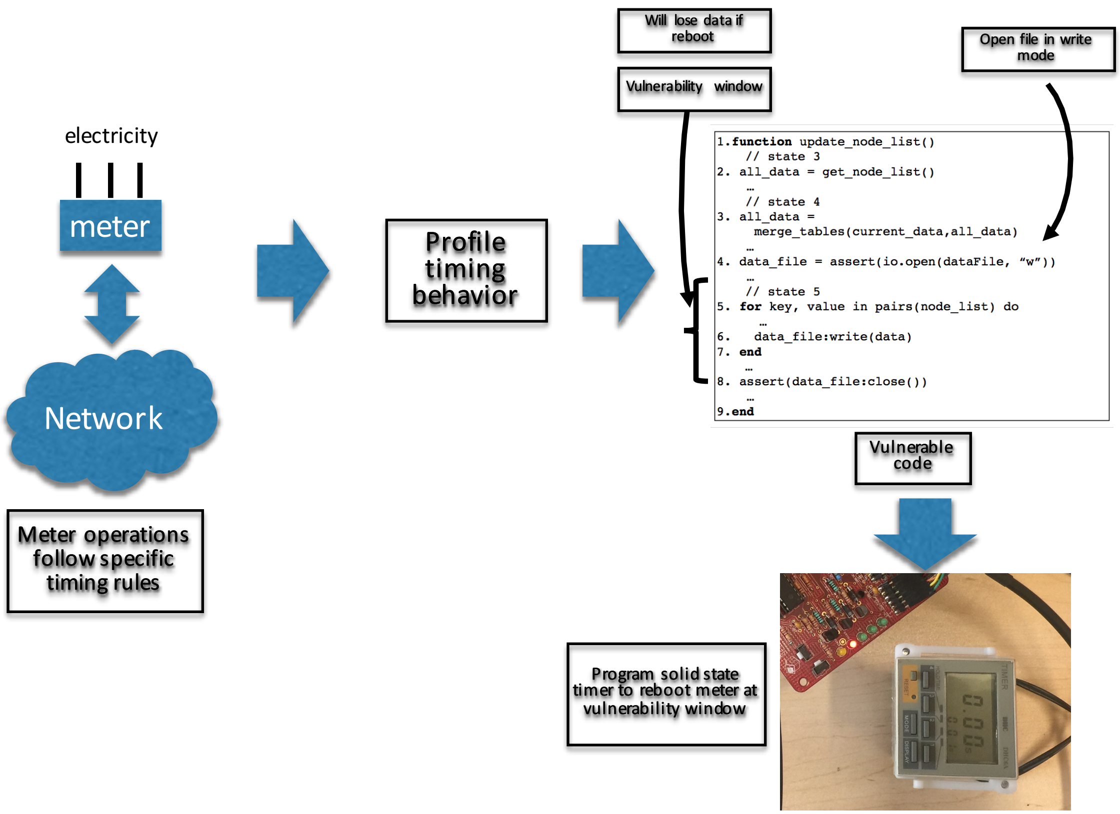
One of the common configs worked in our case

Use USB to 6-pin serial connector from laptop to meter

Replay data request

Receive data on the laptop – data deleted from sensor board

Model Checking: Attack example 3



Model Checking: Performance

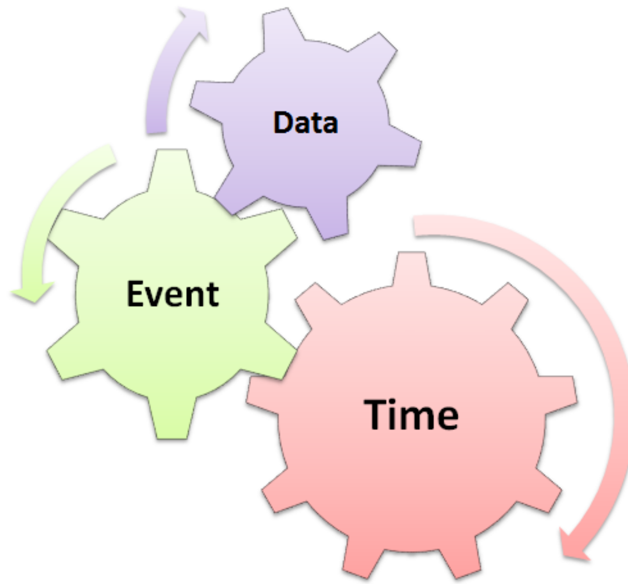
Attacker action	**Time (hrs)**	**Attacks found**
Dropping packets	0.002	12
Replay	0.005	845
System reboot	1.9	6452

