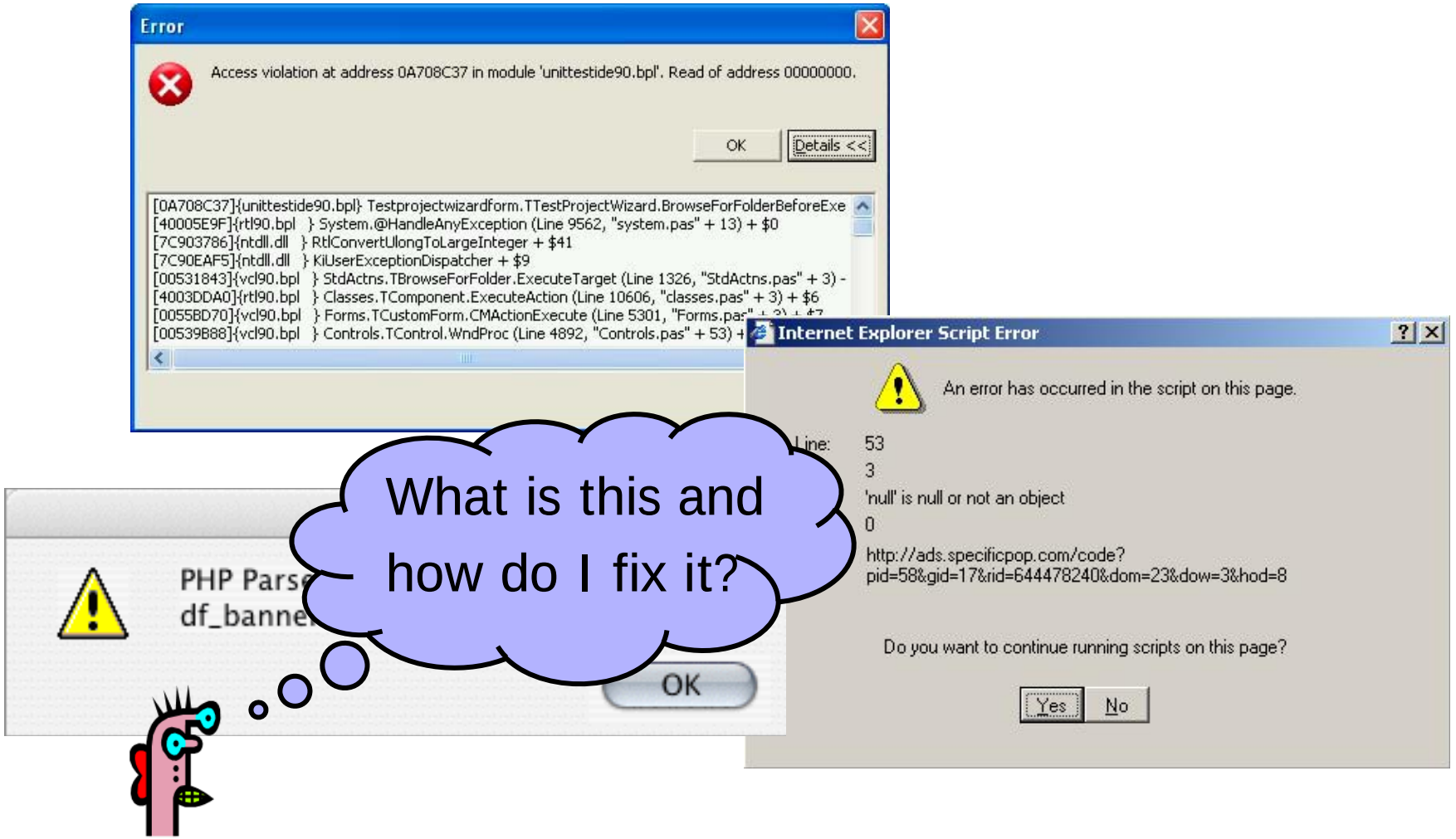
Secure Remote Diagnostics

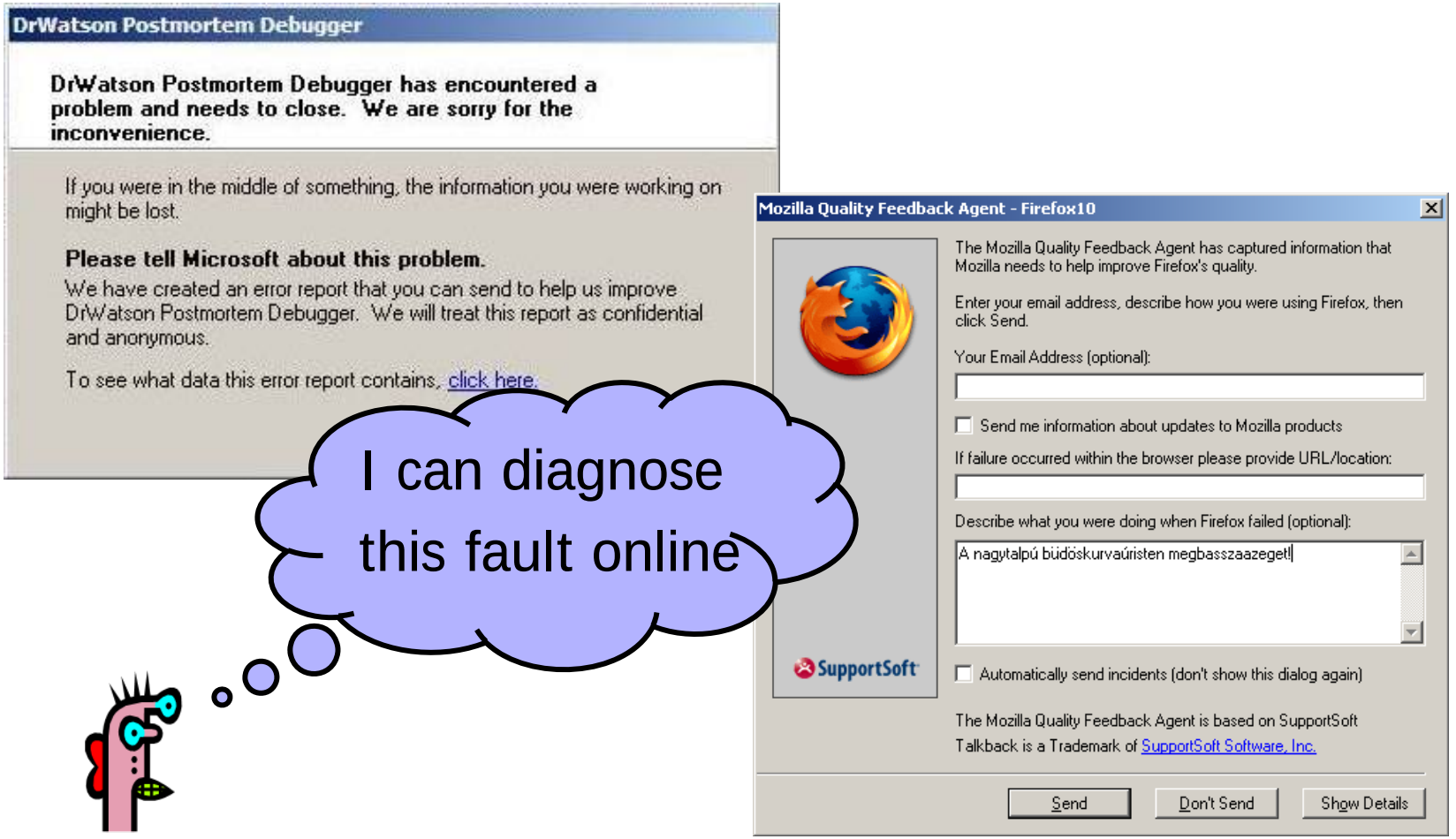
Justin Brickell
Donald E. Porter
Vitaly Shmatikov
Emmett Witchel

