

Towards Practical Secure Computation

Handling sensitive data on untrusted machines



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Workshop on Trustworthy Clouds, Egham

Stefan Katzenbeisser

Security Engineering Group

Technische Universität Darmstadt & CASED

skatzenbeisser@acm.org



Joint work with:

Martin Franz (Deutsche Bank), **Andreas Holzer** (TU Wien),
Helmut Veith (TU Wien)

Handling private data: Does it work?



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Restaurant chain customers' credit card data stolen

The Boston Globe

By Bruce Mohl, Globe Staff | October 1, 2008

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

The Dartmouth-based chain estimated less than 3,500 of the 350,000 customers it served in August and September had their credit card information stolen. The 14-restaurant 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 Pisciotta, a spokeswoman for Not Your Average Joe's, said the chain decided to check their credit card statements with companies about any suspicious charges but not responsible for fraudulent activity on their cards.

"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 statement to the patient involved in which one year of financial records will be included in a hospital statement, released by spokeswoman said.

News Site of the Year | The 2008 Newspaper Awards

TIMES ONLINE

NEWS COMMENT BUSINESS SPORT LIFE & STYLE ARTS & ENTERTAINMENT RICH LIST

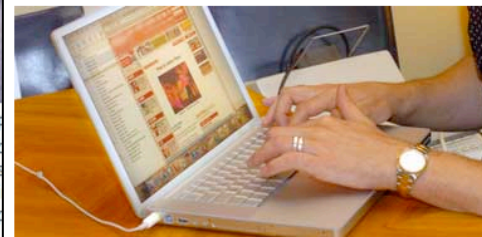
UK NEWS WORLD NEWS POLITICS ENVIRONMENT WEATHER TECH & WEB NEWS RELATED

Where am I? > Home > News > Politics

From The Times

January 19, 2008

Personal data of 600,000 on lost laptop



Michael Evans, Defence Editor

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.

TIMES RECOMMENDS

- > MPs back creation of human-animal embryos
- > Bank Holiday plan to celebrate Armed Forces
- > Wanted: criminal law expert to be new DPP

EXCLUSIVE EXTRACTS



Cherie autobiography

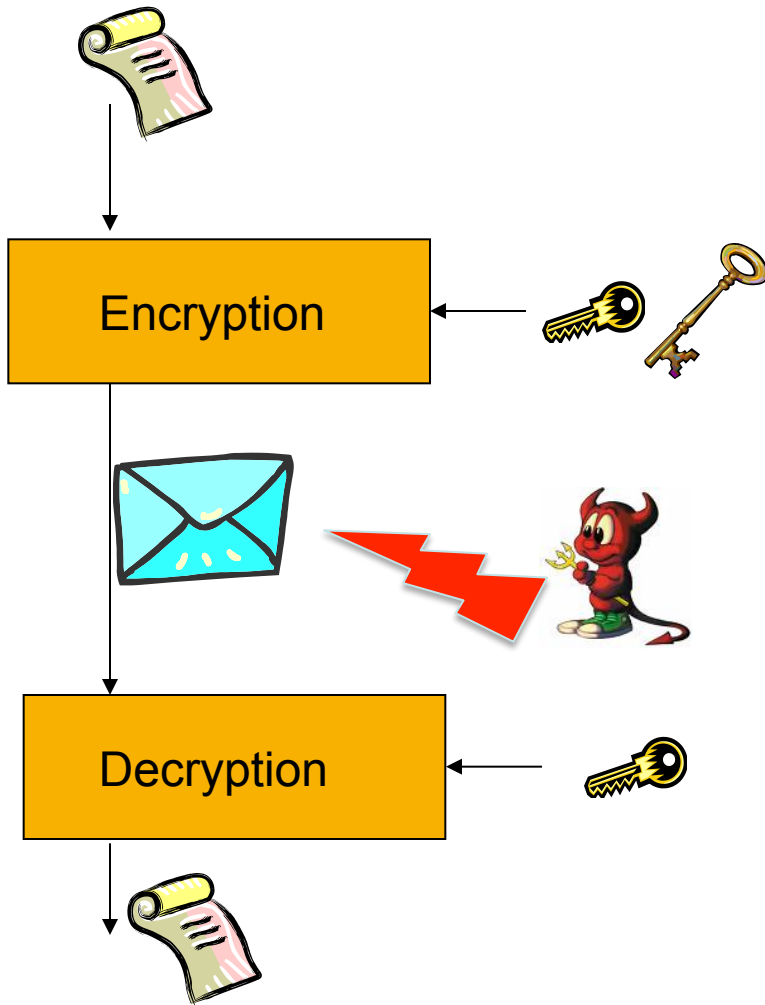
Full coverage and exclusive interviews and extracts from a decade in power

Computing with critical data: The traditional approach

Data secrecy relies on several technical and administrative approaches:

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

Cryptography (1)



Recta transpositionis tabula.

