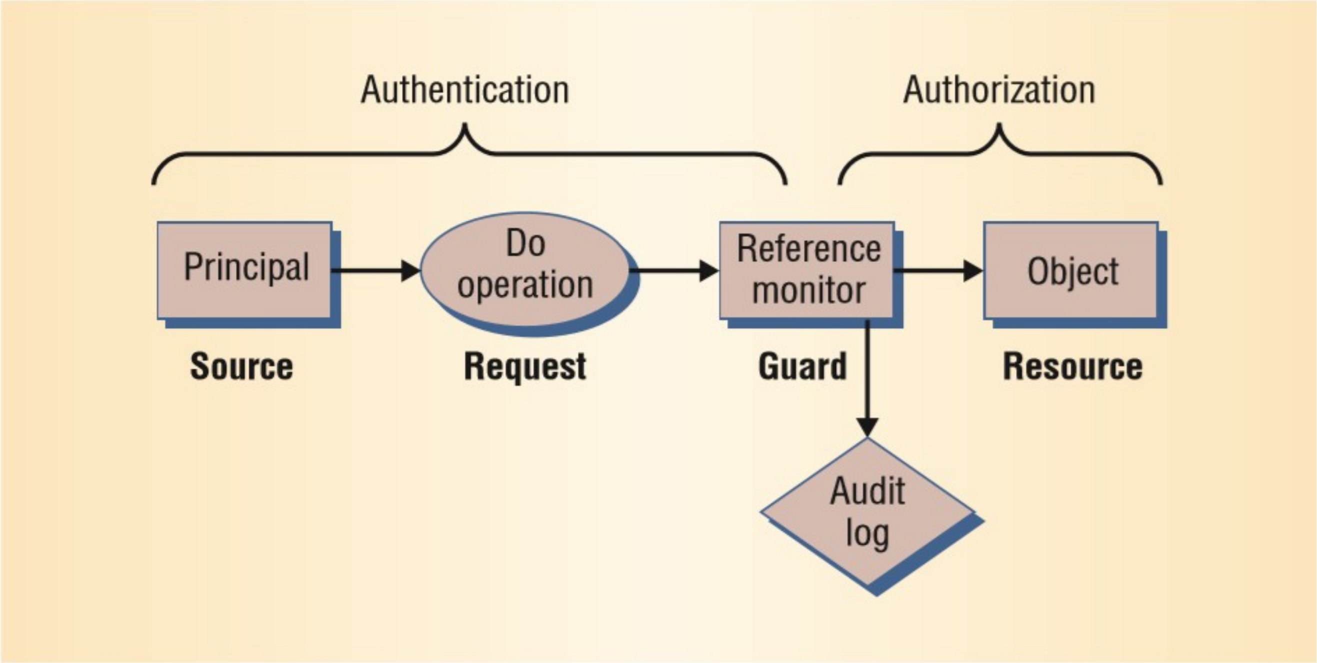
Specifying and Checking Data Use Policies

Sriram Rajamani
Microsoft Research India

Access Control Policies



Core Mechanisms:

authenticating principals—determines who made a request; principals usually are people, but they also can be groups, channels, or programs;

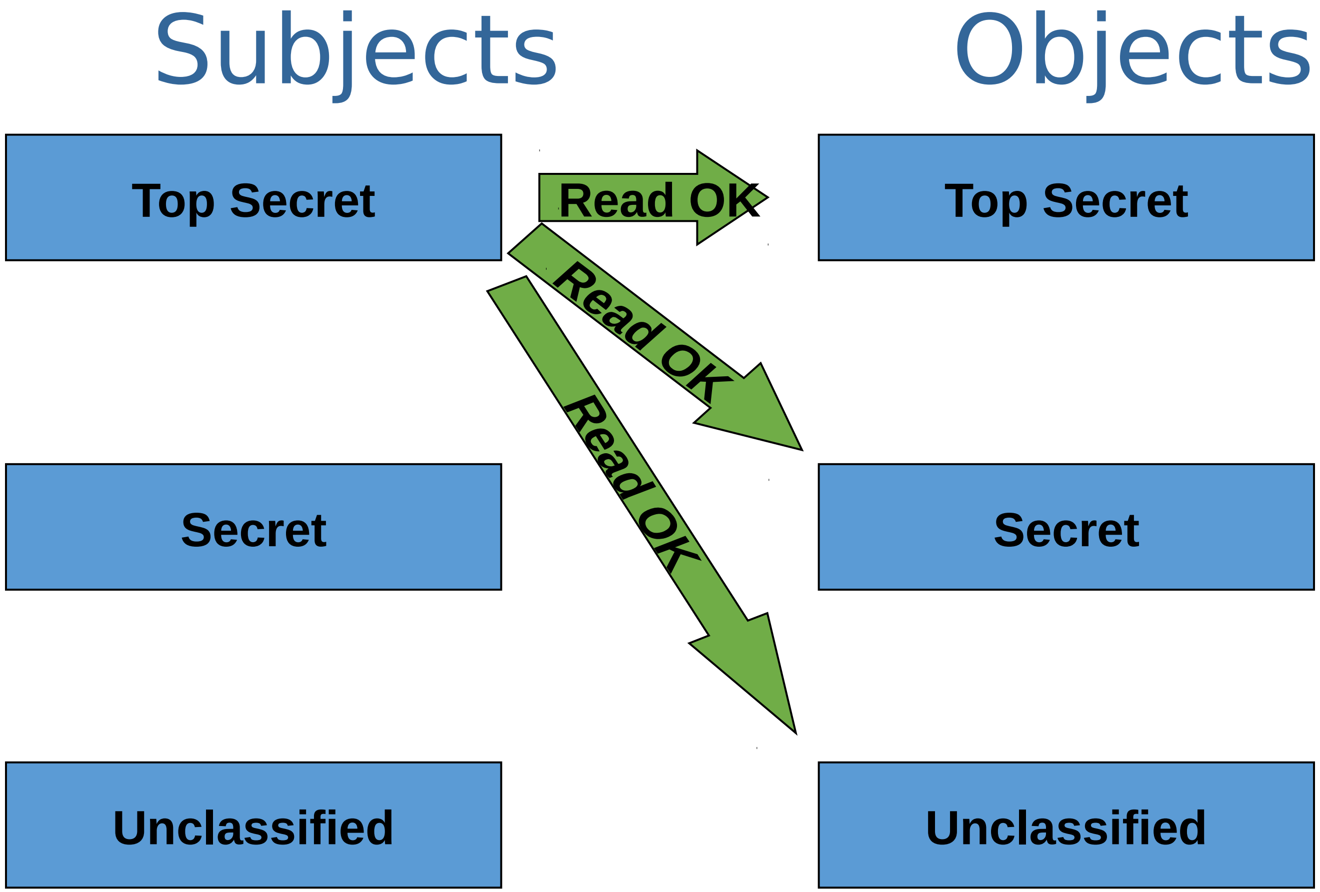
authorizing access—determines who is trusted to do which operations on an object;

auditing the guard's decisions—makes it possible to determine later what happened and why.

Picture credit: "Computer Security in the Real World", B. Lampson 2004

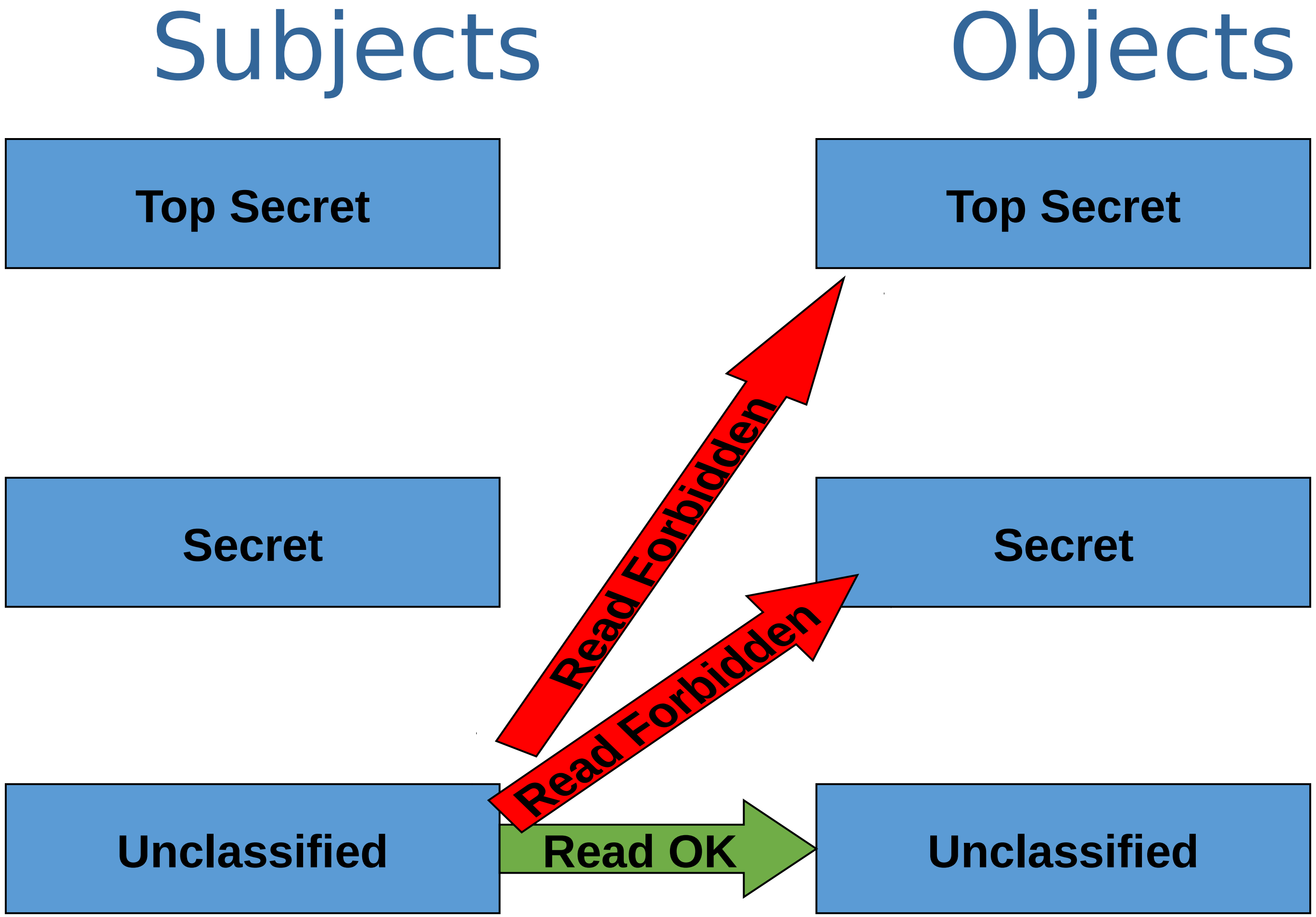
Bell and LaPadula Model

- Each user subject and information object has a fixed security class – labels
- Use the notation ≤ to indicate dominance
- Simple Security (ss) property: the no read-up property
  - A subject s has read access to an object iff the class of the subject C(s) is greater than or equal to the class of the object C(o)
  - i.e. Subjects can read Objects iff C(o) ≤ C(s)