The University of Texas at Austin

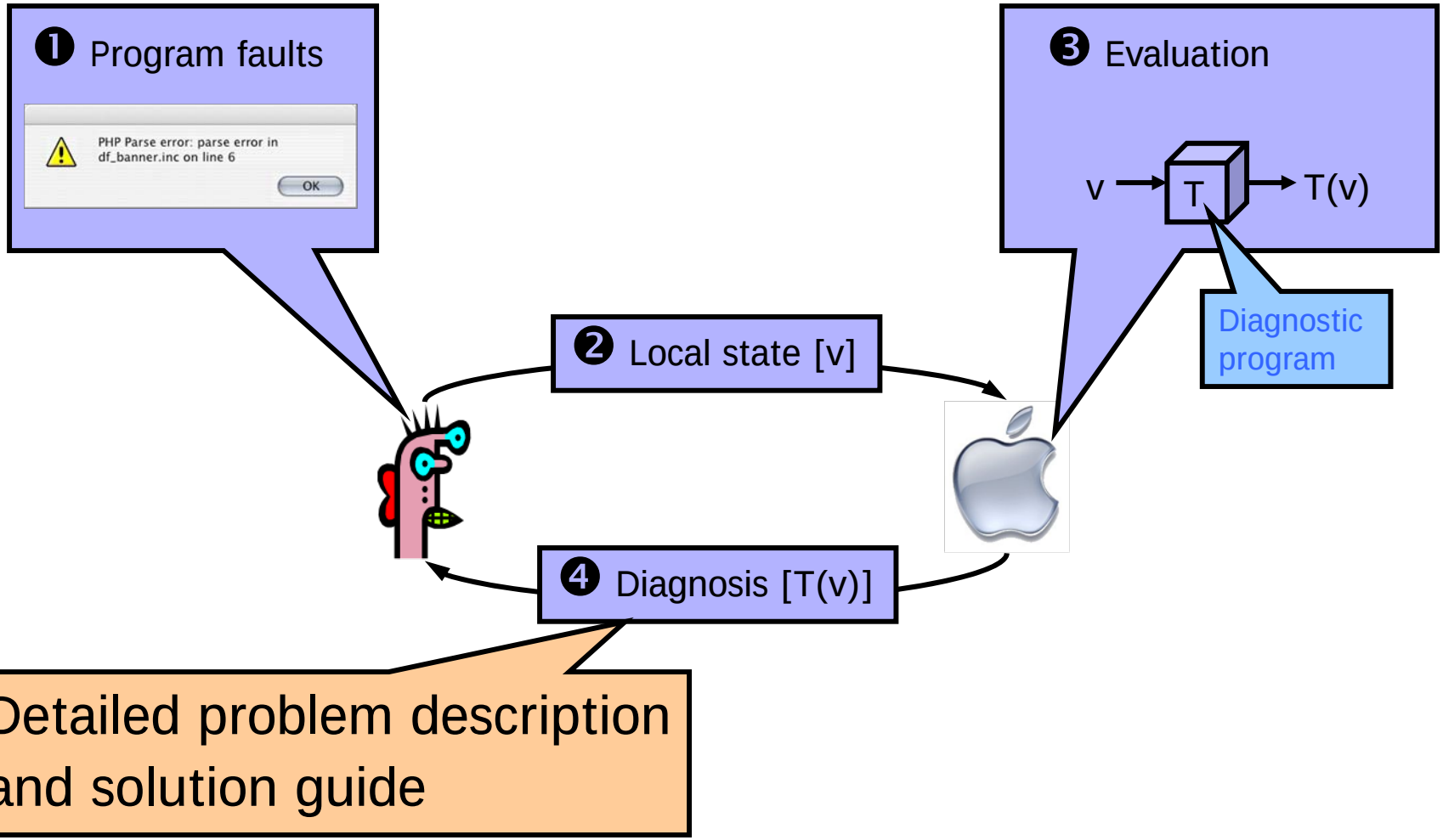
Error messages are cryptic



Troubleshooting is no longer local



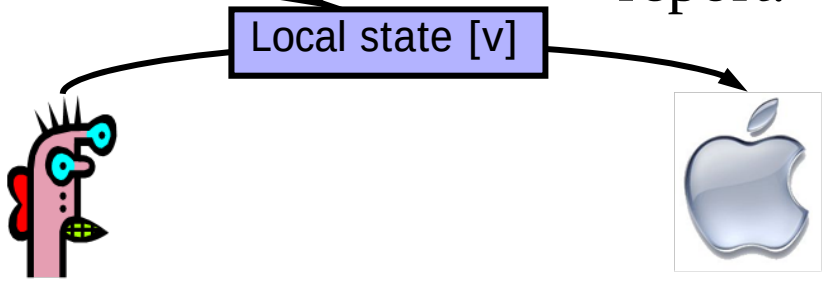
Software diagnostic scenario



Our goal: Protect user privacy

- Program memory
- File snippets
- Other running programs
- Confidential information

"If you are concerned that a report might contain personal or confidential information, you should not send the report."

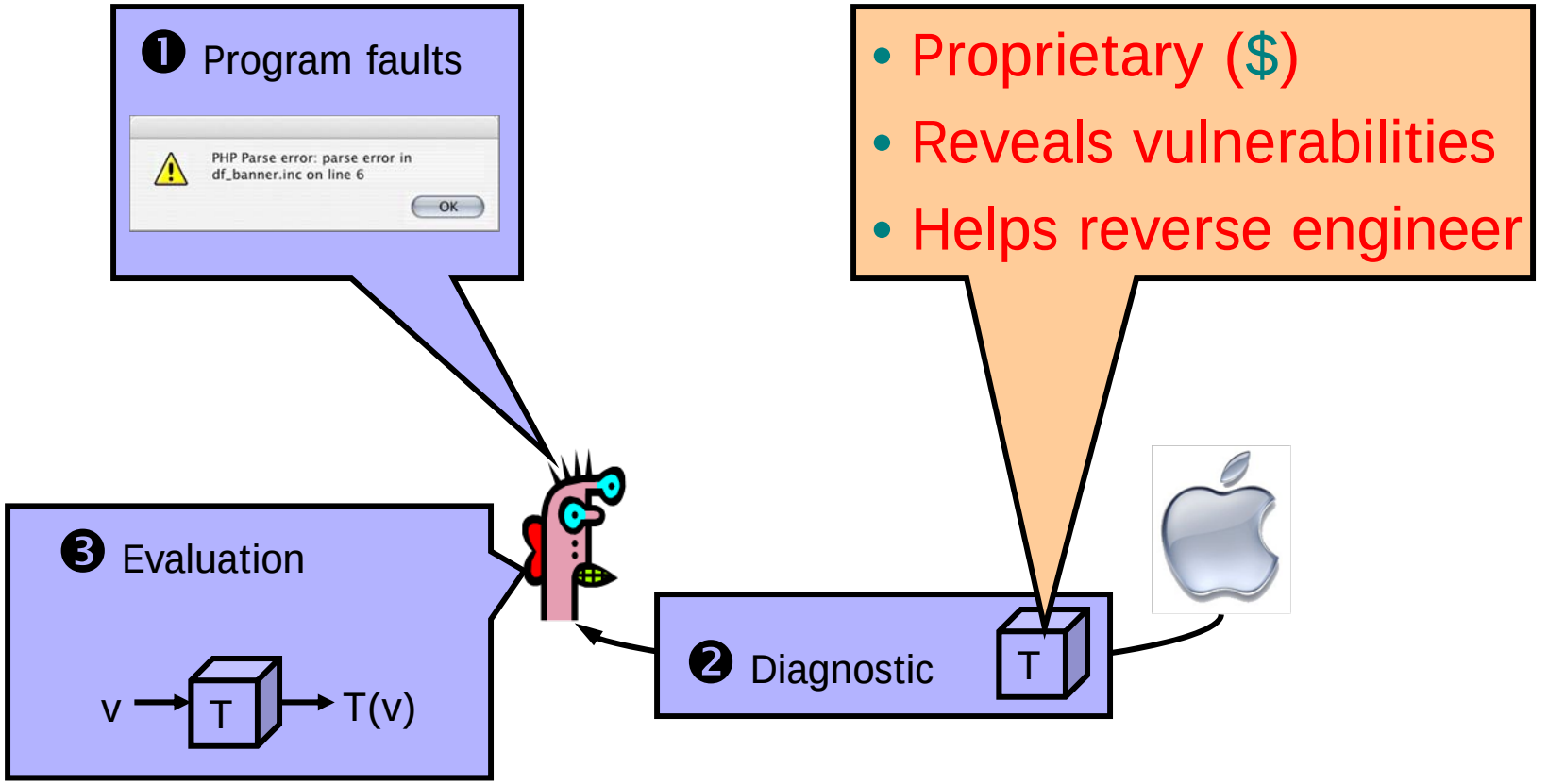


Problem: Users with privacy concerns cannot participate

Talk outline

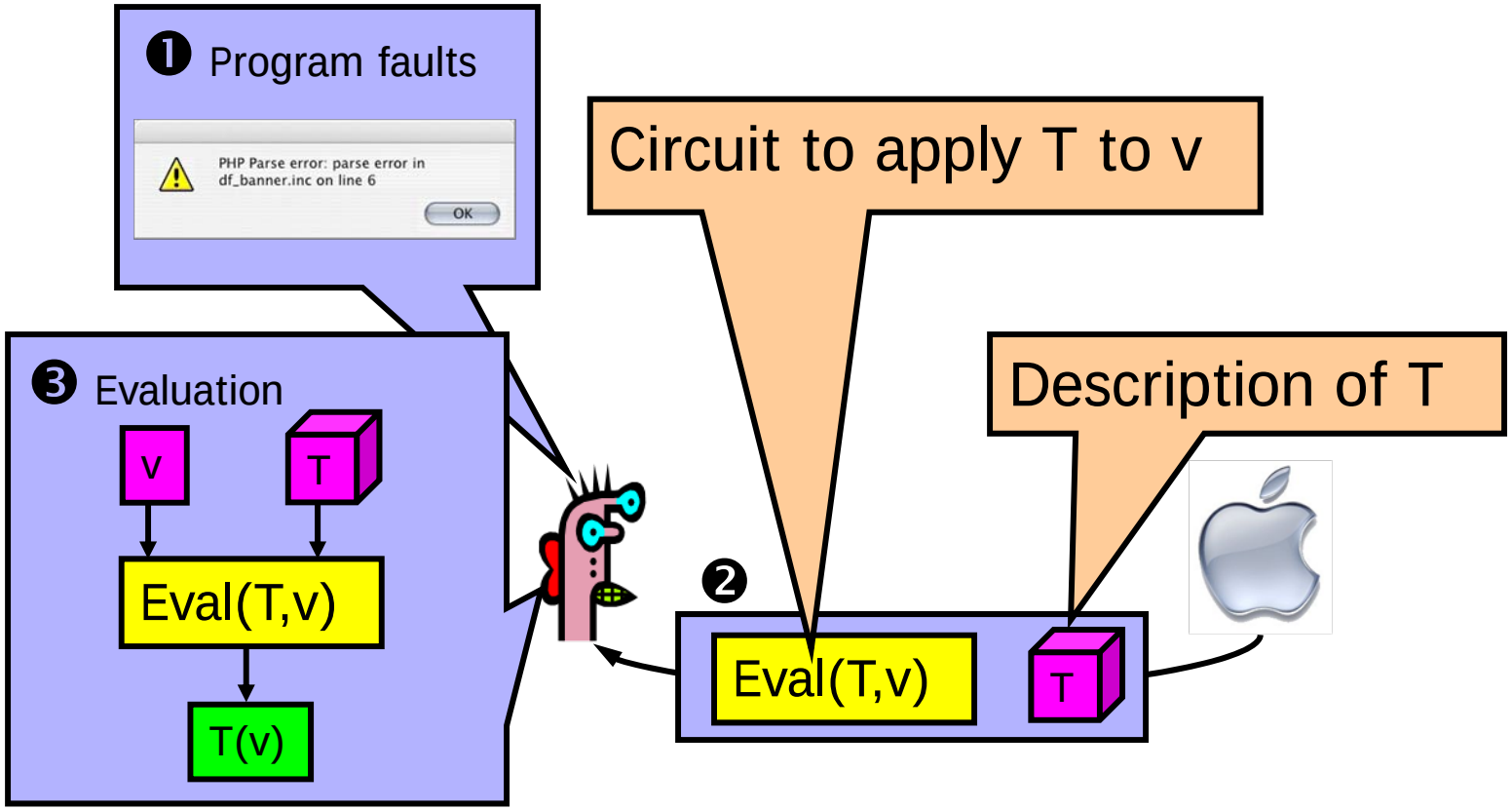
- Two unsatisfactory approaches
- Overview of our approach
- Building blocks
- Protocol walkthrough
- Applications
- Performance
- Conclusion