Outline

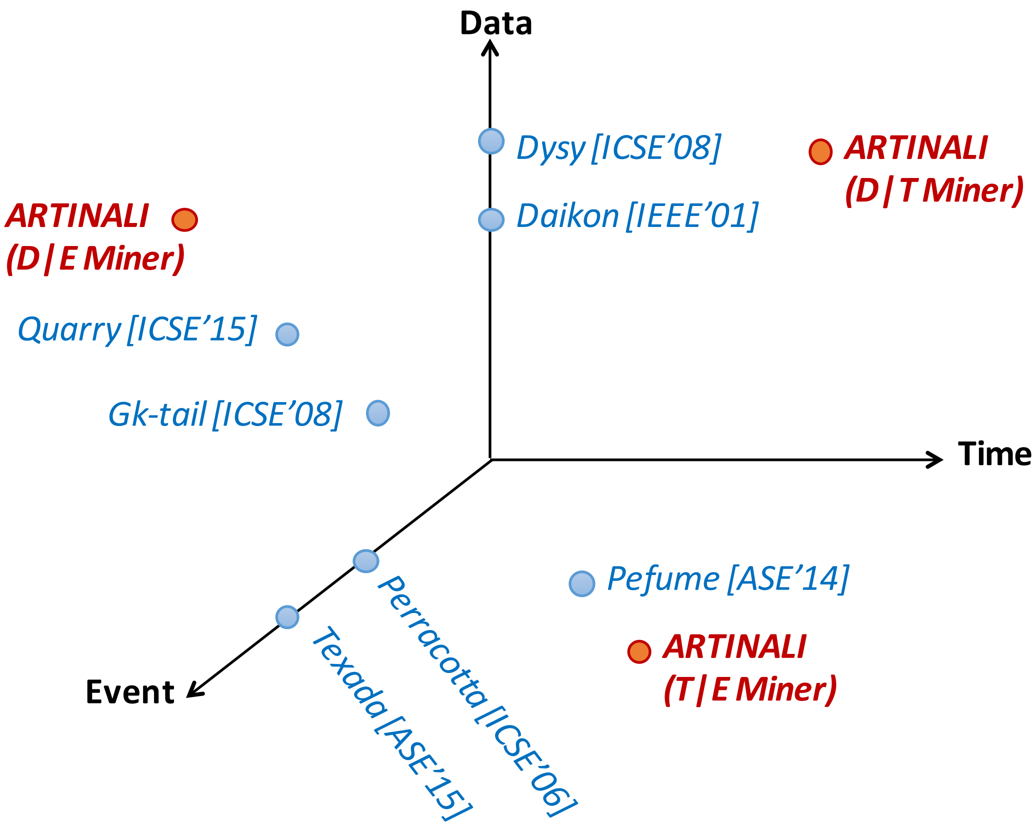
- Motivation and Goals
- Host-based Intrusion Detection System (IDS) for smart meters [EDCC'15 – Distinguished Paper Award][HASE'14]
- Model checking to find design vulnerabilities in smart meters [ACSAC'16]
- Ongoing Work and Conclusions

Invariants: ARTINALI

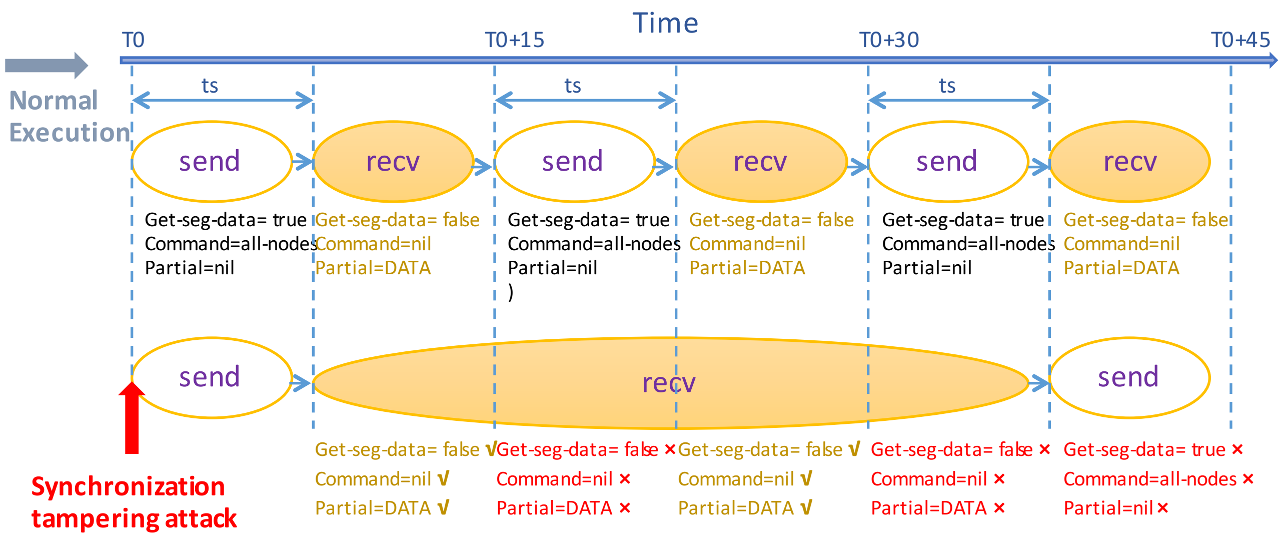
- **A Real-Time-specific Invariant iNference ALgorIthm**
    - Mining independent properties
    - Finding Temporal relationship of independent properties
    - Incorporating time properties into data invariants