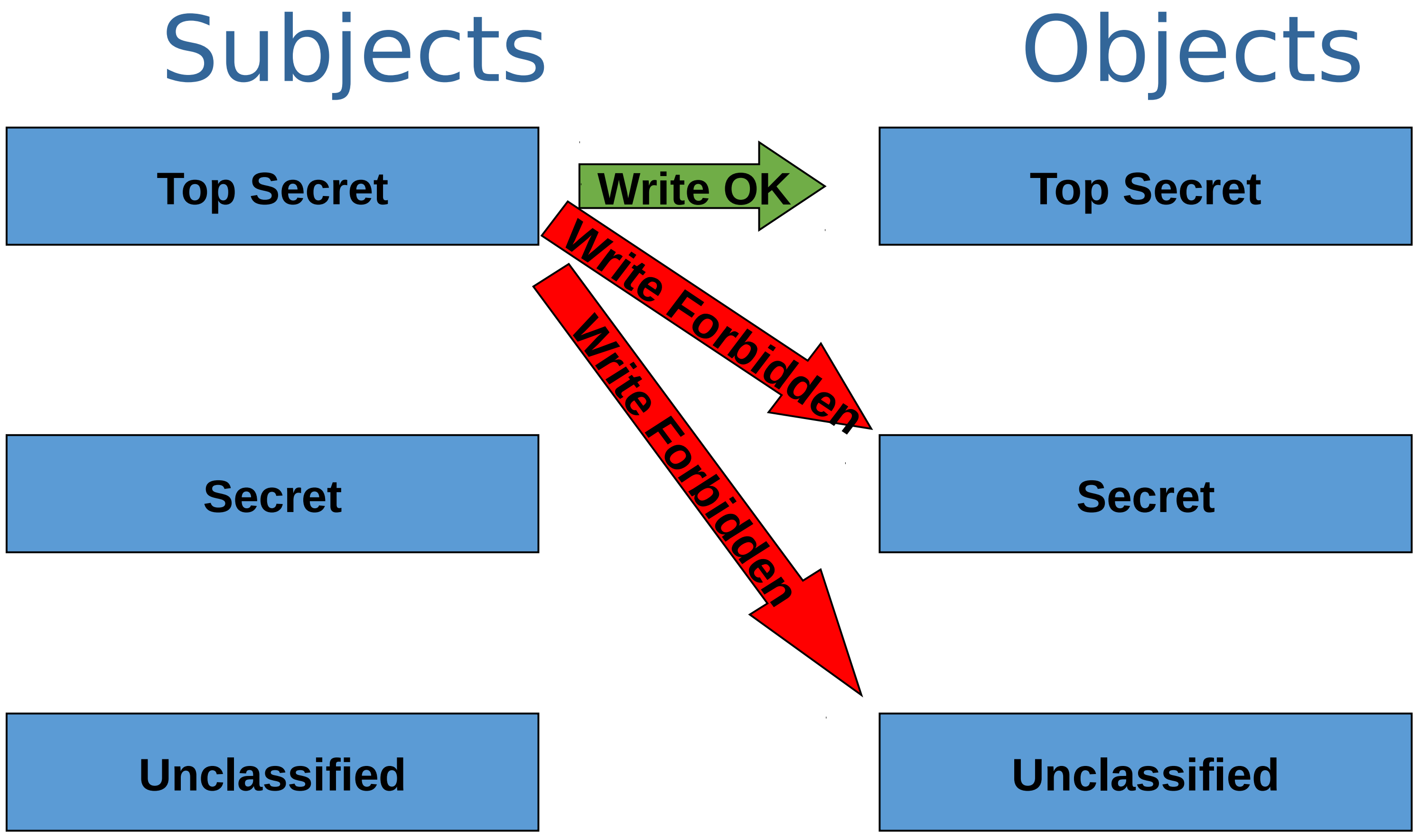
Bell and LaPadula Model

- Each user subject and information object has a fixed security class – labels
- Use the notation ≤ to indicate dominance
- Simple Security (ss) property: the no read-up property
  - A subject s has read access to an object iff the class of the subject C(s) is greater than or equal to the class of the object C(o)
  - i.e. Subjects can read Objects iff C(o) ≤ C(s)



Bell and LaPadula Model

- Each user subject and information object has a fixed security class – labels
- Use the notation ≤ to indicate dominance
- Simple Security (ss) property:
  the no write-down property
  - While a subject has read access to object *O*, the subject can only write to object *P* if C(O) ≤ C (P)

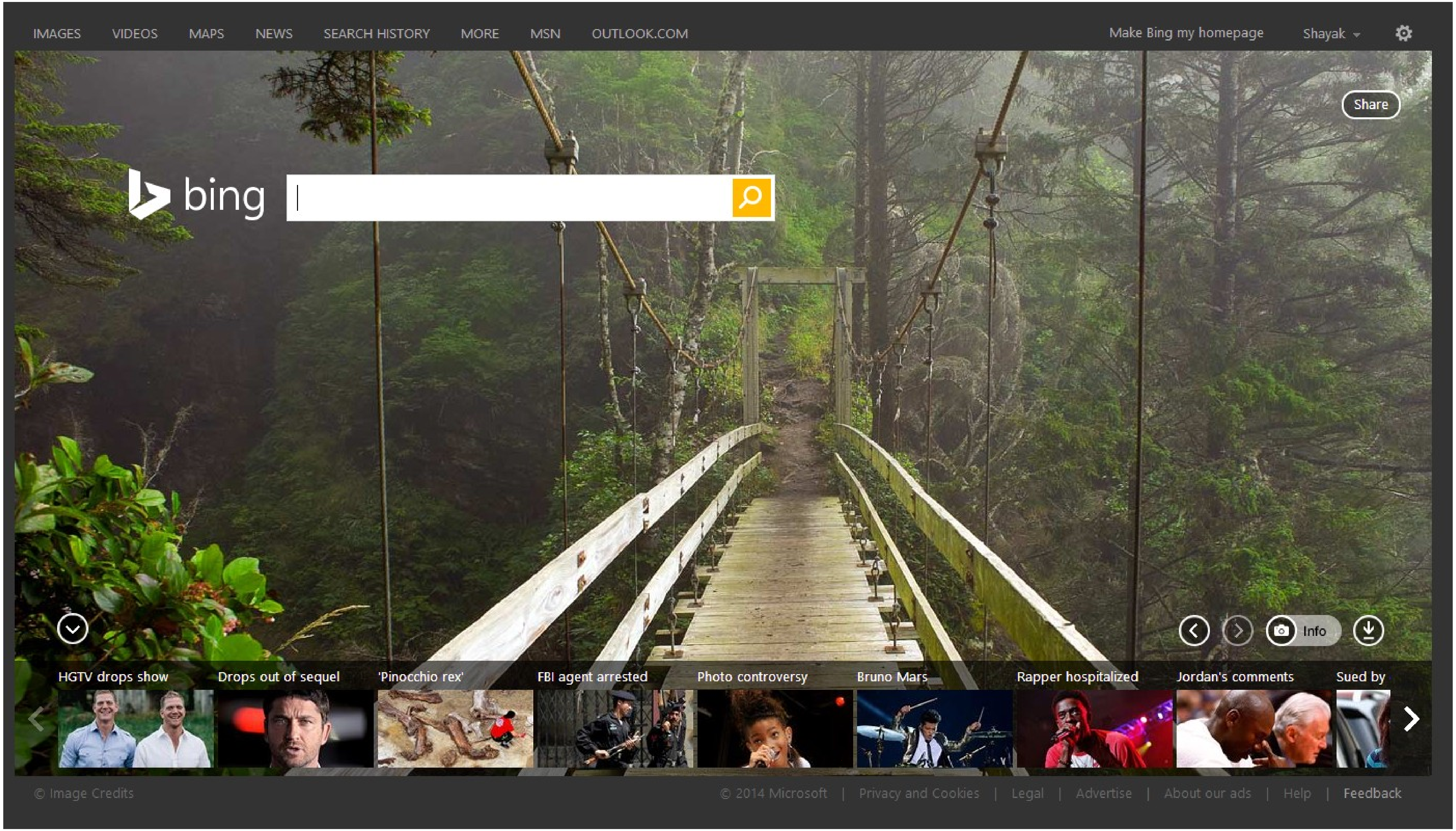


Our interest: Data Use policies

- Once you have access, what are you allowed to do with the data?
- These are called “use policies”
  - Notion of purpose is important: In many cases we need to specify usage for specific purposes such as “for fraud detection” or “for advertising”
  - Policies need to be specified independent of the program (since policies can change depending on regulation changes, for example)
  - Policies need to be specified across implementations in many programming languages
- This talk:
  - Specifying and checking data use policies

Overview of this talk

- Part 1. Specifying and checking data use policies in online services
- Part 2. Specifying and checking data use in enclave programs
- Part 3. Thoughts on combining the above two ideas


IMAGES
VIDEOS
MAPS
NEWS
SEARCH HISTORY
MORE
MSN
OUTLOOK.COM
Make Bing my homepage
Shayak
Share
bing
Info
HGTV drops show
Drops out of sequel
'Pinocchio rex'
FBI agent arrested
Photo controversy
Bruno Mars
Rapper hospitalized
Jordan's comments
Sued by
© Image Credits
© 2014 Microsoft | Privacy and Cookies | Legal | Advertise | About our ads | Help | Feedback

Print

Bing Privacy Statement

This privacy statement applies to Bing websites, services, products and applications that collect data and display these terms. It does not apply to other Microsoft products and services that do not link to the Bing Privacy Statement.

Collecting Your Information

When you use Bing services, Microsoft may collect many kinds of information in order to operate effectively and provide you the best products, services and experiences we can. We collect information when you register, sign in and use our sites and services. We also may get information from other companies. We collect this information in a variety of ways, including from web forms, technologies like cookies, web logging and software on your computer or other device.

When you conduct a search, Microsoft collects the following:

- Search term and time and date of your search
- IP address, browser configuration and approximate location
- Any unique identifiers contained in the cookies

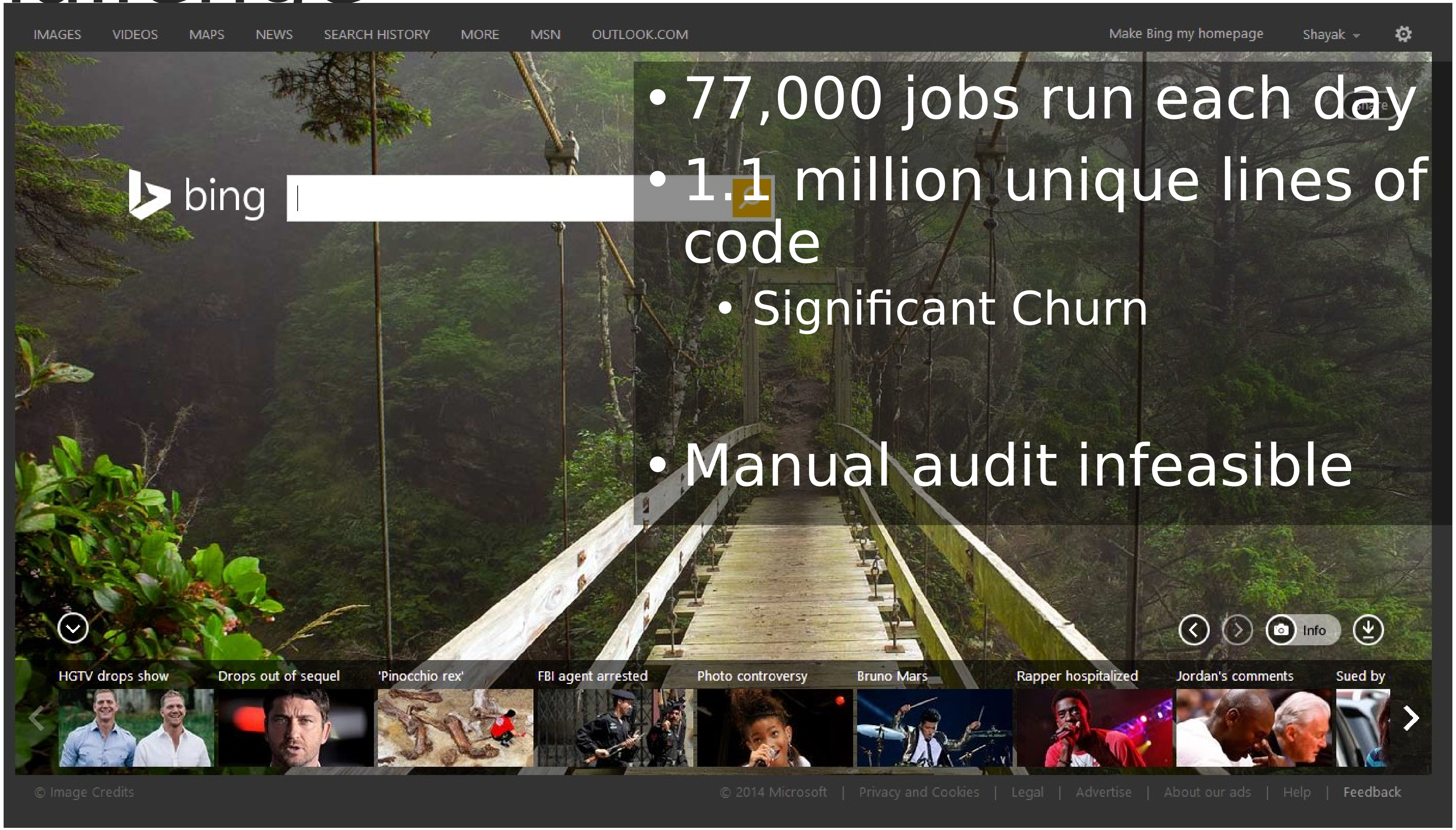
We store search terms (and the cookie IDs associated with search terms) separately from any account information that directly identifies the user, such as name, e-mail address, or phone numbers. We have technological safeguards in place designed to prevent the unauthorized correlation of this data and we remove the entirety of the IP address after 6 months, cookies and other cross session identifiers, after 18 months.