User-side evaluation



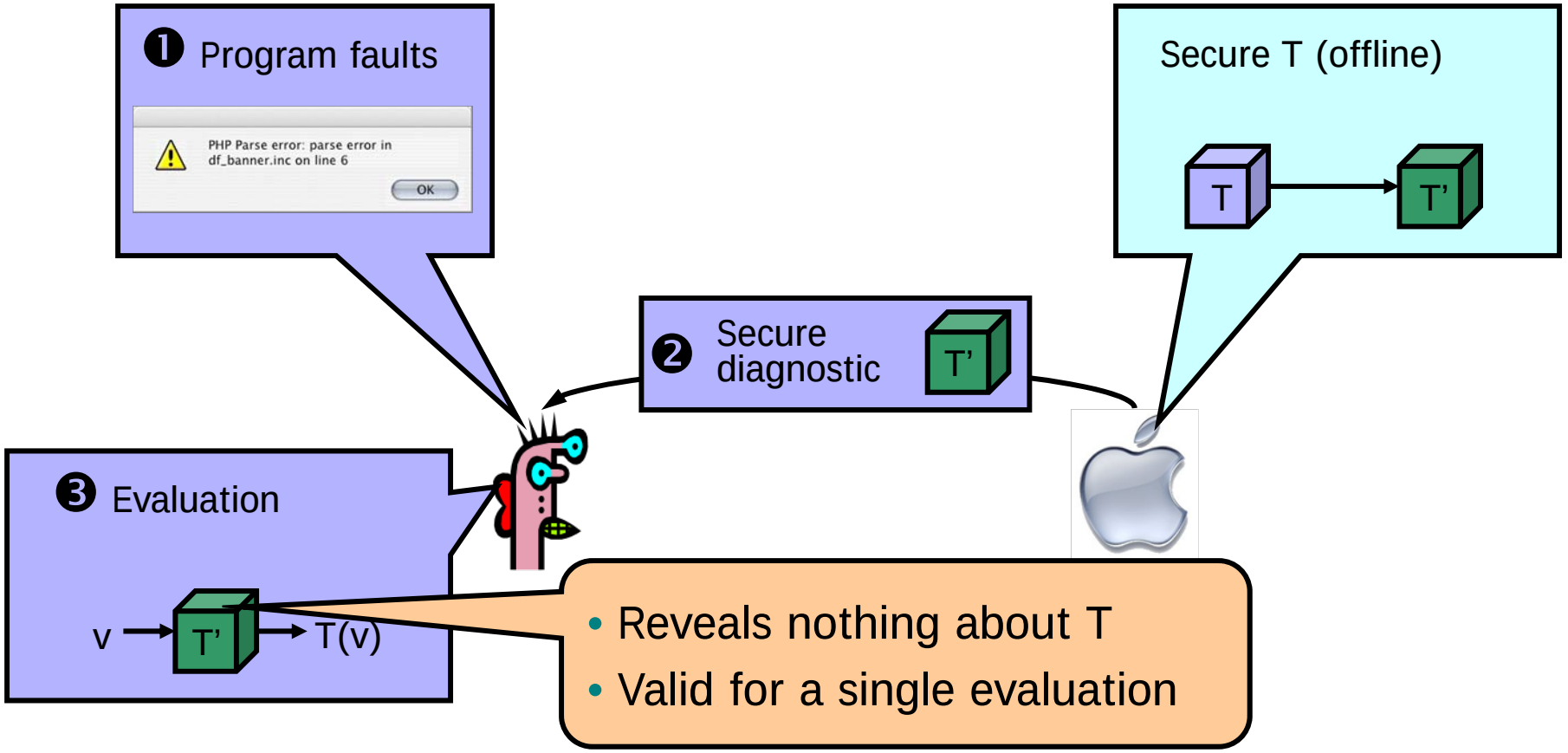
Problem: Vendor also has privacy concerns

Secure multiparty computation



Problem: circuit is too large to be practical

Our approach: Securing T



Goal: Build a practical system for real problems

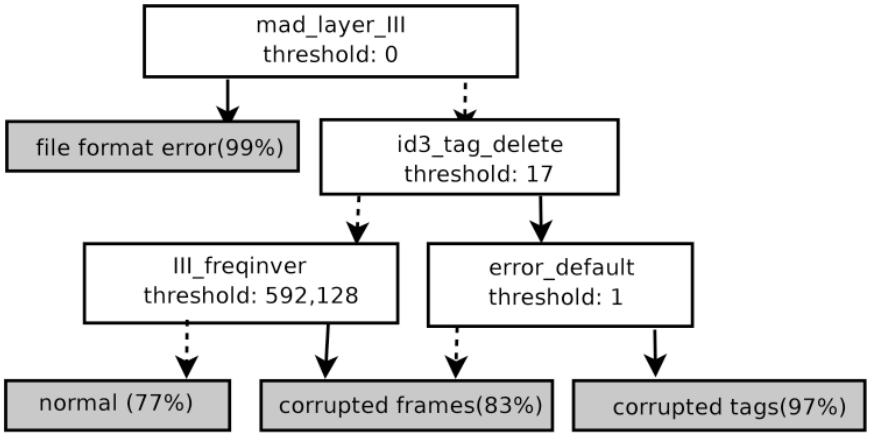
Local state

- A snapshot of the user's local state
- A vector of attribute values, such as
  - Function call counts
  - Function return values
  - Contents of memory locations

  ...

Diagnostic branching program

- Decision nodes compare an attribute value to a threshold
  - Control flows to either the left or right branch
- Classification nodes specify a label



Diagnostic branching program for mpg321

Aside: Creation of diagnostics

- Diagnostic programs are built in several ways
  - Hand-designed using expert knowledge
  - Automatically generated by data mining and analyzing user error reports
    - For example, Microsoft's Dr. Watson
- Clarify [PLDI '07]
  - Project at UT Austin to automatically build "black-box" classifiers for anomalous program behavior

Hiding the diagnostic program

Property to hide	Technique
Per node: thresholds and attributes	Private integer comparison (Yao's method)
Global: subset of attributes that are used	Homomorphic encryption
	Blinding
Global: program topology	Unevaluated nodes are hidden by encryption

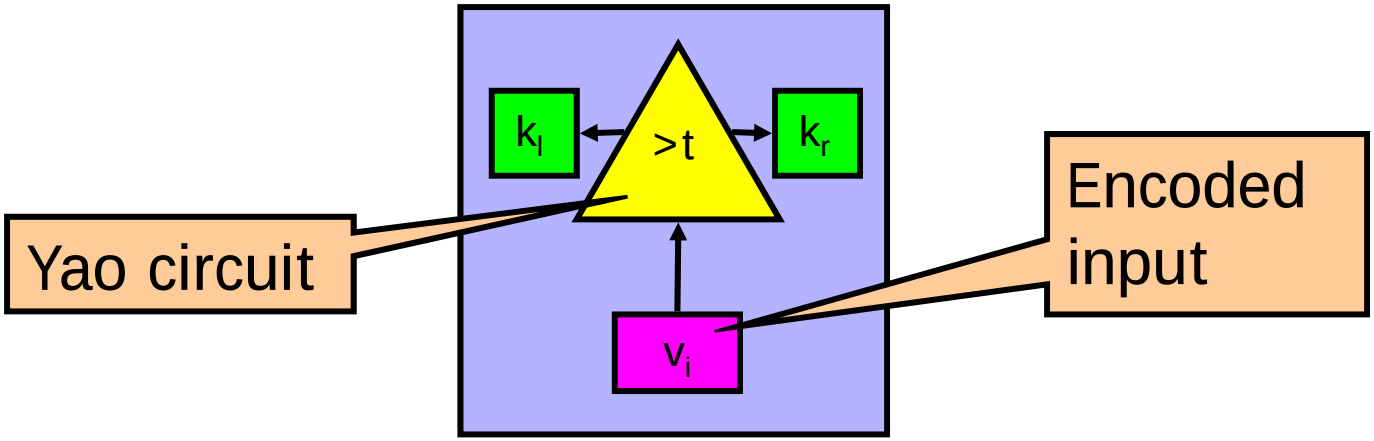
Hide everything about T

Private integer comparison

- We use Yao's method for integer comparison

  IF (v_i > threshold) THEN k_l ELSE k_r

  - Evaluator learns one of two keys conditional on comparison result
  - Evaluator doesn't learn comparison result or the threshold