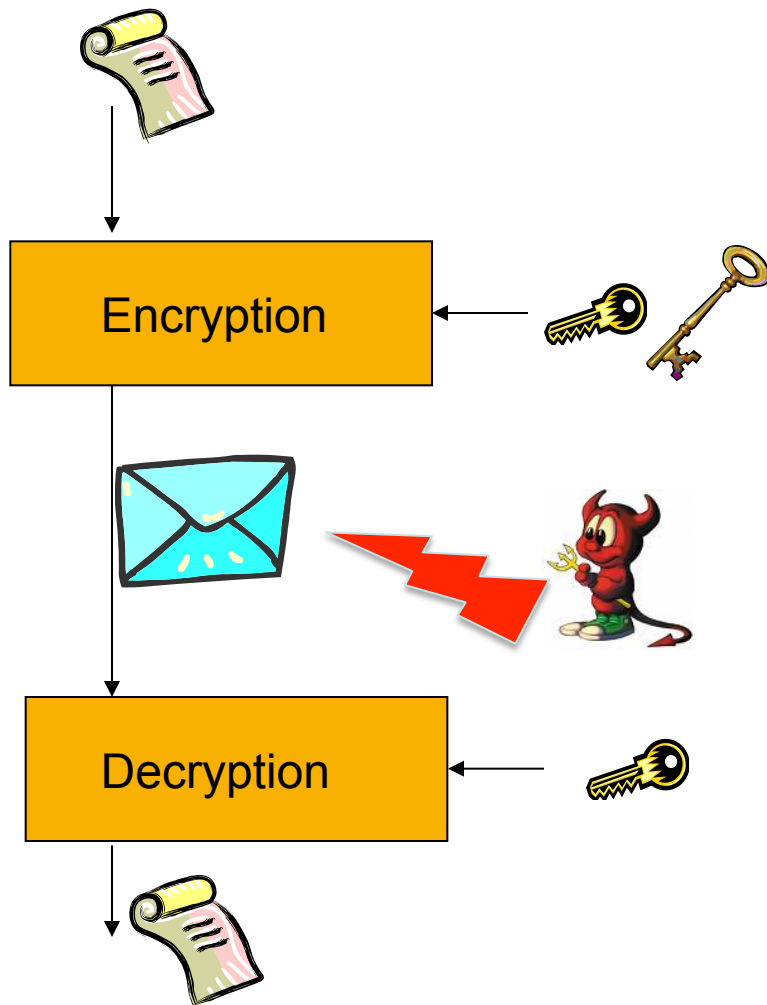
b	i	k	l	m	n	o	p	q	r	s	t	u	x	y	:	w	
i	k	l	m	n	o	p	q	r	s	t	u	x	y	3	w	a	b
k	l	m	n	o	p	q	r	s	t	u	x	y	3	w	a	b	c
l	m	n	o	p	q	r	s	t	u	x	y	3	w	a	b	c	d
m	n	o	p	q	r	s	t	u	x	y	3	w	a	b	c	d	e
n	o	p	q	r	s	t	u	x	y	3	w	a	b	c	d	e	f
o	p	q	r	s	t	u	x	y	3	w	a	b	c	d	e	f	g
p	q	r	s	t	u	x	y	3	w	a	b	c	d	e	f	g	b
q	r	s	t	u	x	y	3	w	a	b	c	d	e	f	g	b	i
r	s	t	u	x	y	3	w	a	b	c	d	e	f	g	b	i	k
s	t	u	x	y	3	w	a	b	c	d	e	f	g	b	i	k	l
t	u	x	y	3	w	a	b	c	d	e	f	g	b	i	k	l	m
u	x	y	3	w	a	b	c	d	e	f	g	b	i	k	l	m	n
x	y	3	w	a	b	c	d	e	f	g	b	i	k	l	m	n	o
y	3	w	a	b	c	d	e	f	g	b	i	k	l	m	n	o	p
3	w	a	b	c	d	e	f	g	b	i	k	l	m	n	o	p	q
w	a	b	c	d	e	f	g	b	i	k	l	m	n	o	p	q	r
a	b	c	d	e	f	g	b	i	k	l	m	n	o	p	q	r	
b	c	d	e	f	g	b	i	k	l	m	n	o	p	q	r		
c	d	e	f	g	b	i	k	l	m	n	o	p	q	r			
d	e	f	g	b	i	k	l	m	n	o	p	q	r				
e	f	g	b	i	k	l	m	n	o	p	q	r					
f	g	b	i	k	l	m	n	o	p	q	r						
g	b	i	k	l	m	n	o	p	q	r							
b	i	k	l	m	n	o	p	q	r								
i	k	l	m	n	o	p	q	r									
k	l	m	n	o	p	q	r										
l	m	n	o	p	q	r											
m	n	o	p	q	r												
n	o	p	q	r													
o	p	q	r														
p	q	r															
q	r																
r																	

x y 3 w a b c d e f g b i k l m n o p q r
 y 3 w a b c d e f g b i k l m n o p q r
 3 w a b c d e f g b i k l m n o p q r
 w a b c d e f g b i k l m n o p q r