Bing provides search services to select partners and its users. Some examples include Yahoo! and Nokia. In order to provide these services, Bing services receive certain search related information from these partners that may include date, time, IP address, a unique identifier and other search related data.

↑Top of page

Learn More

Data Use Policies: Compliance Challenge



- 77,000 jobs run each day
- 1.1 million unique lines of code
  - Significant Churn
- Manual audit infeasible

Legalese

Policy Clause C	$::=$	$D \mid A$
Deny Clause D	$::=$	DENY $T_1 \cdots T_n$ EXCEPT $A_1 \cdots A_m$
		$\mid$ DENY $T_1 \cdots T_n$
Allow Clause A	$::=$	ALLOW $T_1 \cdots T_n$ EXCEPT $D_1 \cdots D_m$
		$\mid$ ALLOW $T_1 \cdots T_n$
Attribute T	$::=$	$\langle$attribute-name$\rangle$ $v_1 \cdots v_l$
Value v	$::=$	$\langle$attribute-value$\rangle$

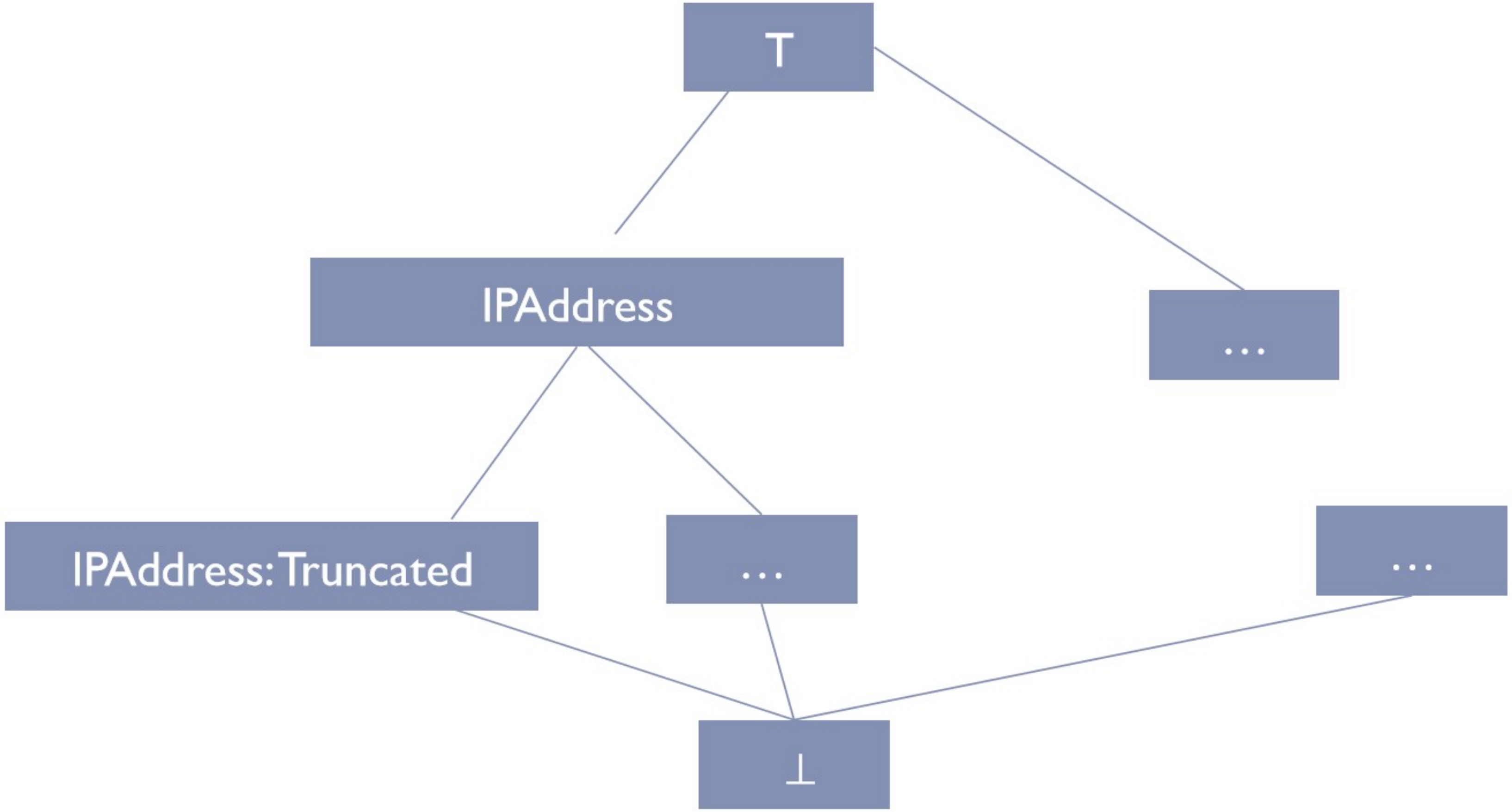
S. Sen, S. Guha, A. Dutta, S. Rajamani, J. Tsai and J. Wing,
“Bootstrapping Privacy Compliance in Big Data Systems,” In Proceedings of the 35th IEEE Symposium on Security & Privacy (Oakland), San

Legalese

DENY ***Datatype*** **IPAddress**
UseForPurpose **Advertising**
EXCEPT
ALLOW
Datatype **IPAddress:Truncated**
ALLOW
UseForPurpose **AbuseDetect**
EXCEPT
DENY ***Datatype***
IPAddress, AccountInfo

We will **not** use **full IP Address** for **Advertising**. IP Address may be used for **detecting abuse**. In such cases, it will not be combined with **account information.**

Lattice of policy labels



- If IPAddress is allowed, then everything below is allowed
- If IPAddress:Truncated is denied then everything above it is denied
- Type state is modeled as transition over labels

More encodings for Bing policies

ALLOW	
EXCEPT	
DENY *DataType* IPaddress:Expired	◁ "we remove the entirety of the IP address after 6 months"
DENY *DataType* UniqueIdentifier:Expired	◁ "[we remove] cookies and other cross session identifiers, after 18 months"
DENY *DataType* SearchQuery, PII *InStore* Store DENY *DataType* UniqueIdentifier, PII *InStore* Store	◁ "We store search terms (and the cookie IDs associated with search terms) separately from any account information that directly identifies the user, such as name, e-mail address, or phone numbers."
DENY *DataType* BBEPData *UseForPurpose* Advertising	◁ "we do not use any of the information collected through the Bing Bar Experience Improvement Program to identify, contact or target advertising to you"
DENY *DataType* BBEPData, PII *InStore* Store	◁ "we take steps to store [information collected through the Bing Bar Experience Improvement Program] separately from any account information we may have that directly identifies you, such as name, e-mail address, or phone numbers"
DENY *DataType* BBEPData:Expired	◁ "we delete the information collected through the Bing Bar Experience Program at eighteen months."
DENY *DataType* UserProfile, PII *InStore* Store	◁ "we store page views, clicks and search terms used for ad targeting separately from contact information you may have provided or other data that directly identifies you (such as your name, e-mail address, etc.)."
DENY *DataType* PII *UseForPurpose* Advertising DENY *DataType* PII *InStore* AdStore	◁ "our advertising systems do not contain or use any information that can personally and directly identify you (such as your name, email address and phone number)."
DENY *DataType* SearchQuery *UseForPurpose* Sharing EXCEPT ALLOW *DataType* SearchQuery:Scrubbed	◁ "Before we [share some search query data], we remove all unique identifiers such as IP addresses and cookie IDs from the data."