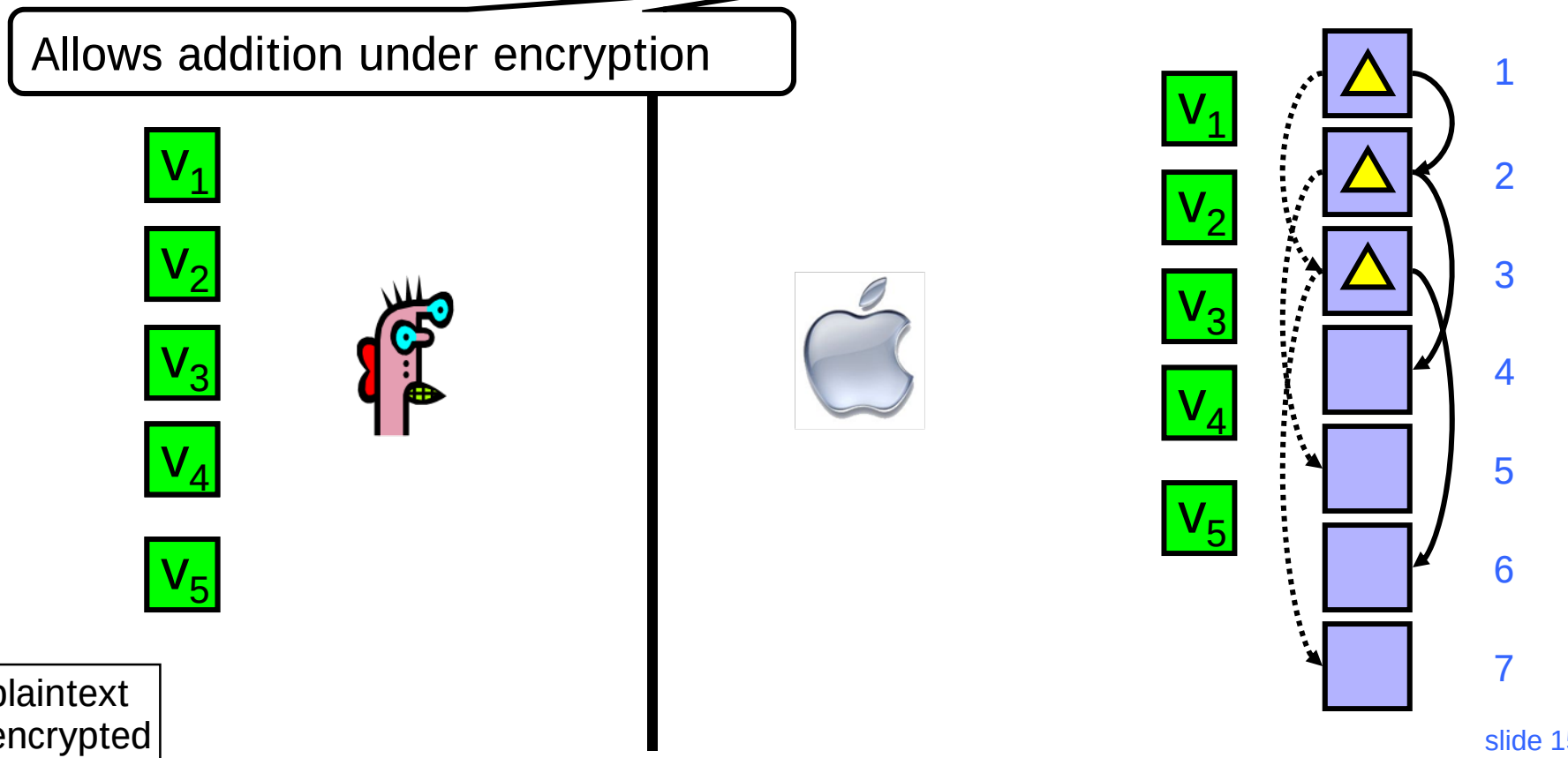


Protocol walkthrough

Hiding subset of attributes that are used

1. User encrypts attribute values using an instance of additively homomorphic encryption

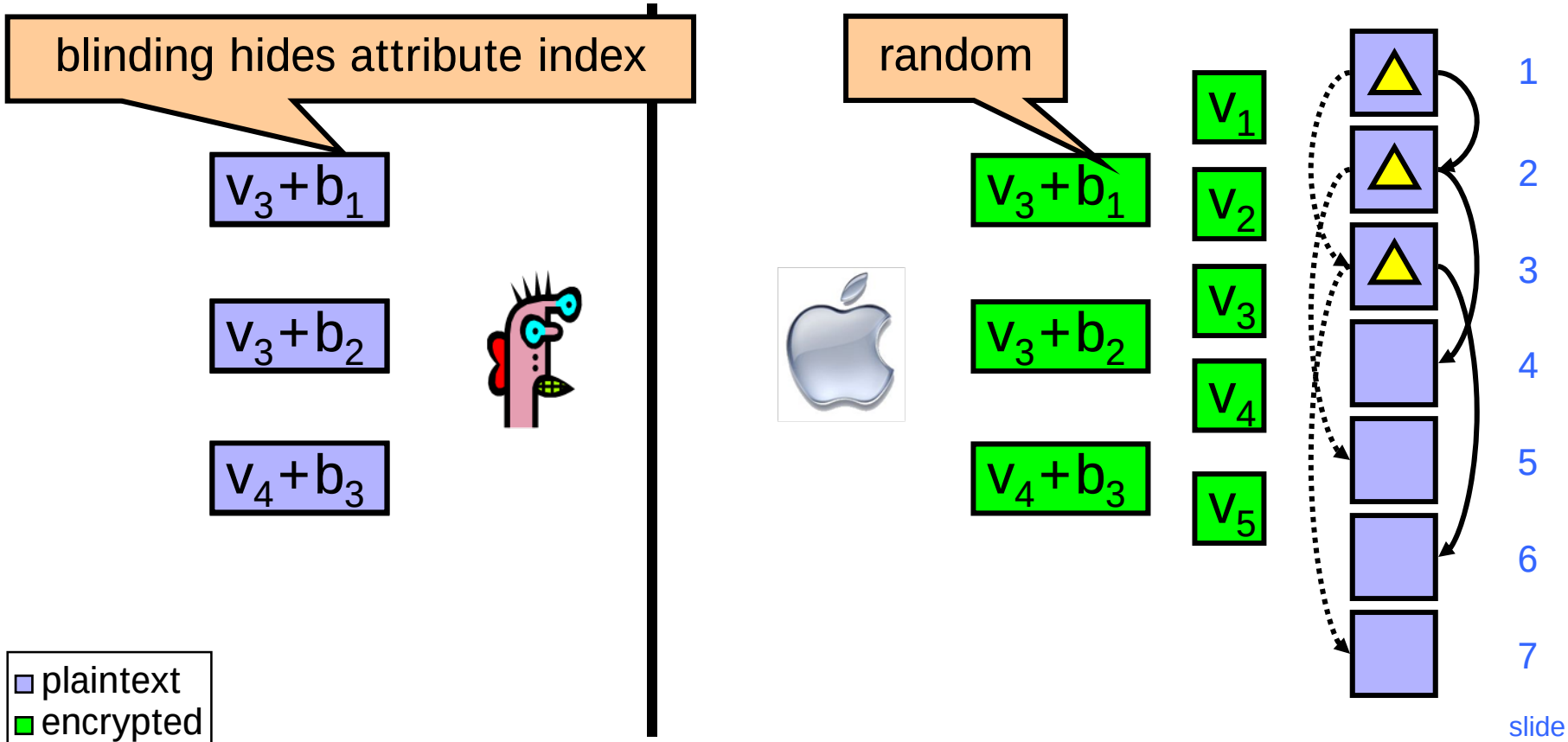
Allows addition under encryption



Protocol walkthrough

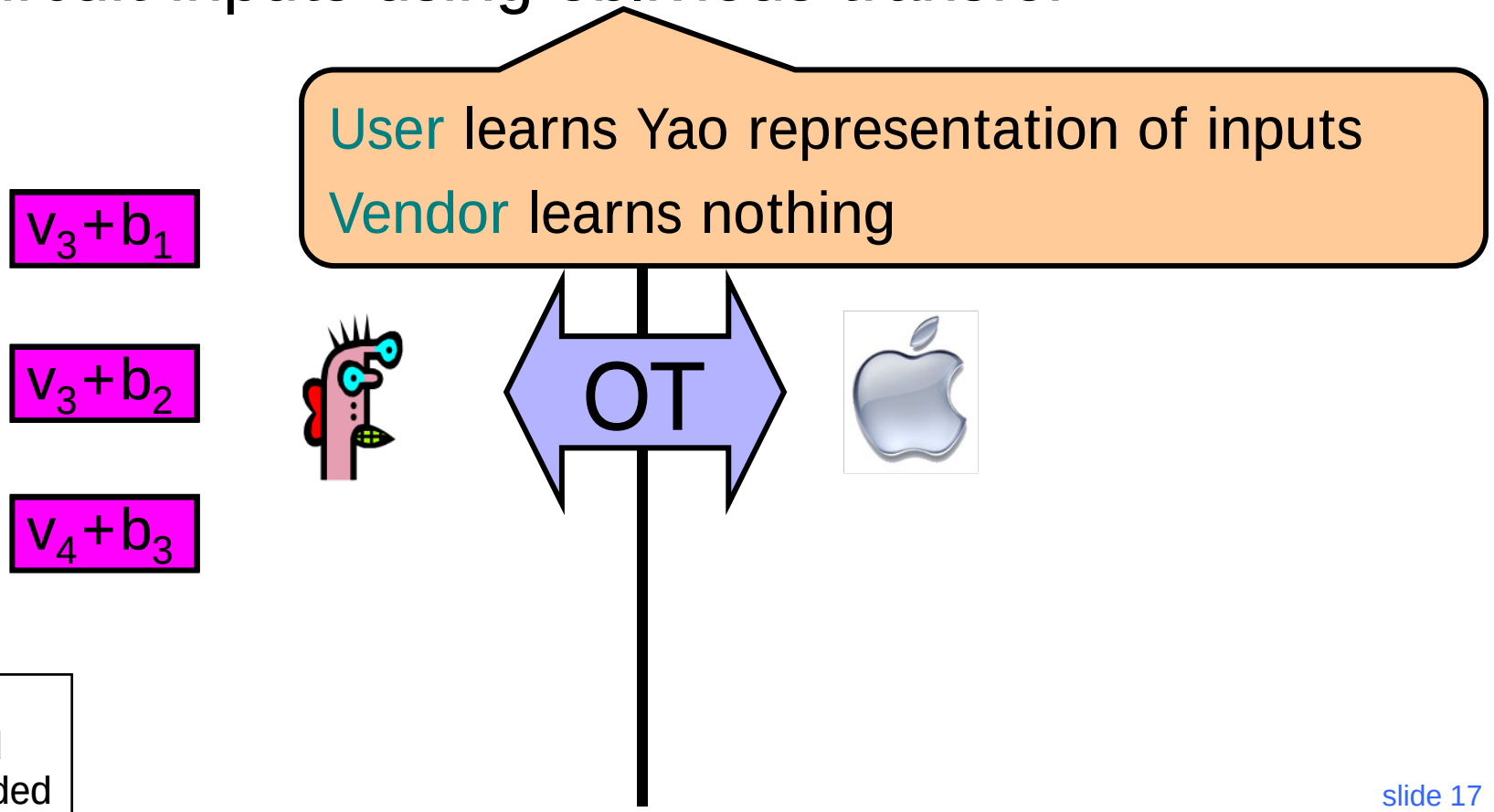