In hac tabula literarū canonica siue recta
 latinarum literarum ipsarum per mutationem
 alphabeta, quot in ea per totum sunt mono
 & nigesies quatuor & uiginti, quae faciunt
 tidē multiplicata, paulo efficiunt minus q̄



Cryptography (2)

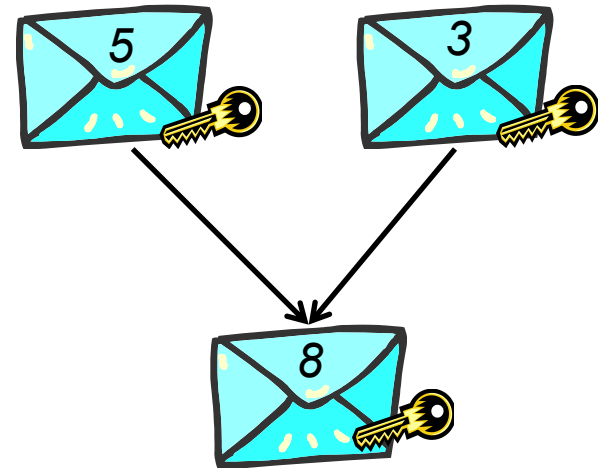


Doesn't cryptography solve the problem?

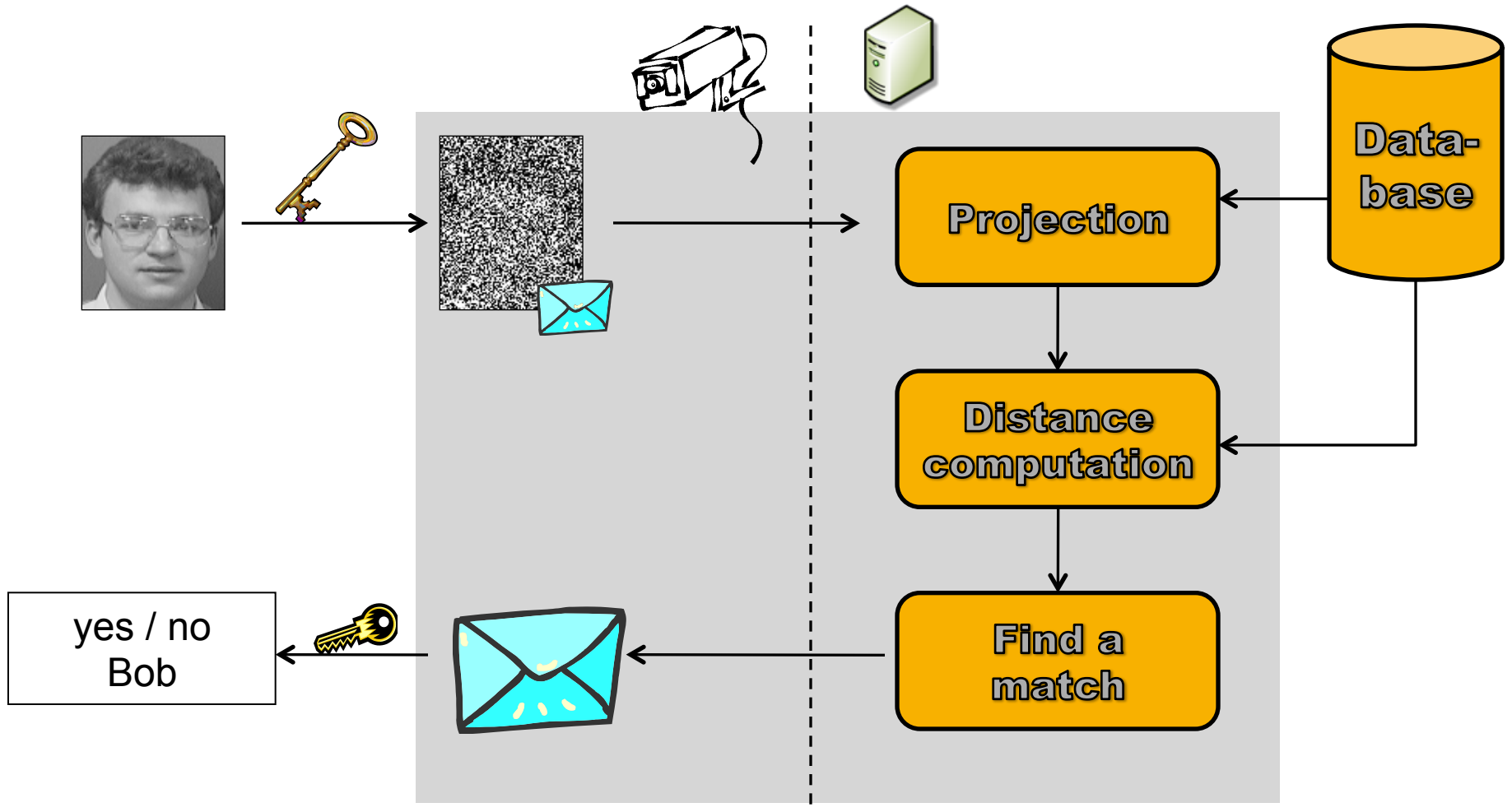
- Encrypted data „look random“
- ... but usually need to be decrypted before use.
- How can confidentiality be guaranteed during **use of data**?
- How can this be done on potentially compromised hosts?
- Protection of outsourced data?