TABLE V
AN ENCODING OF PRIVACY PROMISES BY BING AS OF OCTOBER 2013

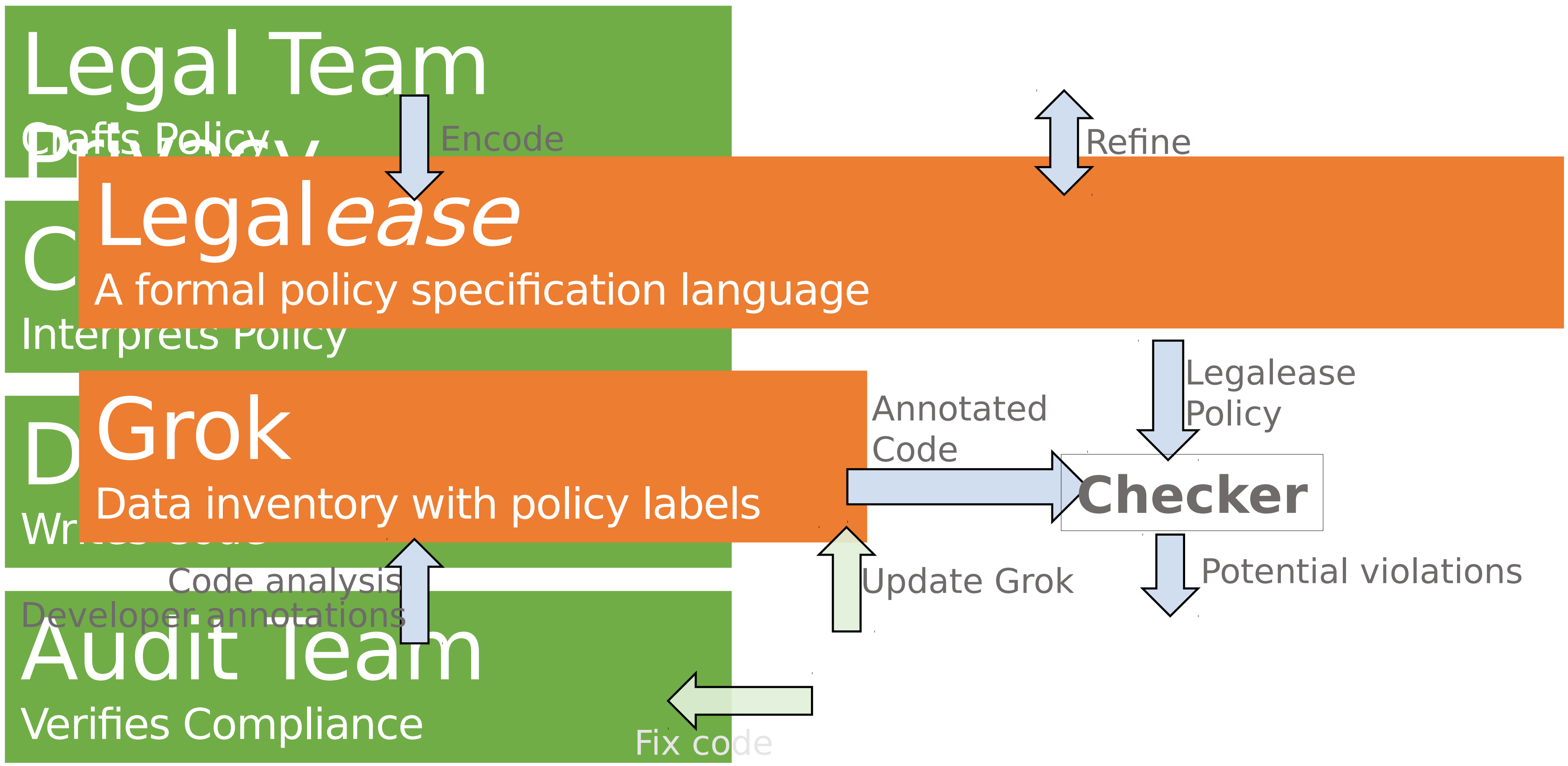
Another example

ALLOW	
EXCEPT	
DENY *DataType* PII *UseForPurpose* Sharing	◁ "We do not share personal information with companies, organizations and individuals outside of Google unless one of the following circumstances apply:"
EXCEPT ALLOW *DataType* PII:OptIn	◁ "We require opt-in consent for the sharing of any sensitive personal information."
EXCEPT ALLOW *AccessByRole* Affiliates	◁ "We provide personal information to our affiliates or other trusted businesses or persons to process it for us"
EXCEPT ALLOW *UseForPurpose* Legal	◁ "We will share personal information [if necessary to] meet any applicable law, regulation, legal process or enforceable governmental request."
DENY *DataType* DoubleClickData, PII EXCEPT ALLOW *DataType* DoubleClickData, PII:Optin	◁ "We will not combine DoubleClick cookie information with personally identifiable information unless we have your opt-in consent"

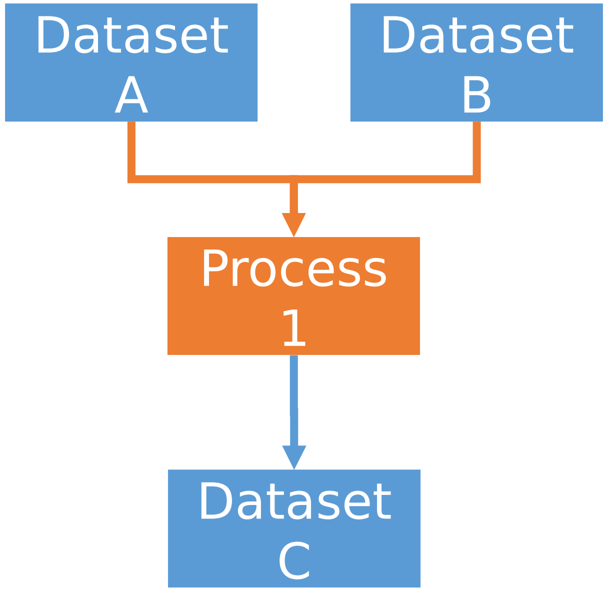
TABLE VI

AN ENCODING OF PRIVACY PROMISES BY GOOGLE AS OF OCTOBER 2013

A Streamlined Audit Workflow



Map-Reduce Programming Systems



Scope, Hive, Dremel

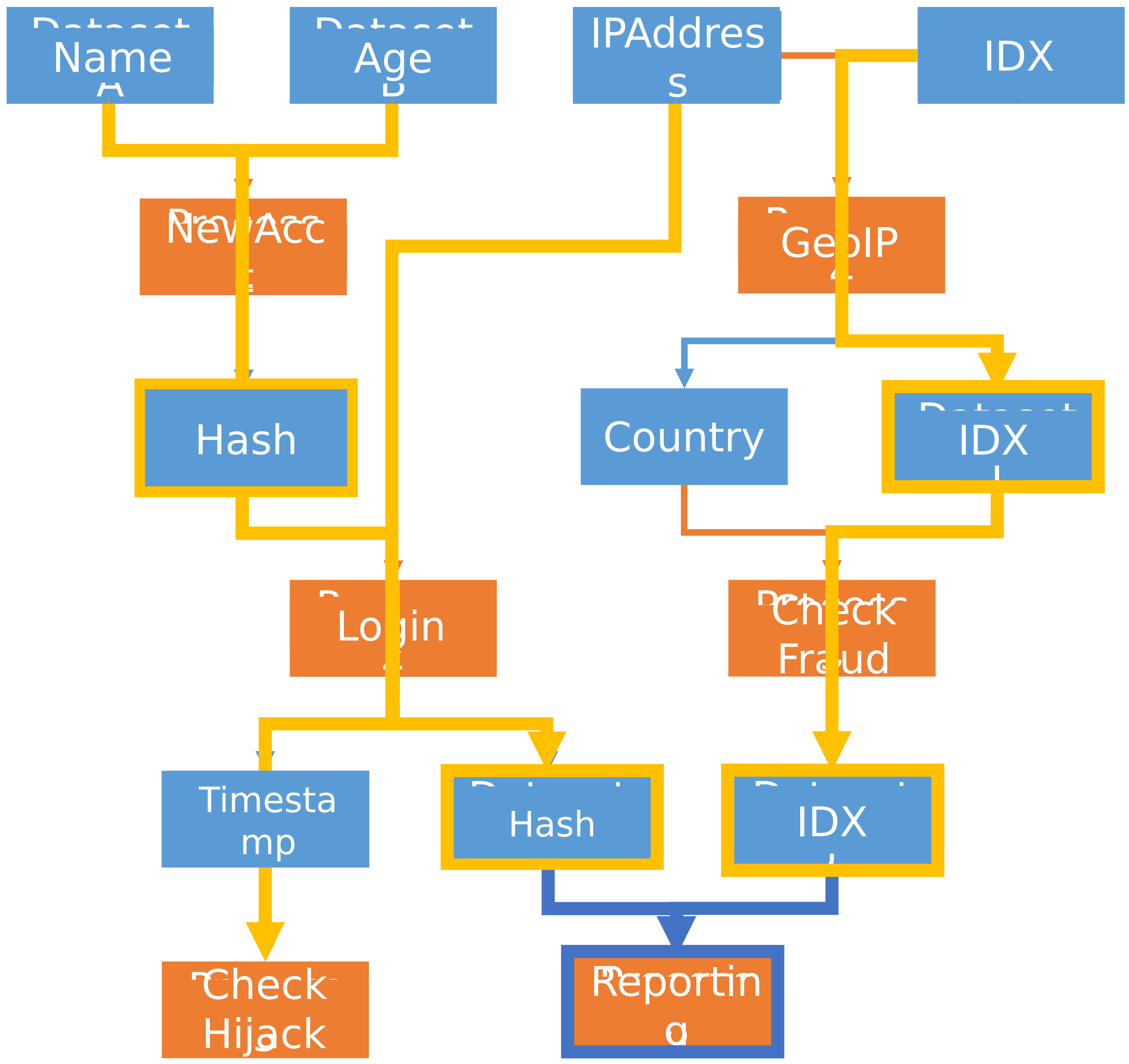
Data in the form of Tables

Code Transforms Columns to Columns

- No Shared State
- Limited Hidden Flows

Data Inventory

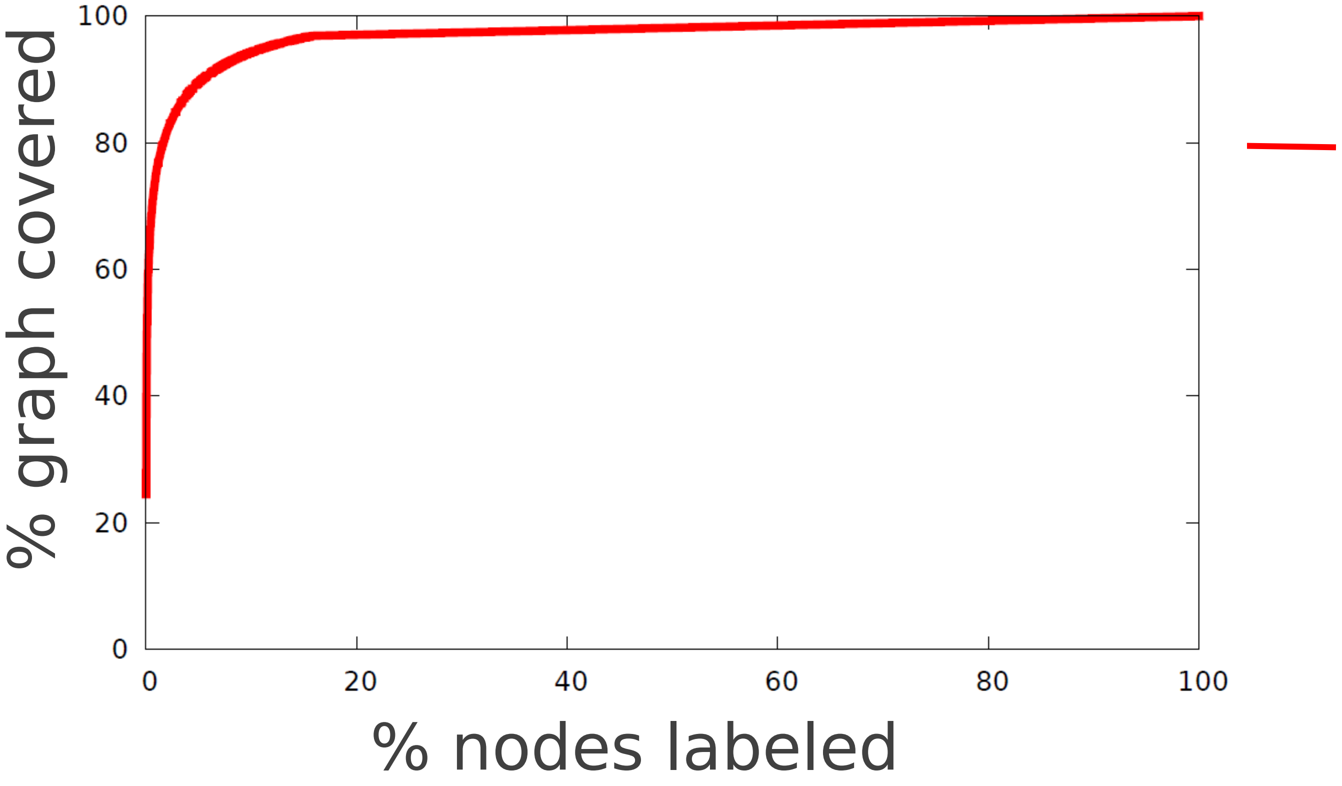
- Annotate code + data with policy data types. Expensive!
- We performs “type inference”. That is propagate labels via data flow graph