Hiding subset of attributes that are used

2. Vendor blinds attribute values under encryption for each decision node



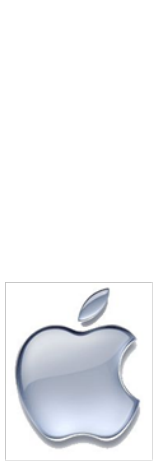
Protocol walkthrough

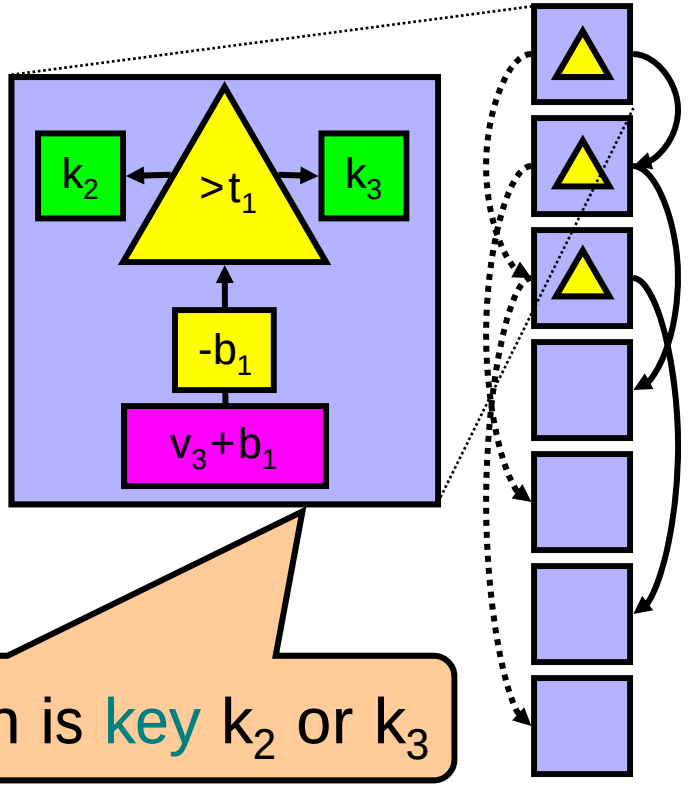
3. Blinded attribute values are converted to Yao circuit inputs using oblivious transfer



