CBMC-GC: Secure Two-Party Computations in ANSI C

Stefan Katzenbeisser
Security Engineering Group
Technische Universität Darmstadt & CASED
skatzenbeisser@acm.org

Joint work with:
Martin Franz (CASED), Andreas Holzer (TU Wien), Helmut Veith (TU Wien)
Data Privacy — The Traditional Approach

Data privacy relies on several technical and administrative approaches:

- Legal requirements
- Policies
- Audits
- Training
- Technical means (access control, network security, intrusion detection)
- Physical security
Does it work?

Restaurant chain customers' credit card data stolen

By Bruce Mohl, Globe Staff | October 12, 2008

Not Your Average Joe's, a Massachusetts chain, yesterday that thieves have stolen credit card information from customers.

The Dartmouth-based chain estimated that more than 3,500 of the 350,000 customers it served in August and September had their credit card information stolen. The 14-location chain said it is working with the US Secret Service and major credit card companies to determine how the data theft occurred and precisely how many customers were affected.

Today, the chain plans to post on its website a notice to customers about the security breach.

Diana Pisciotto, a spokeswoman for Not Your Average Joe's, said the chain decided not to tell customers it could check their credit card statements to notify them of any suspicious charges.

"We're doing this out of an abundance of caution and forthright with our customers," she said.

Stolen computer contained info from 88,000 patients at Staten Island hospital

by Staten Island Advance

Wednesday April 30, 2008, 4:37 PM

Computer equipment stolen from an administrative office in Clifton in December contained personal information from 88,000 patients that have been treated at Staten Island University Hospital.

After four months with no arrests, hospital administrators are just now beginning the process of sending out letters to patients whose names, Social Security and health insurance numbers were contained in computer files on a desktop computer and a backup hard drive stolen Dec. 29 from the hospital's finance office at 1 Edgewater Plaza.

"The hospital is in the process of issuing a notice to each patient involved in which one year of free credit monitoring will be offered," said a hospital statement, released by spokesmen.

Personal data of 600,000 on lost laptop

Michael Evans, Defence Editor

January 19, 2006

A junior Royal Navy officer is facing a court martial after a laptop containing the personal data of 600,000 people, including serving personnel and thousands of people who have shown an interest in a military career, was stolen from his car.

The loss of the laptop was considered to be so serious that Des Browne, the Defence Secretary, will make a statement to the Commons early next week.
Privacy-Enhancing Technologies (PETs)

- Strike a balance between data availability and privacy

- **Paradigm:** keep data encrypted, PETs **compute with encrypted data**

- **Privacy By Design:** Cryptographic protocols precisely limit amount of information available
  
  - Cryptographic tools are available!
    - Homomorphic encryption
    - Yao’s Garbled circuits
    - Customized protocols (private set intersection, ...)

![Diagram showing envelopes with keys and numbers]
Example: Private Face Detection

Projection → Distance computation → Find a match → Database

yes / no Bob
Example: Private Processing of Genome Data

Two-Party Computation

Physician → Two-Party Computation → Bioinformatics Institute

Medical Test
Nice approach, but is it ready for practice?

- Cryptographic protocols are ready, but tedious to use

- Lack of a good tool chain that a programmer can use

- Research prototypes are available:
  - Fairplay, FairplayMP, Sharemind, Tasty
  - Fast GC frameworks (implementation support for Java)

- We need “usable” compilers that helps a programmer implement PETs!
Recap:
Yao’s garbled circuits

AND

\[
\begin{array}{c|c|c}
\wedge & x = 1 & x = 0 \\
\hline
y = 1 & 1 & 0 \\
y = 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{align*}
y = 1 & : E(k_1^1, E(k_2^1, k_3^1)) \\
y = 0 & : E(k_1^0, E(k_2^1, k_3^0)) \\
\end{align*}
\]
Our choice as basis: Bit-precise Model Checker CBMC

- Transforms C program into a Boolean formula
- Transformation is “bit precise”
  - models evolution of program memory

- Bounded model checker:
  - Unrolls program up to a fixed number of loop iterations
  - Heuristics on how much unrolling is needed

- Boolean formula consists of program model and negated property
- SAT solver checks for solution

http://www.cprover.org
CBMC-GC

Central idea: use transformation from C code to SAT formula provided by CBMC for secure computing