Combine Noisy Sources

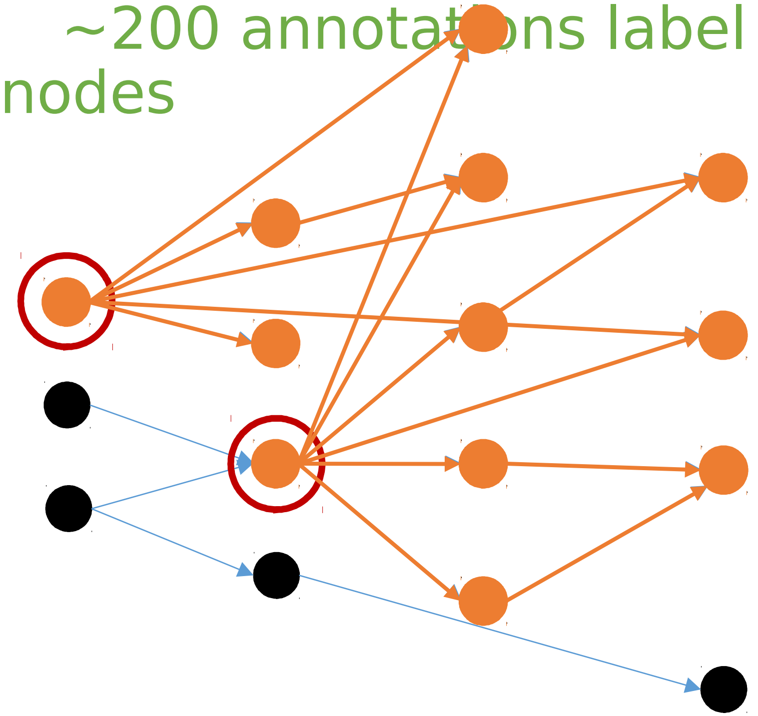
Variable Name Analysis	Developer Annotations	Auditor Verification
Carefully curated regular expressions Leverages developer conventions Significant Noise	Expensive Low Noise	Very Expensive Definitive Need very few of these

Bootstrapping Inference Using Annotations



Pick the nodes which will label the most of the graph

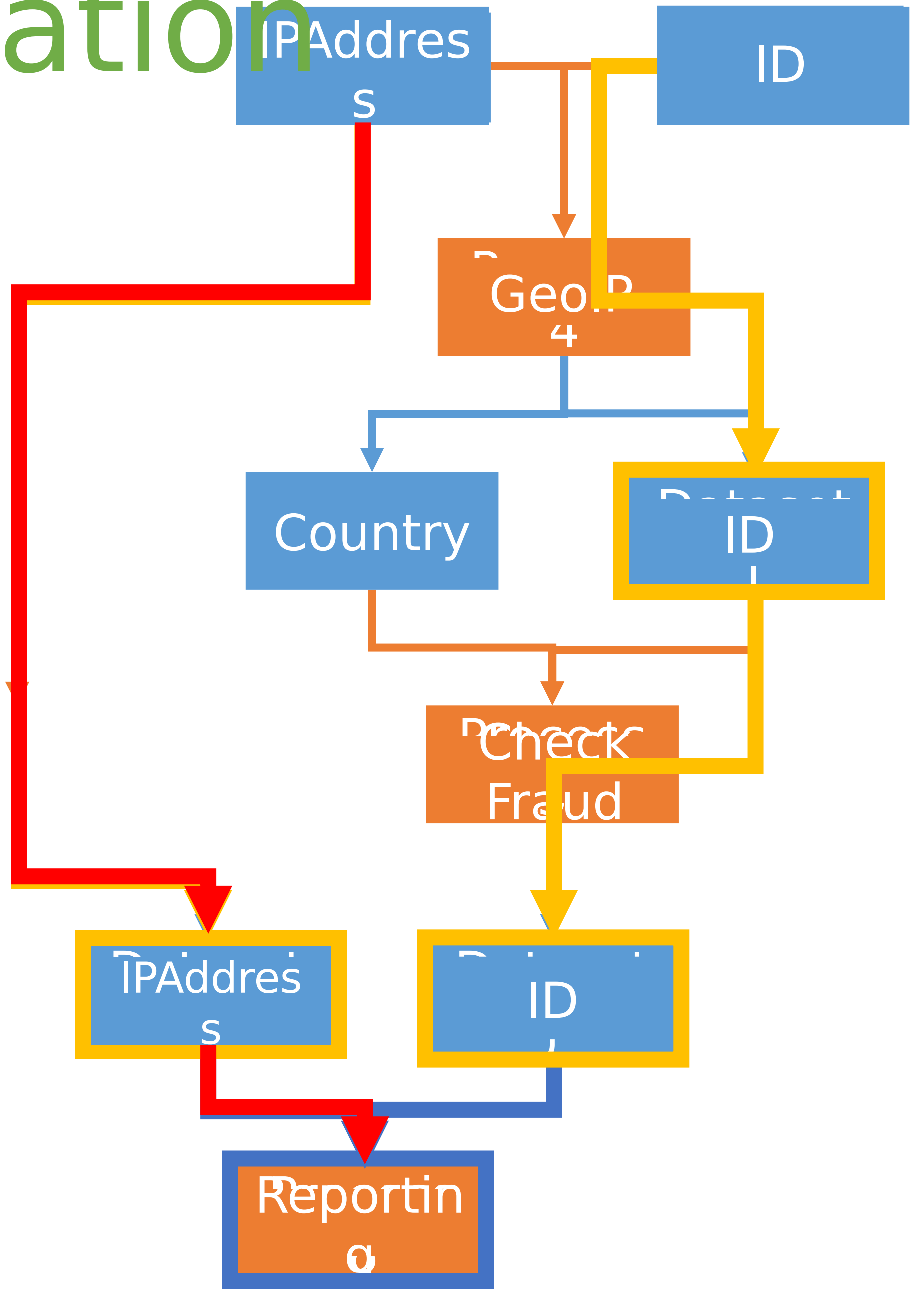
~200 annotations label 60% of nodes



A small number of annotations is enough to get off the ground.

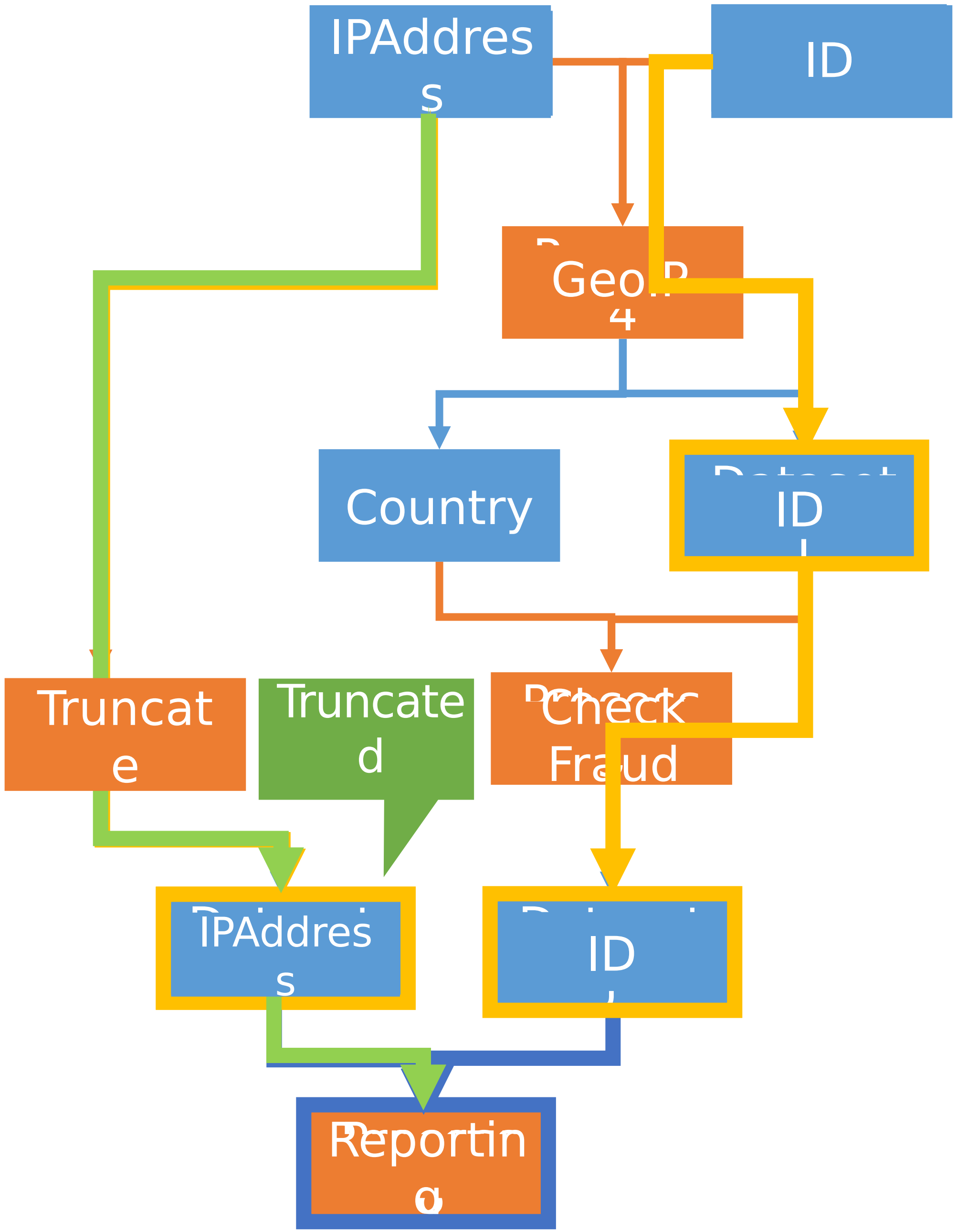
Example Policy Violation

IPAddress is used for reporting (advertising)



Example Fix

IPAddress is truncated before it is passed to reporting (advertising) job



Current state

- Used extensively to check data use policies inside Microsoft (in Bing, and several other services)
- Used to check GDPR compliance

S. Sen, S. Guha, A. Dutta, S. Rajamani, J. Tsai and J. Wing,
“Bootstrapping Privacy Compliance in Big Data Systems,” In Proceedings of the 35th IEEE Symposium on Security & Privacy (Oakland), San

Overview of this talk

- Part 1. Specifying and checking data use policies in online services
- Part 2. Specifying and checking data use in enclave programs
- Part 3. Thoughts on combining the above two ideas

Introducing Azure confidential computing

Posted on 14 September, 2017

Mark Russinovich, CTO, Microsoft Azure

Microsoft spends one billion dollars per year on cybersecurity and much of that goes to making Microsoft Azure the most trusted cloud platform. From strict physical datacenter security, ensuring data privacy, encrypting data at rest and in transit, novel uses of machine learning for threat detection, and the use of stringent operational software development lifecycle controls, Azure represents the cutting edge of cloud security and privacy.