Protocol walkthrough

4. Vendor replaces decision nodes with secure integer comparison circuits (offline)





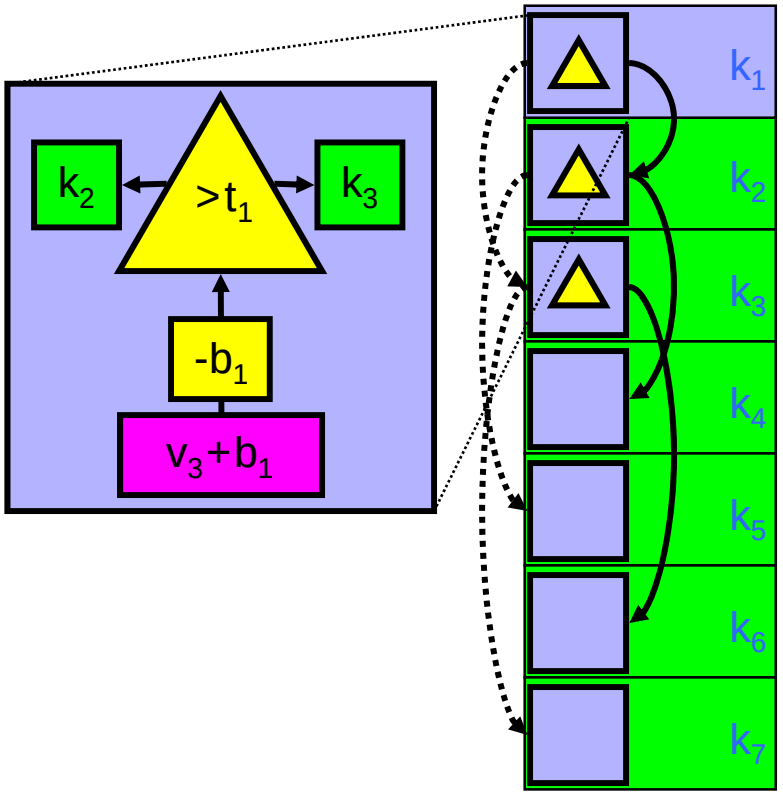
Output of comparison is key k_2 or k_3

plaintext
encrypted
Yao-encoded

Protocol walkthrough

Hiding topology of T

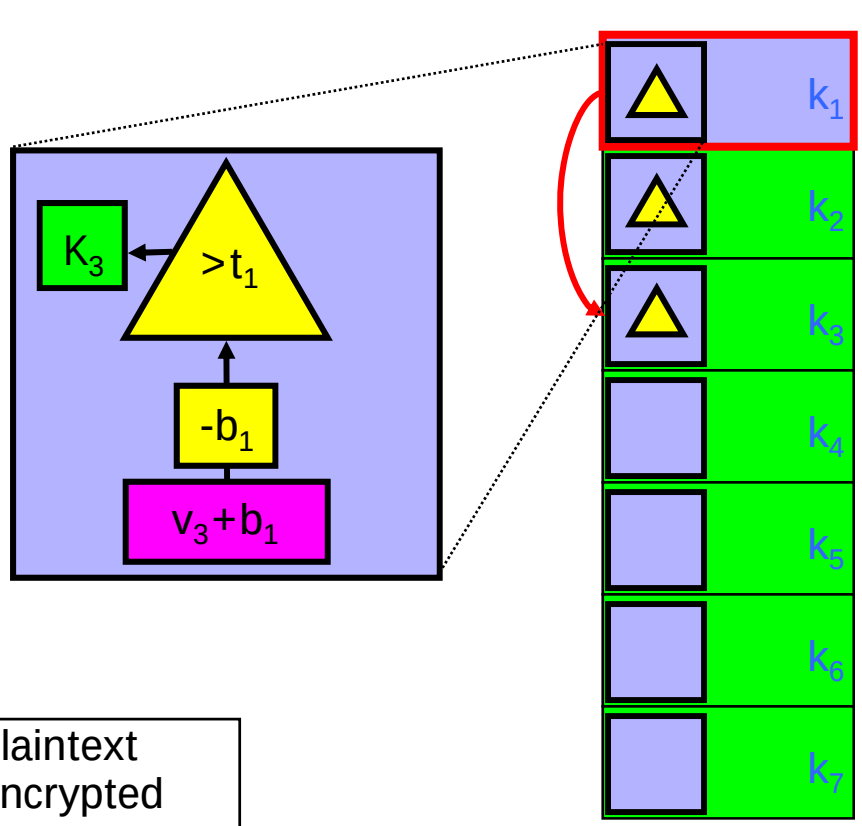
5. Vendor encrypts each node

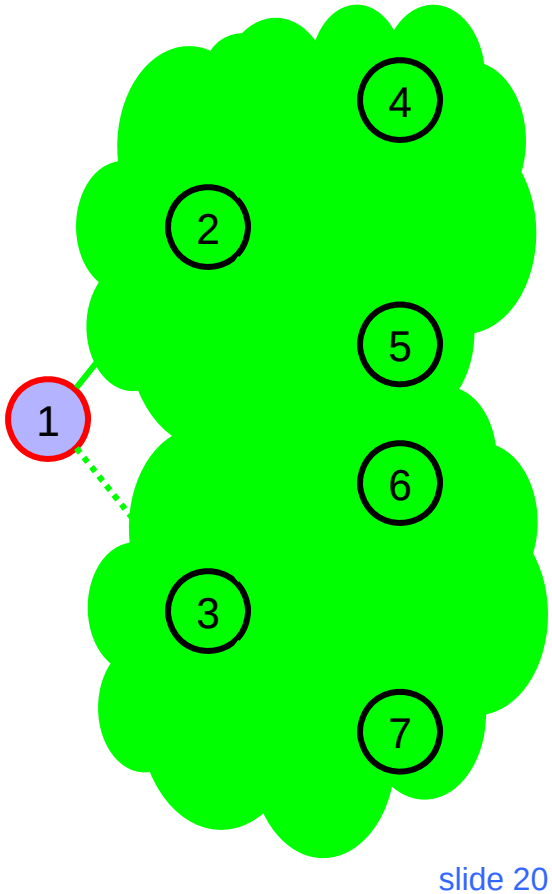


plaintext
encrypted
Yao-encoded

Protocol walkthrough

6. User evaluates encrypted branching program

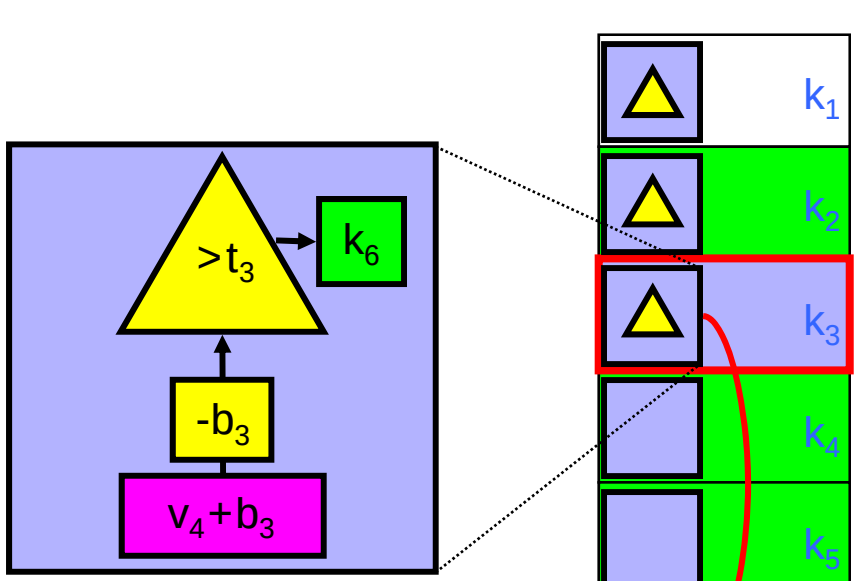


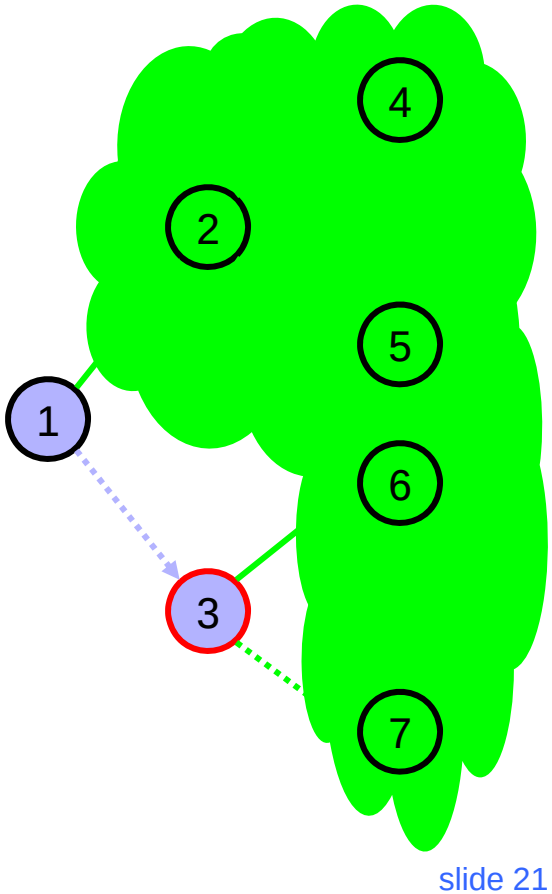


plaintext
encrypted
Yao-encoded

Protocol walkthrough

6. User evaluates encrypted branching program

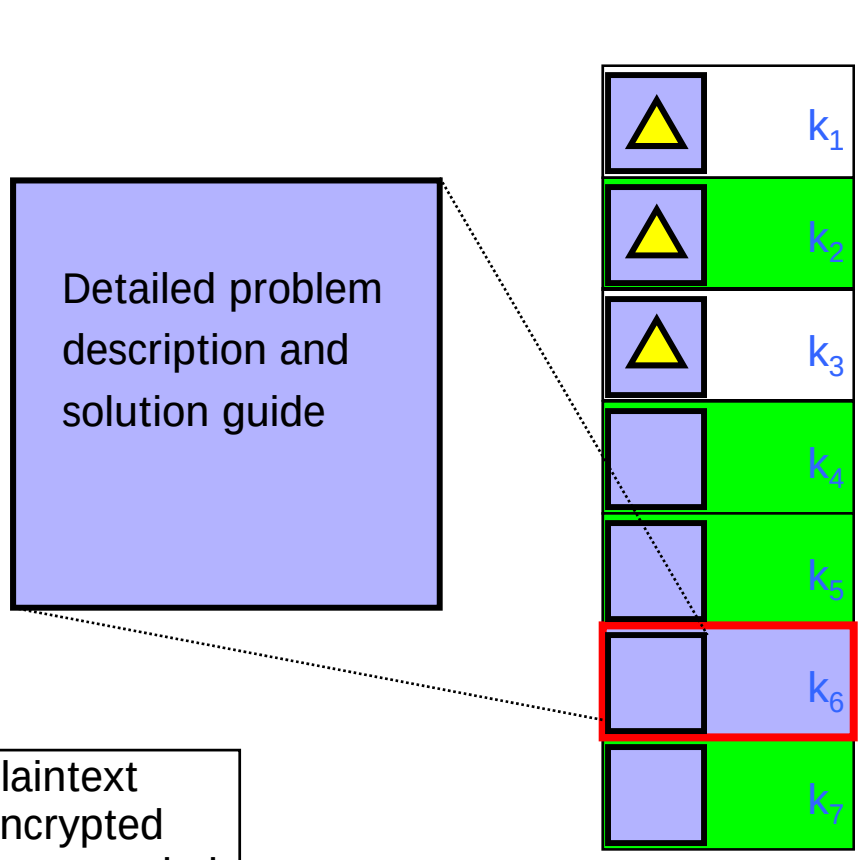


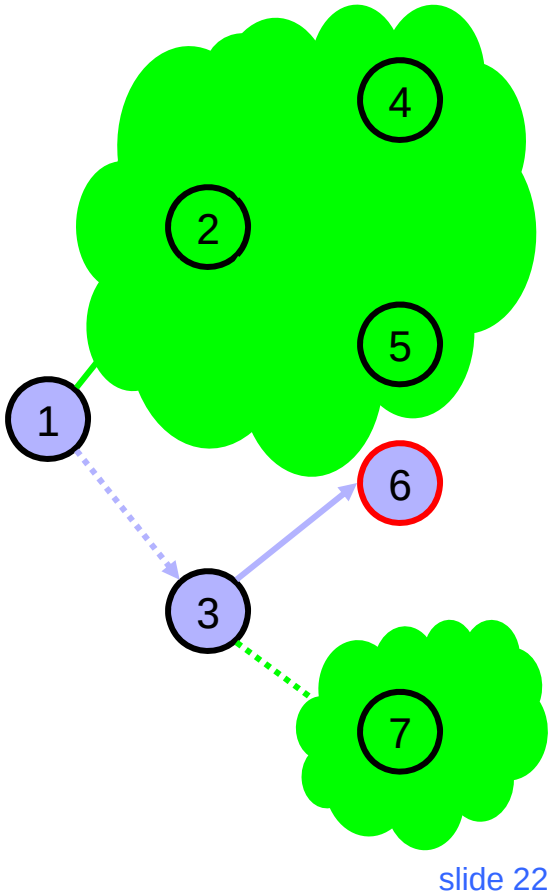