ANSI C Program

Preprocessing:
- Loop unrolling
- Constant propagation
- Array handling

Circuit synthesis:
- Core CBMC engine „Placeholder“ gates

Circuit optimization:
- Instantiation of placeholder gates

Netlist of circuit
CBMC-GC: Example, Yao’s Millionaires

```c
void millionaires() {
    int INPUT_A_mila;
    int INPUT_B_milb;
    int OUTPUT_res;

    if (INPUT_A_mila > INPUT_B_milb)
        OUTPUT_res = 1;
    else
        OUTPUT_res = 0;
}
```

Local variables code inputs and outputs

Computations specified as C program
Matrix multiplication

```c
#define S 8 // size of matrices
int INPUT_A_a[S][S];
int INPUT_B_b[S][S];
int OUTPUT_c[S][S];
void multiply()
{
    int i, j, k;
    for (i = 0; i < S; i++)
        for (j = 0; j < S; j++)
            for (k = 0; k < S; k++)
                OUTPUT_c[i][j] += INPUT_A_a[i][k] * INPUT_B_b[k][j];
}
```

More complex data types like arrays, structs, enums

(Limited) support for pointer arithmetic
Limitations

CBMC-GC inherits limits from CBMC:

- **Bounded programs**: bounds for all loops must be known
  - in practice no problem
- No support for **floating point arithmetic**
- No support for **library functions** (yet)
- Limited **pointer arithmetic**
- **Integer data types** of fixed size
  - limits efficiency in secure computations
#define K 11 // length of array
#define MEDIAN 5 // position of median

int INPUT_A_a[K];
int OUTPUT_median;

void median_bubblesort() {
    int i, j, tmp, tmp1, tmp2;
    for (i = K - 1; i > 0; i--) {
        for (j = 0; j < i; j++) {
            tmp1 = INPUT_A_a[j]; tmp2 = INPUT_A_a[j + 1];
            if (tmp1 > tmp2) {
                INPUT_A_a[j] = tmp2; INPUT_A_a[j + 1] = tmp1;
            }
        }
    }
    OUTPUT_median = INPUT_A_a[MEDIAN];
}

CBMC can determine loop bounds by static analysis
CBMC-GC supports recursion
Example: Mergesort

```c
int b[K]; // temporary array for mergesort

void mergesort(int l, int r) {
    int i, j, k, m;

    if (r > l) {
        m = (r + l)/2; mergesort(l, m); mergesort(m + 1, r);
        for (i = m + 1; i > l; i--)
            b[i - 1] = INPUT_A_a[i - 1];
        for (j = m; j < r; j++)
            b[r + m - j] = INPUT_A_a[j + 1];

        for (k = l; k <= r; k++) {
            if (b[i] < b[j])
                INPUT_A_a[k] = b[i]; i++;
            else
                INPUT_A_a[k] = b[j]; j--;
        }
    }
}
```

Recursion; CBMC can determine bounds by static analysis
CBMC-GC: Optimizations

- Array access are slow
  - requires evaluation of a MUX circuit
  - remove some by static analysis

- CBMC is optimized for SAT performance
  - introduce placeholder gates
  - later instantiation with optimized circuits

- Optimization engine extensible

Preprocessing:
- Loop unrolling
- Constant propagation
- Array handling

Circuit synthesis:
- Core CBMC engine
  - "Placeholder" gates
Experimental results

We used CBMC-GC in conjunction with framework for execution of garbled circuits by Huang et al (USENIX 2011)

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Number of gates</th>
<th>Execution time, preprocessing</th>
<th>Execution time, circuit evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>3000 random arithmetic operations</td>
<td>2,298,441 (608,668)</td>
<td>970 ms</td>
<td>9,774 ms</td>
</tr>
<tr>
<td>8x8 matrix multiplication</td>
<td>3,257,345 (905,728)</td>
<td>680 ms</td>
<td>18,173 ms</td>
</tr>
<tr>
<td>Median, bubble sort, 31 elements</td>
<td>149,040 (45,120)</td>
<td>733 ms</td>
<td>1,644 ms</td>
</tr>
<tr>
<td>Median, merge sort, 31 elements</td>
<td>1,339,084 (436,916)</td>
<td>660 ms</td>
<td>3,790 ms</td>
</tr>
</tbody>
</table>
Conclusions

- Automatic compilation of two-party protocols is indeed possible
- Not as fast as hand-written code, but nevertheless usable in practice
- Will hopefully stimulate research in optimization issues, separates crypto functionality from compiler design
- Future research: other basic tools, other languages, optimizations, overcoming current limitations of CBMC-GC, ...
To come:

http://forsyte.at/software/cbmc-gc/