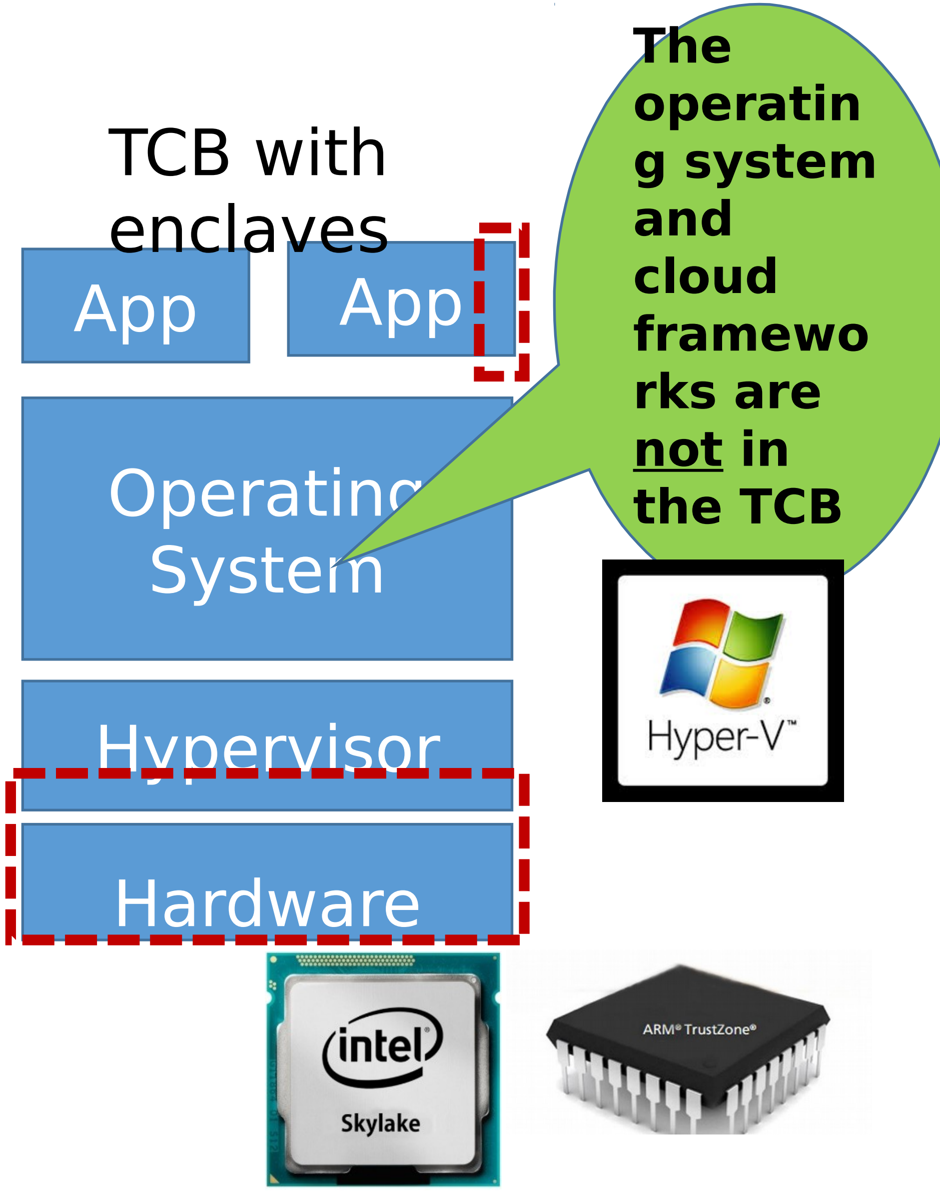
Today, I'm excited to announce that Microsoft Azure is the first cloud to offer new data security capabilities with a collection of features and services called Azure confidential computing. Put simply, confidential computing offers a protection that to date has been missing from public clouds, encryption of data while in use. This means that data can be processed in the cloud with the assurance that it is always under customer control. The Azure team, along with Microsoft Research, Intel, Windows, and our Developer Tools group, have been working on confidential computing software and hardware technologies for over four years. The bottom of this post includes a list of Microsoft Research papers related to confidential computing. Today we take that cutting edge one step further by now making it available to customers via an Early Access program.

Data breaches are virtually daily news events, with attackers gaining access to personally identifiable information (PII), financial data, and corporate intellectual property. While many breaches are the result of poorly configured access control, most can be traced to data that is accessed while in use, either through administrative accounts, or by leveraging compromised keys to access encrypted data. Despite advanced cybersecurity controls and mitigations, some customers are reluctant to move their most sensitive data to the cloud for fear of attacks against their data when it is in-use. With confidential computing, they can move the data to Azure knowing that it is safe not only at rest, but also in use from the following threats:

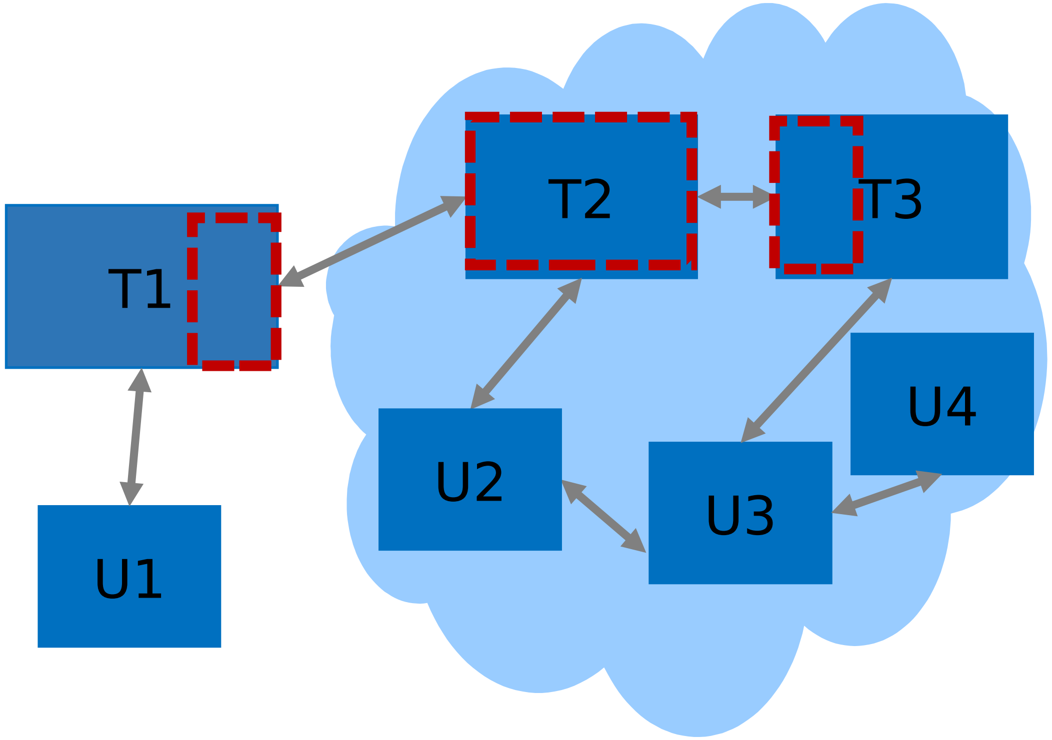
- Malicious insiders with administrative privilege or direct access to hardware on which it is being processed
- Hackers and malware that exploit bugs in the operating system, application, or hypervisor
- Third parties accessing it without their consent

Enclaves and TCB

- Adversary *cannot* observe or compromise code inside the enclaves.
- Adversary can compromise all code outside the enclaves (including OS), can send and receive messages to enclaves, and create new enclaves with arbitrary code
- Adversary can cause interrupts and switch context out of the enclaves.
- Difficulty: Defending against side-channels



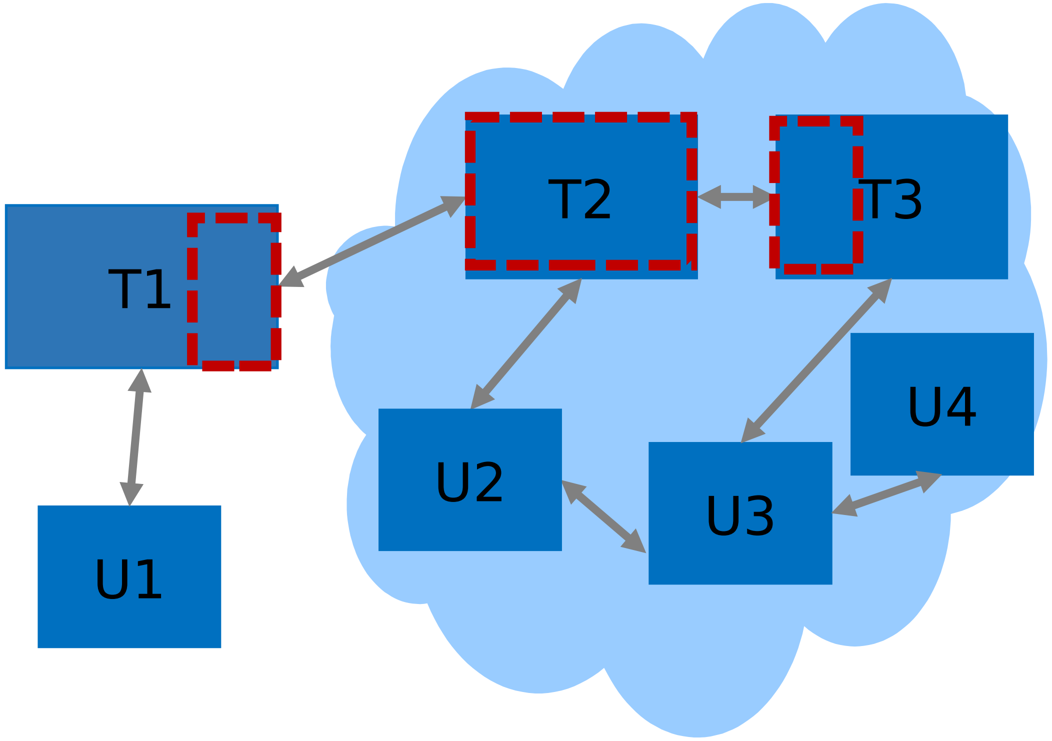
World view



- Every service contains trusted and untrusted components
  - Data is encrypted in untrusted components
  - Keys are available inside enclaves and data is in clear inside enclaves

- How do we know that code inside enclaves doesn’t leak secrets? Two approaches:
  - Release source code to the user and let them do manual code review
  - Perform automated verification of confidentiality of the code inside the enclave

Confidentiality



Defined as a "hyper-property" over traces (sequence of states): Adversary should not be able to infer secrets based on observations, no matter what actions it makes.

Adversarial semantics: Allow adversary to havoc non-enclave memory after every program instruction and observe non-enclave memory after every program instruction

Confidentiality: For a pair of traces t_1 and t_2 that potentially differ in the value of secrets, if the actions of the adversary in t_1 and t_2 are identical, then the observations of the adversary must be equivalent

Verifying Confidentiality – Take one

```
void Reduce(BYTE *nameEnc, BYTE *valuesEnc, BYTE *outputEnc) {
    KeyAesGcm *aesKey = ProvisionKey();

    char name[KEY_SIZE];
    aesKey->Decrypt(nameEnc, name, KEY_SIZE);

    char valuesBuf[VALUES_SIZE];
    aesKey->Decrypt(valuesEnc, valuesBuf, VALUES_SIZE);
    StringList *values = (StringList*)valuesBuf;

    long long usage = 0;
    for (char *value = values->begin();
        value != values->end(); value = values->next()) {
        long lvalue = mystrtol(value, NULL, 10);
        usage += lvalue;
    }

    char cleartext[BUF_SIZE];
    sprintf(cleartext, "%s %lld", name, usage);
    aesKey->Encrypt(cleartext, outputEnc, BUF_SIZE);
}
```

Check that key is neither leaked nor overwritten

Check control flow integrity

Check that secrets are encrypted before written to output buffer

Verifying Confidentiality

```
void Reduce(BYTE *nameEnc, BYTE *valuesEnc, BYTE *outputEnc) {
        KeyAesGcm *aesKey = ProvisionKey();

        char name[KEY_SIZE];
        aesKey->Decrypt(nameEnc, name, KEY_SIZE);

        char valuesBuf[VALUES_SIZE];
        aesKey->Decrypt(valuesEnc, valuesBuf, VALUES_SIZE);
        StringList *values = (StringList*)valuesBuf;

        long long usage = 0;
        for (char *value = values->begin();
                value != values->end(); value = values->next()) {
                long lvalue = mystrtol(value, NULL, 10);
                usage += lvalue;
        }

        char cleartext[BUF_SIZE];
        sprintf(cleartext, "%s %lld", name, usage);
        aesKey->Encrypt(cleartext, outputEnc, BUF_SIZE);
}
```