Invariants: ARTINALI VS. Previous work



Invariants: Synchronization Tampering Attack

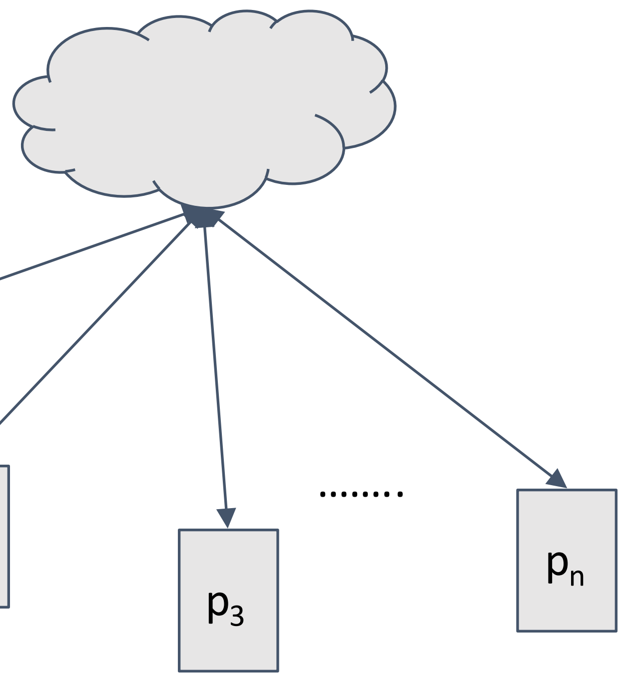


Detection : violation in time per event invariant:

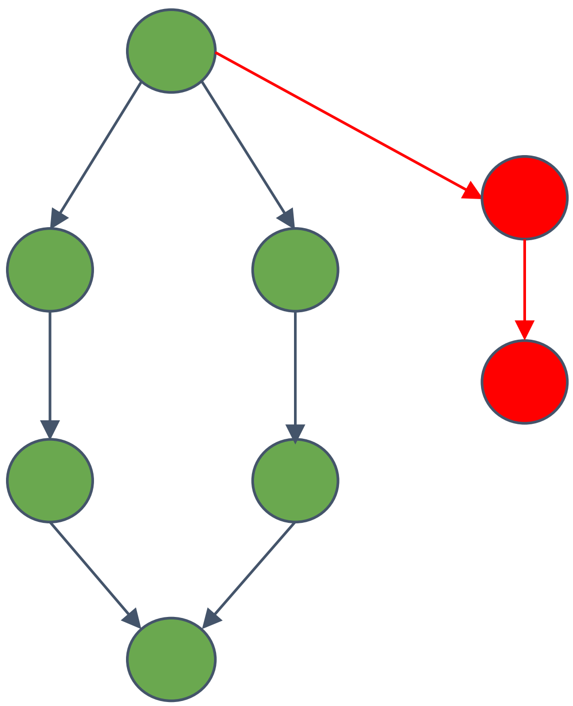
*send (T0 + K*15) → send (T0+(K+1)*15)*

Diversity: Motivation

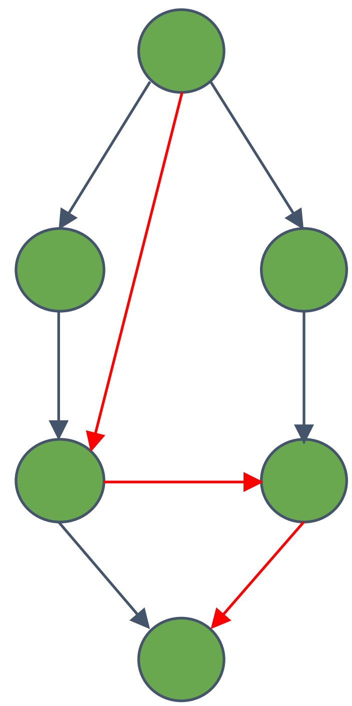
- One compromised device will not lead to attacks on other similar devices