plaintext
encrypted
Yao-encoded

Protocol walkthrough

6. User evaluates encrypted branching program





Protocol generality

This is a general protocol

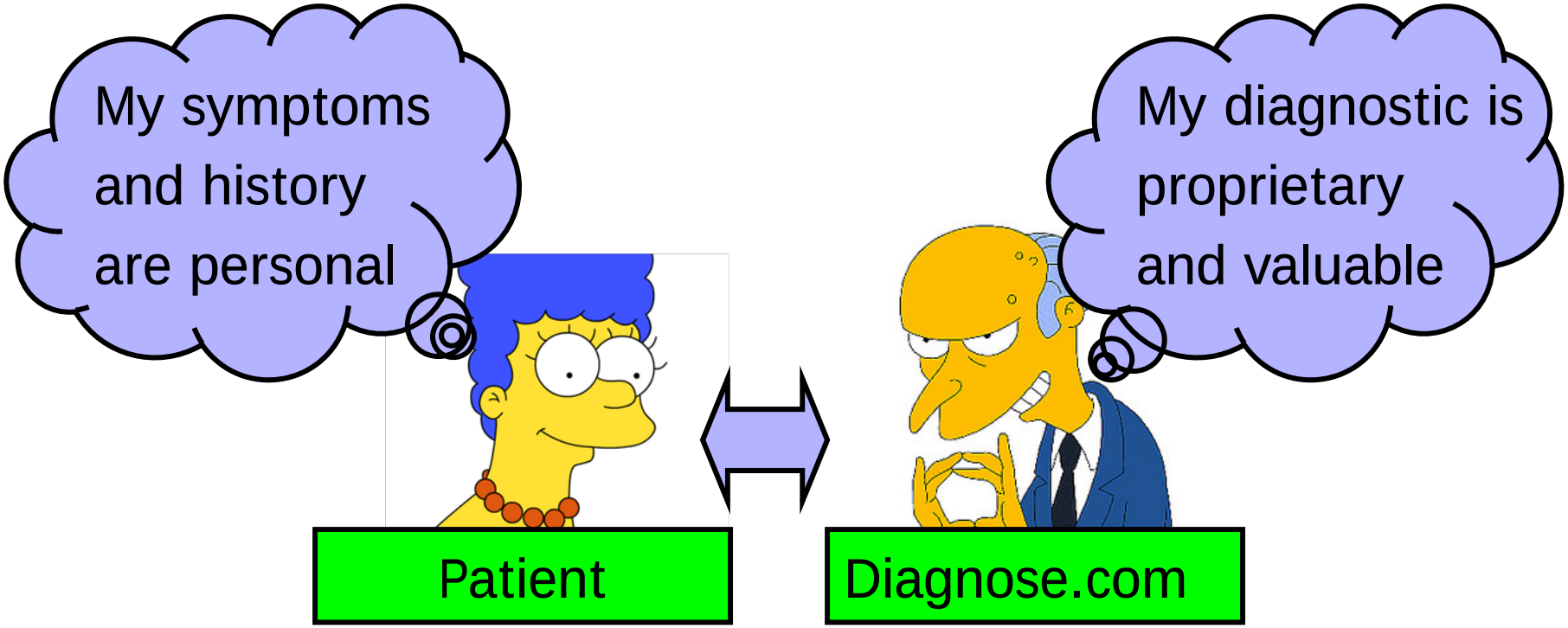
Any branching program, any attribute vector

Let’s look at some specific applications

Application: Software diagnostics

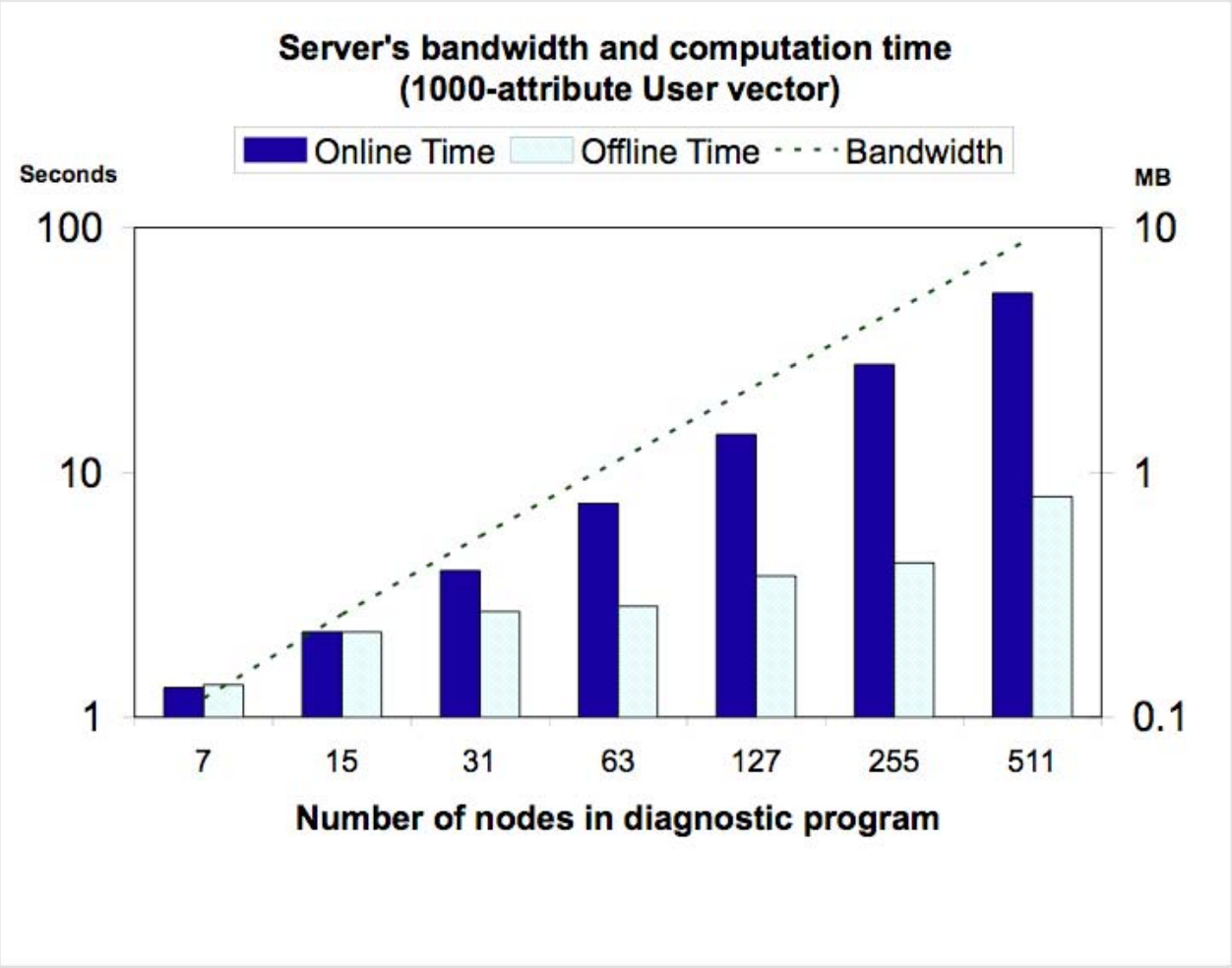
- Clarify [PLDI '07] automatically builds branching programs to classify anomalous program behaviors
- We have a working system to evaluate these branching programs securely
- The system is practical
  - For example, classifier for 4 cryptic gcc errors
    - 37 nodes, 2920 attributes
    - CPU: 5 seconds vendor, 7 seconds user
    - Bandwidth: 656kB vendor, 707kB user