Check that key is neither leaked nor overwritten

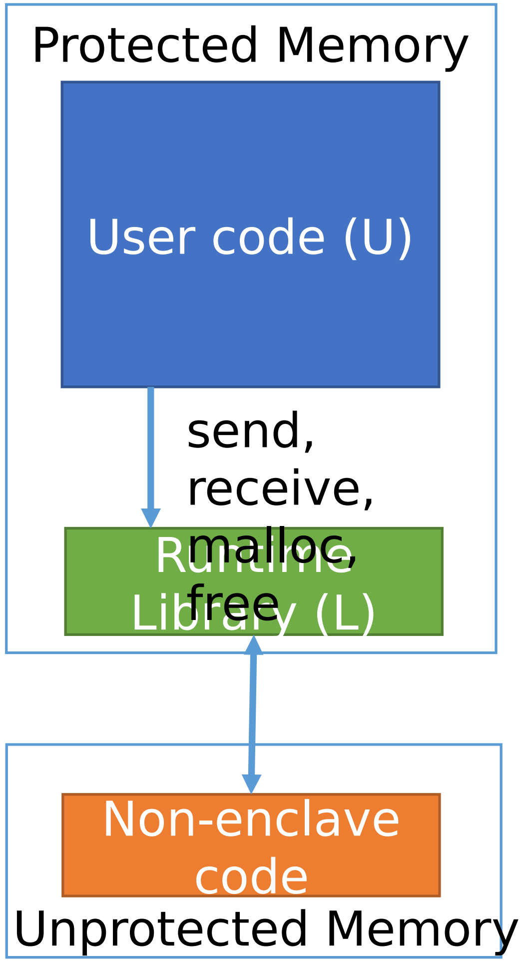
Check control flow integrity

Check that secrets are encrypted before written to output buffer

Rohit Sinha, Sriram Rajamani, Sanjit A. Seshia, and Kapil Vaswani, **Moat: Verifying Confidentiality of Enclave Programs**, in *CCS 2015*

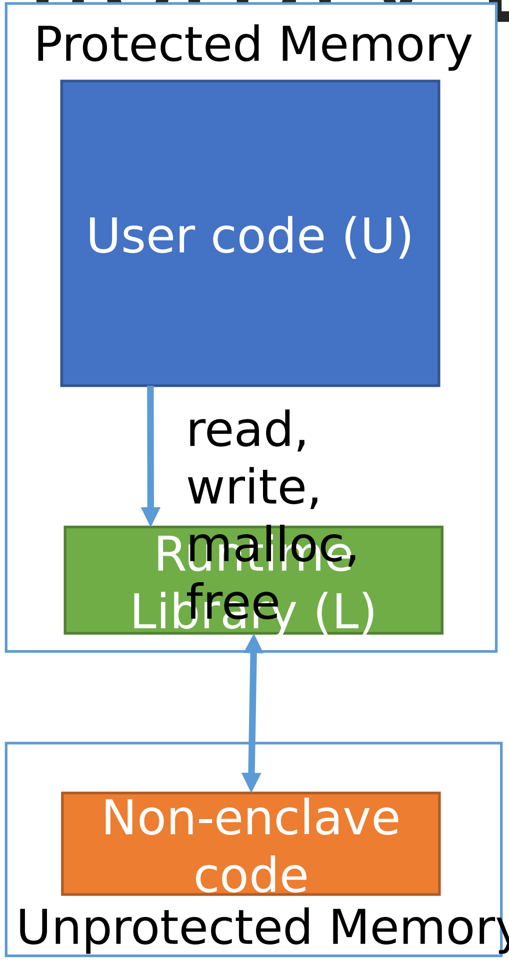
- Check that keys are neither leaked not overwritten
- Check control flow integrity
- Check that secrets are encrypted with the appropriate key before output
  - This requires tracking which memory locations hold secrets
  - Either requires annotations or does not scale
- Check that enclave is created properly using provider instructions (eg. SGX)
  - Requires modelling and analysing special instructions (eg. SGX

Verifying Confidentiality – Take two



- Impose structure, and decompose the problem.
- Link user application U with a small runtime L with a restricted API, which provides memory management and communication.
- Can now decompose the check to two parts:
  - Check that user code U accesses L only through the APIs
  - Check that implementation of L does not leak secrets

Reducer example rewritten with library L




```
void Reduce(Channel<String*>& channel)
{
     char *name = channel.recv<char*>(KEY_SIZE);

      StringList *values = (StringList*)
      channel.recv<StringList*>(VALUE_SIZE);

      long long usage = 0;
      for (char *value = values->begin();
    value != values->end();
          value = values->next()) {
    long lvalue = mystrtol(value, NULL, 10);
          usage += lvalue;
      }

       char cleartext[BUF_SIZE];
       sprintf(cleartext, BUF_SIZE, "%s %lld",

                  name, usage);
       channel.send<char*>(cleartext);
}
```

No need to worry about key management or special instructions (push these problems to L)

Entire address space of U can be considered as secret (avoids fine grained flow-tracking)

Still need to check for stack overrun, control flow integrity, and memory accesses

Information Release Confinement (IRC)

- A trace satisfies IRC, if every update to adversary-observable state is either done by “call send” action from U, or from adversary-initiated actions.
- A user program U satisfies IRC if all traces of U satisfy IRC.
- IRC together with a suitable implementation of L guarantees confidentiality.

CFI-RW

\Confidential: Verifier

User code U satisfies CFI-RW if for all procedures *p* within U, *p* satisfies the following properties:

1. No reads or writes to L’s memory. No writes to non-SIR memory
2. Any **ret** instruction within *p* uses the return address saved by the **call** into *p*
3. Any **call** instruction within *p* targets the starting address of a procedure in U, or to L’s API entry procedure.
4. Any (direct / indirect) **jmp** instruction targets a legal instruction within *p*

Modular verifier for CFI-RW

1. Processes each procedure *p* of U separately (at the level of machine code)
2. Instruments assertions for each store, call, ret, and jump instruction in *p*
3. Generates a verification condition *VC(p)* which is discharged using Z3 (via Boogie)

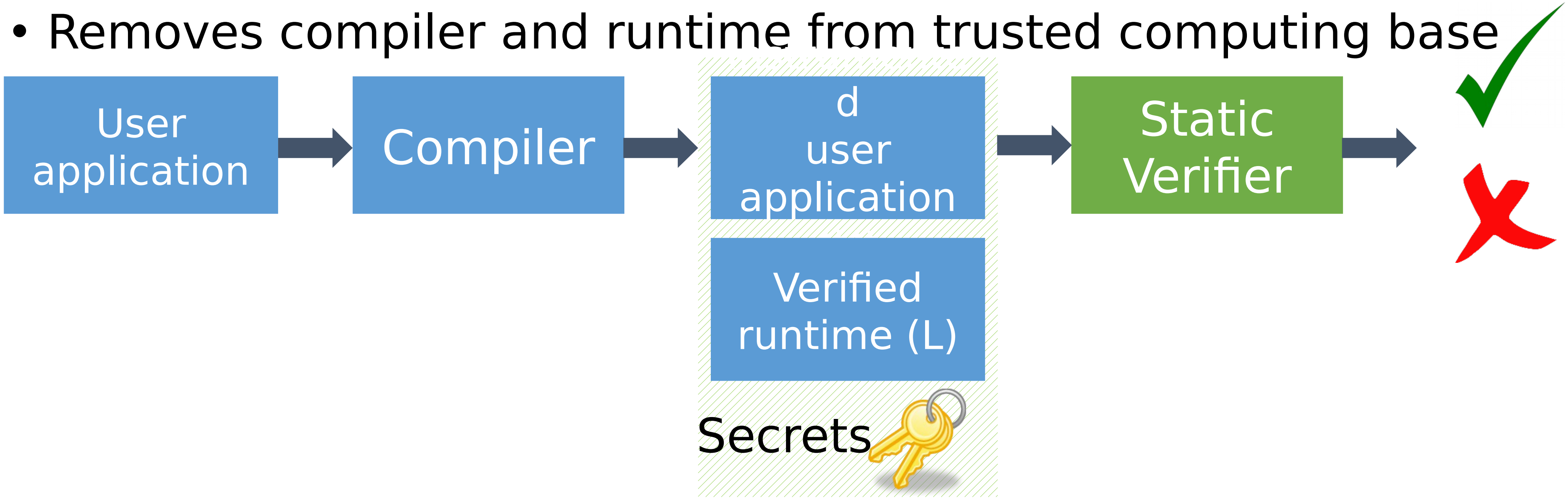
Insight: \Guard style compiler enables local and modular verification (with compiler outside TCB),

Theorem: ***For any trace t, if t satisfies CFI-RW and particular specifications on procedures in L, then t satisfies IRC***

Theorem: If all procedures p satisfy VC(p) then program U satisfies CFI-RW

Verifying confidentiality

- Use a compiler that instruments memory accesses
- Verify that instrumented binary does not leak secrets
  - Removes compiler and runtime from trusted computing base



A Design and Verification Methodology for Secure Isolated Regions
Rohit Sinha, Manuel Costa, Akash Lal, Nuno Lopes, Sanjit Seshia, Sriram Rajamani, and Kapil Vaswani
ACM Conference on Programming Languages Design and Implementation (PLDI), June 2016

Page Fault Side Channel

- Adversary in SGX can observe page addresses during page faults (at the granularity of pages, not locations)
- Question: can we plug this side channel using a compiler and verifier?

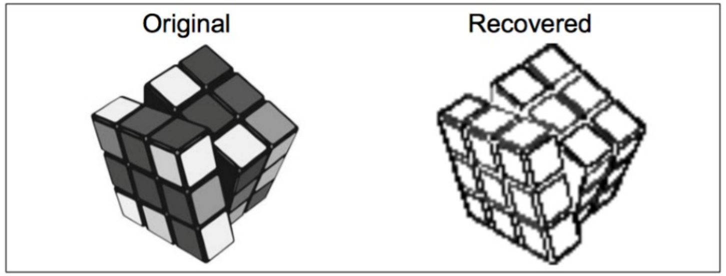
Idea: Can we build a compiler which ensures Page Access Obliviousness (PAO)?

i.e, sequence of pages accessed by the program are independent of secrets inside the enclave.

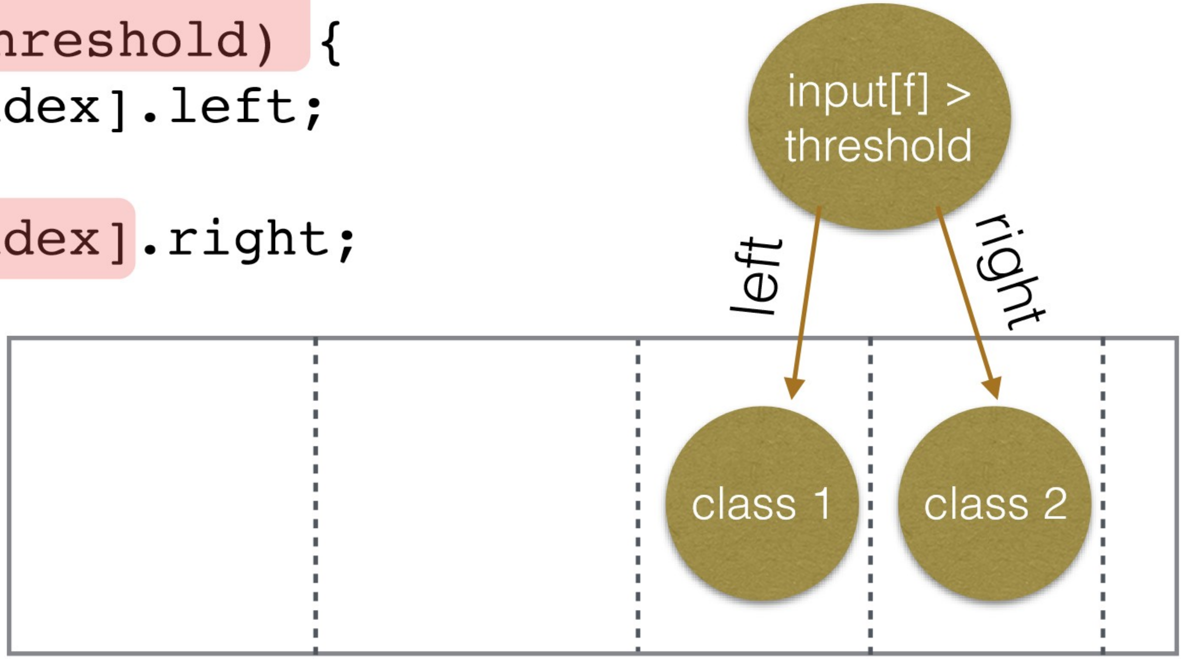
Page Faults Reveal Your Secrets

Attacker learns page-level accesses

Controlled Channel Attacks [XBP15]




```
void decisionTreeEvaluate(input)
{
  while (decision not yet made) {
    if (input[feature] >
        tree[index].threshold) {
      index := tree[index].left;
    } else {
      index := tree[index].right;
    }
  }
  ...
}
```