Diversity: Code Reuse Attacks

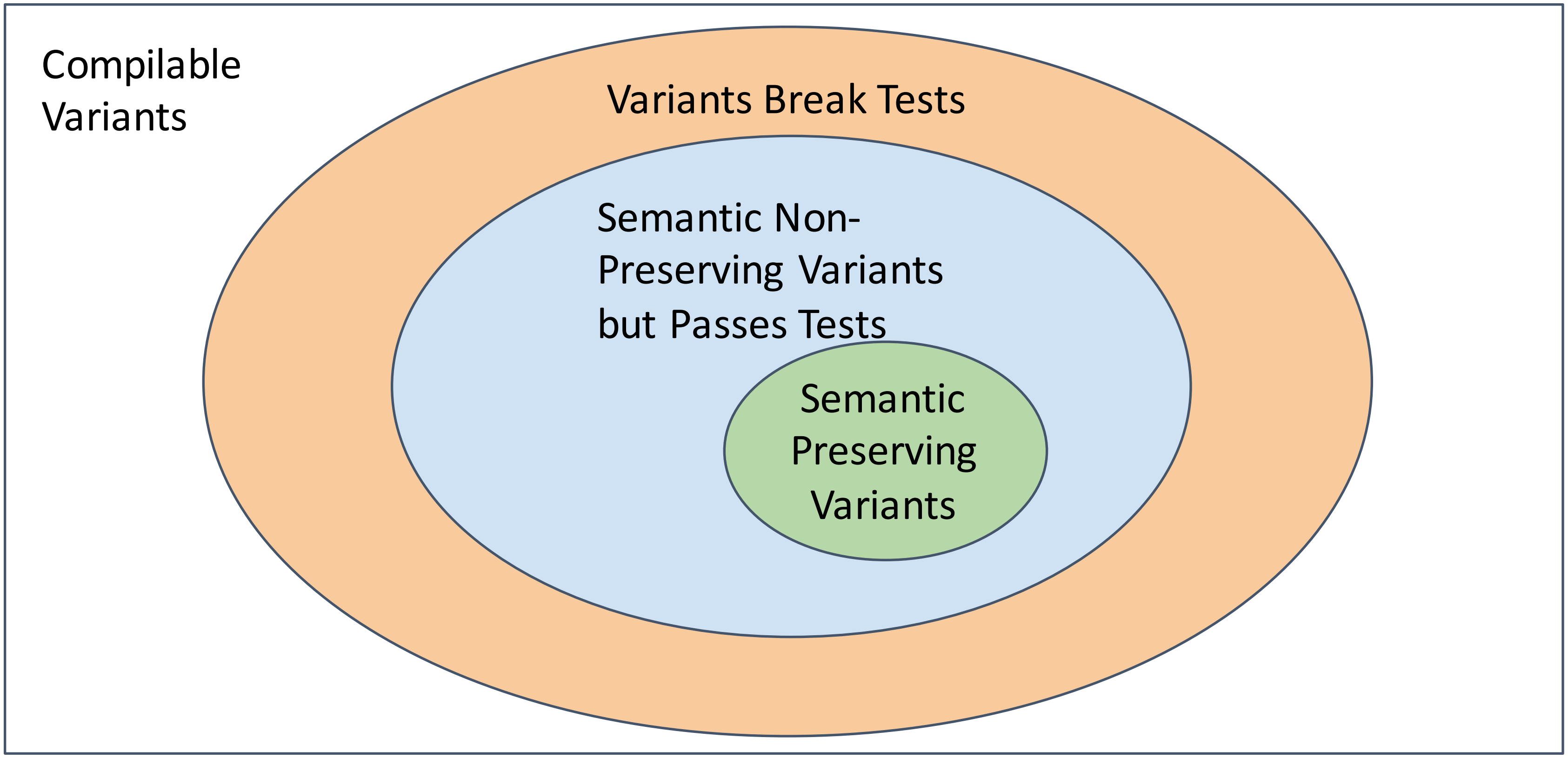


Code Injection Attack



Code Reuse Attack

Diversity: Functional Correctness vs Security?



Conclusions

- **IoT Security and Reliability are important**
  - Challenging due to memory and resource constraints
  - Physical access to the device is possible
- **Smart Meters: Important class of IoT device**
  - Host-Based IDS to detect intrusions
  - Model checking to find design defects
- **Ongoing Work**
  - Extracting invariants for runtime monitoring (ArtiNali)
  - Enhancing diversity among deployed variants (NVerD)

Security and Reliability of the Internet Of Things (IoT): A Smart Meter Case Study

Karthik Pattabiraman

Farid Molazem Tabrizi, Maryam Raiyat, Abraham Chan, Ivan Beschastnikh

University of British Columbia (UBC)