Application: Medical diagnostics

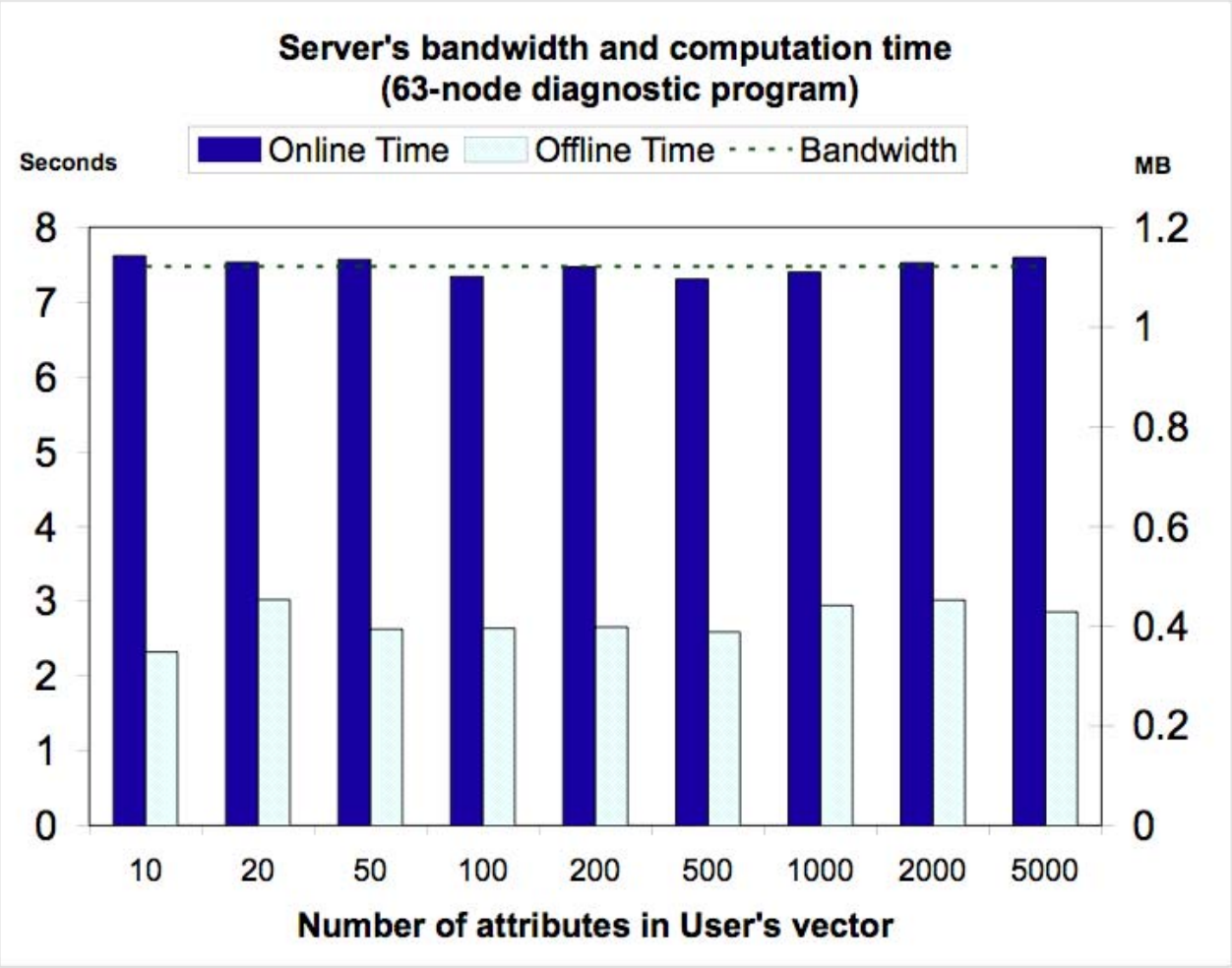


When the diagnostic T and the data v are both private, our tool can securely compute T(v)

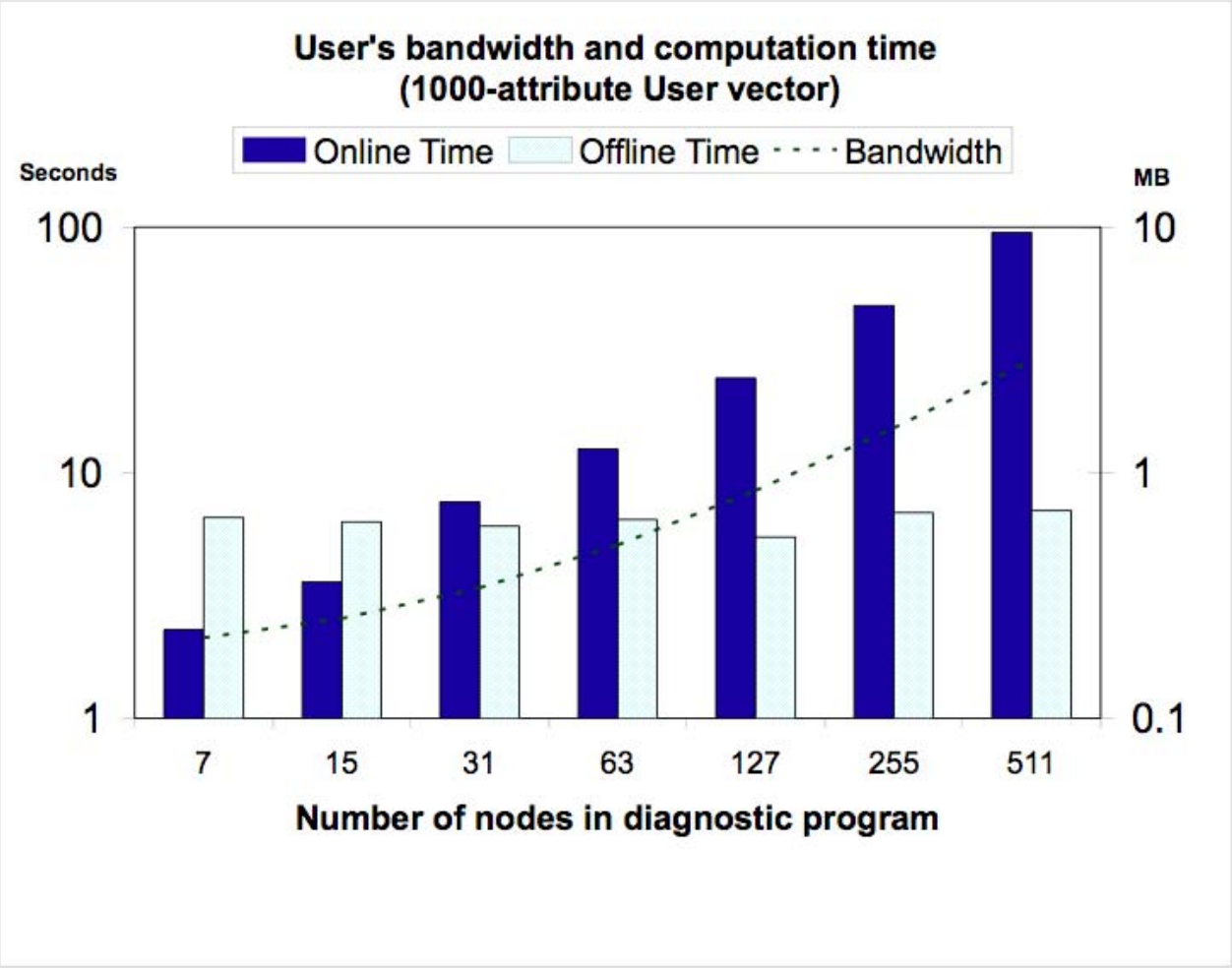
Server performance: Size of T



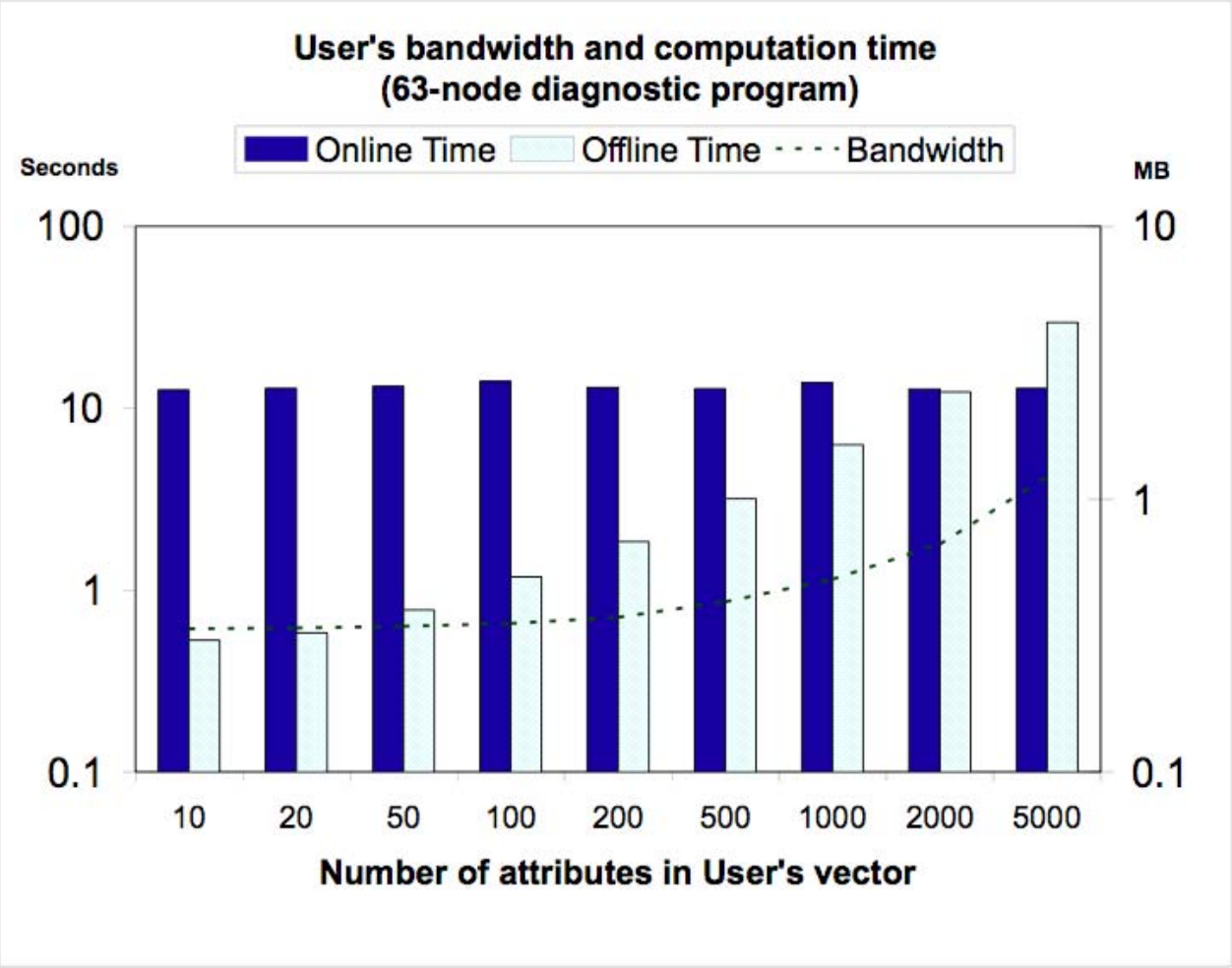
Server performance: Size of v



User performance: Size of T



User performance: Size of v



Conclusions

- Novel solution for secure branching program evaluation
- Provably secure
- Much more efficient than generic techniques
- Well-suited to software diagnostics
  - Online computation is independent of the size of local state
- Performance acceptable for real-world applications