Page Access Obliviousness

Access to code and data (at the level of pages) should be independent of secrets

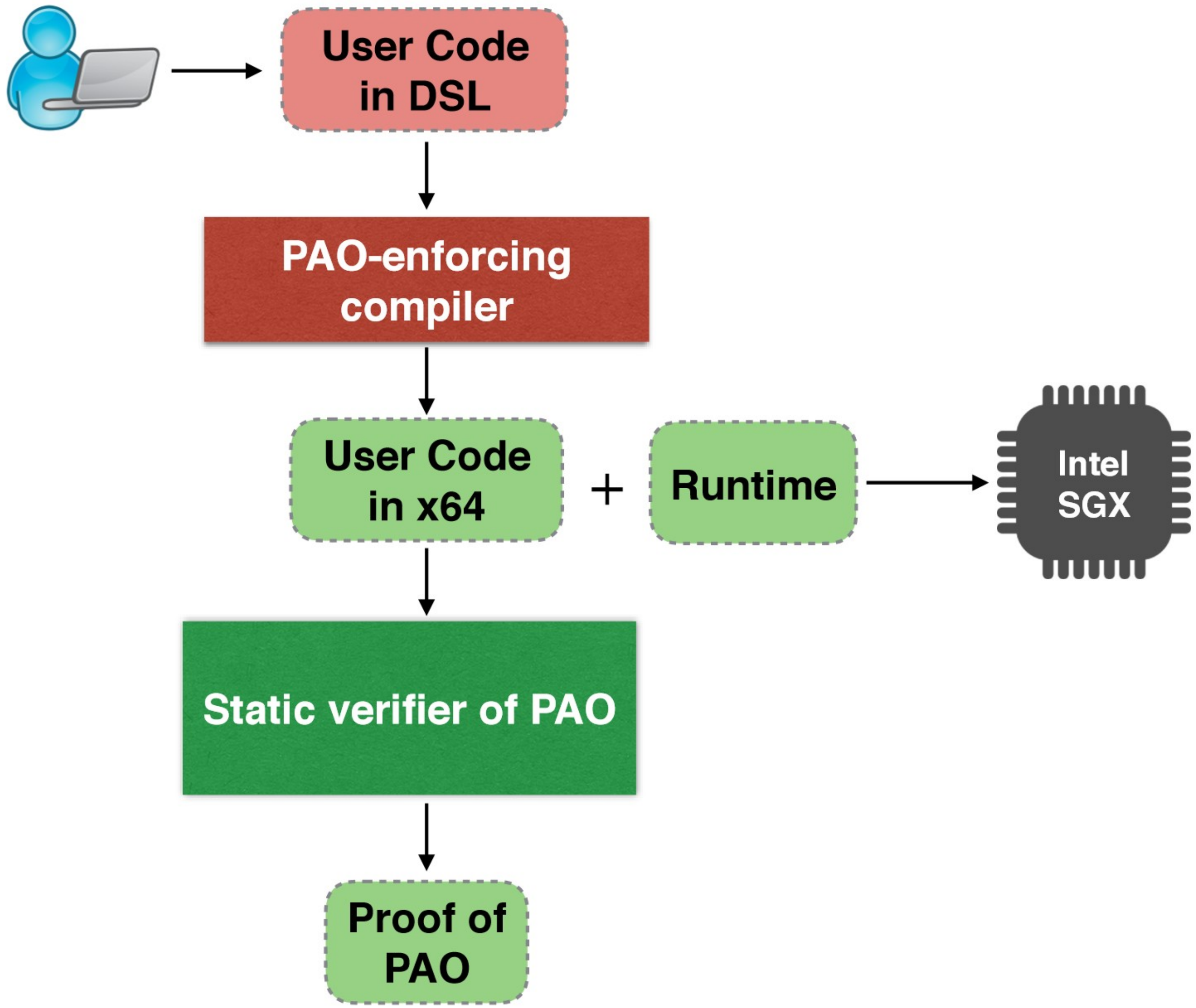
Formal Specification of Page Access Obliviousness

Attacker Observes:

- page containing current instruction: `Execute(rip mod 4096)`
- page containing addresses accessed by current instruction:
  - `mov regd [rega]: Read(rega mod 4096)`
  - `mov [rega] regd: Write(rega mod 4096)`
  - `ret: Read(rsp mod 4096)`
  - …

Formulated as non-interference: any pair of executions with the same attacker operations must have the same observations i.e. observations independent of secrets

Compiler and Verifier for PAO



How the compiler works

- Prohibits data dependent loops
- Adds dummy data accesses, lays out code and data to create page access obliviousness

```
if (s) {
  b[i] := a[k];
} else {
  c := 0;
}
```

If: Read (s), Read(a[k]), Write (b[i]),

Else: Read (s), Write(c),

How the compiler works

- Prohibits data dependent loops
- Adds dummy data accesses, lays out code and data to create page access obliviousness

```
if (s) {
  b[i] := a[k];
} else {
  c := 0;
}
```

If: Read (s), Read(a[k]), Write (b[i]), **<u>Write (c)</u>**

Else: Read (s), Write(c), **<u>Read(a[k])</u>**, **<u>Write (b[i])</u>**

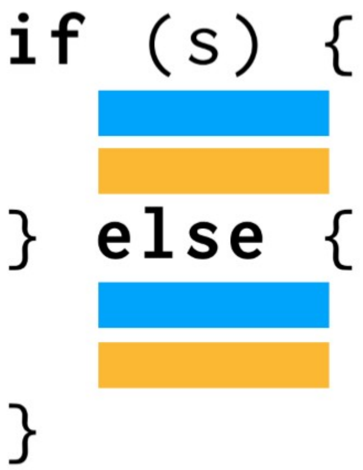
How the compiler works

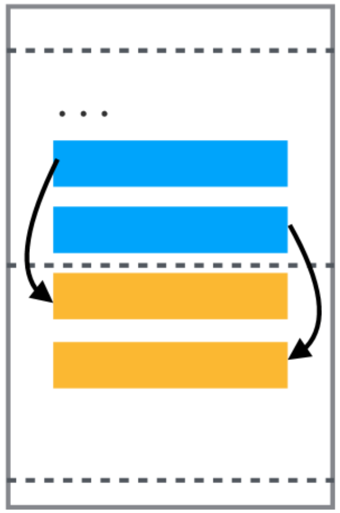
- Prohibits data dependent loops
- Adds dummy data accesses, lays out code and data to create page access obliviousness

```
if (s) {
  b[i] := a[k];
} else {
  c := 0;
}
```

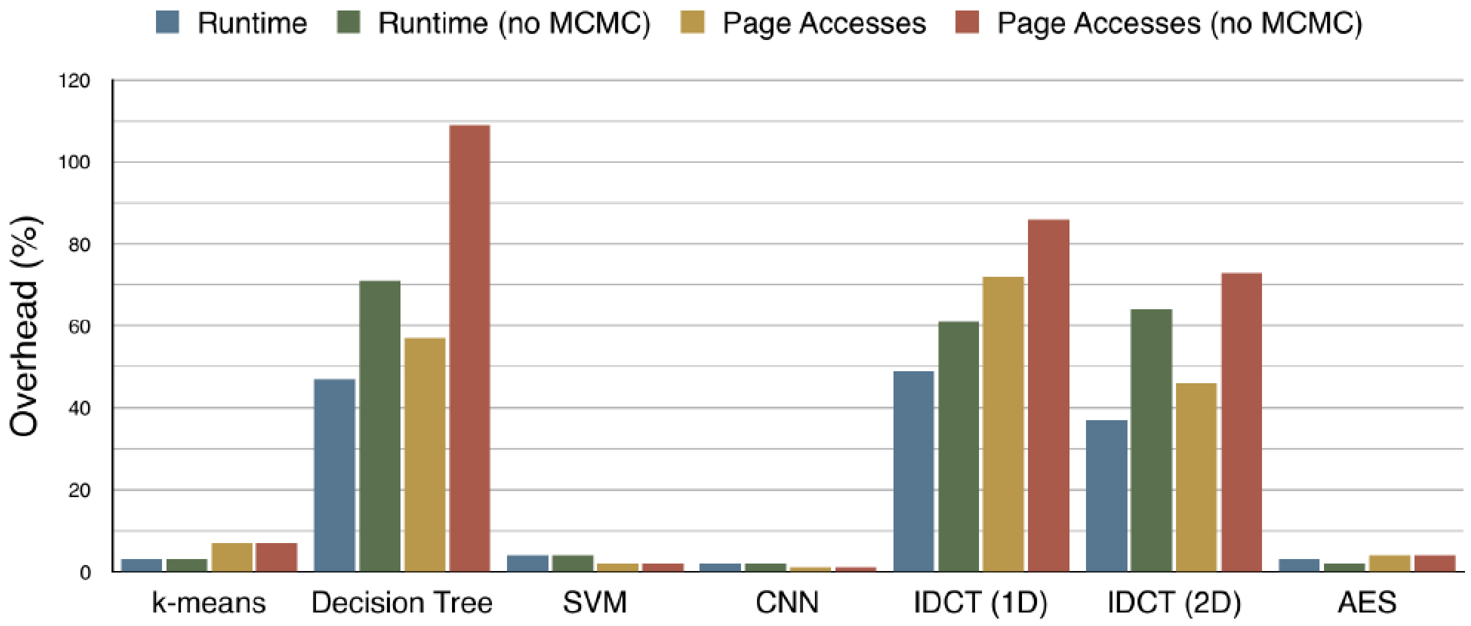
If: Read (s), Read(a[k]), Write (b[i]),

Else: Read (s), Write(c), **<u>Read(a[k])</u>**





Performance



- Can be done with some overhead
- Overhead reduces due to MCMC optimization of dummy accesses

[A Compiler and Verifier for Page Access Oblivious Compilation]
Rohit Sinha, Sriram Rajamani, Sanjit Seshia

Overview of this talk

- Part 1. Specifying and checking data use policies in online services
- Part 2. Specifying and checking data use in enclave programs
- Part 3. Thoughts on combining the above two ideas

Specifying data use policies in an adversarial setting

- Multiple data owners $o_1, o_2, \ldots$
- Each owner has their own policy $p_1, p_2, \ldots$ on who can access the data and what they can do with it
- Code that processes the data can be written using a variety of systems (Hadoop, Spark, Hive, etc)
- Goal: enforce that polices are adhered to about not only immediate users of data, but also derived uses of data

(ongoing collaboration with Ankush Desai, Pramod Subramanyan, Sanjit Seshia)

Summary

- Data use policies in non-adversarial settings can be specified independent of the code (e.g. Legalese)
- Data use in enclaves can be verified at the binary level
- Ongoing: Combining the two ideas to check data use in adversarial settings

References

Bootstrapping Compliance in Big Data Systems
Shayak Sen, Saikat Guha, Anupam Datta, Sriram K. Rajamani, Janice Y. Tsai, Jeannette M. Wing
IEEE Symposium on Security and Privacy (S&P), May 2014.

Moat: Verifying Confidentiality of Enclave Programs
Rohit Sinha, Sriram Rajamani, Sanjit A. Seshia, and Kapil Vaswani
ACM Conference on Computer and Communications Security (CCS), October 2015.

A Design and Verification Methodology for Secure Isolated Regions

Rohit Sinha, Manuel Costa, Akash Lal, Nuno Lopes, Sanjit Seshia, Sriram Rajamani, and Kapil Vaswani
ACM Conference on Programming Languages Design and Implementation (PLDI), June 2016.

A Compiler and Verifier for Page Access Oblivious Compilaton
Rohit Sinha, Sriram Rajamani, Sanjit Seshia