Secure Computation

- Strike a balance between data availability and privacy
- **Paradigm:** keep data encrypted, **compute with encrypted data**
- Follows the principle of **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, ...)



Example: Private Face Detection



Example: Private Processing of Genome Data

Physician



Genom O



$[\text{Prob}[O | \lambda]]$

$$\begin{aligned}\alpha_1(i) &= \pi_i b_i(o_1) \\ \alpha_{t+1}(i) &= b_i(o_{t+1}) \sum_{j=1}^N a_{i,j} \alpha_t(j) \\ \text{Prob}[O | \lambda] &= \sum_{i=1}^N \alpha_T(i)\end{aligned}$$

Bioinformatics Institute

HMM



Cryptographic Tools: Secure Multi-Party Computation (1)

- N parties have secret inputs x_1, x_2, \dots, x_N
 - **Goal:** jointly compute $f(x_1, x_2, \dots, x_N)$ so that the private inputs of the parties are not revealed (in addition to what leaked through the function and the shared result)
 - Special case: **Secure Two-Party Computation** for $N = 2$
 - Yao (1982): Every computable function *can be computed securely in the two-party case*
-

Cryptographic Tools: Secure Multi-Party Computation (2)

- Security notions for secure multiparty computations are non-trivial
- At first sight, security may depend on the function
- Some functions may *inevitably* leak information, since result of computation is known to all parties

$$f(x, y) = y$$

- SMP hides only those parts of the input *that cannot be derived* from the shared function value
- Different solutions:
 - Yao's Garbled Circuits
 - Homomorphic encryption
 - Secret sharing

Millionaire's problem

- Two millionaires want to compare their wealth
- Both are “rich” ...
... both want to know who is “richer” ...
... but do not want to disclose wealth to each other

- Simple instance of function to be computed in a secret way:

$$f(x, y) := \begin{cases} 1 & x \geq y \\ 0 & \text{otherwise} \end{cases}$$

- *Communist Millionaire's Problem*: both parties want to check whether they are equally rich:

$$f(x, y) := \begin{cases} 1 & x = y \\ 0 & \text{otherwise} \end{cases}$$

Cryptographic Tools: Secure Multi-Party Computation (3)

Attacker Models:

- ***Semi-Honest Attacker***

- “Honest But Curious”
- Conforms to the protocol specification

- ***Malicious Attacker***

- May deviate from the protocol by computing values dishonestly
 - Much more difficult to cope with
 - May require Zero-Knowledge Proofs to show validity of computation
-

The “Holy Grail”?

Fully Homomorphic Encryption



TECHNISCHE
UNIVERSITÄT
DARMSTADT

eWEEK®

COMARCH CRM & MARKETING Mobil - Interak

MOBILE CLOUD SECURITY STORAGE ENTERPRISE APPS INNOVATION

HOT TOPICS: Android Apple IT Management New Era Networks Slide Shows More

Security / IBM Discovers Encryption Scheme That Could Improve Cloud Security, Spam Fl...

IBM Discovers Encryption Scheme That Could Improve Cloud Security, Spam Filtering

By Brian Prince | Posted 2009-06-25 [Email](#) [Print](#)

[f Share](#) 0 [t Tweet](#) 0 [+ Google +](#) 0 [in Share](#) 0 [f Like](#) 3 [f Recommend](#) 3

A researcher at IBM reports having developed a fully homomorphic encryption scheme that allows data to be manipulated without being exposed. Researcher Craig Gentry's discovery could prove to be important in securing cloud computing environments and fighting encrypted spam.

An IBM researcher has uncovered a way to analyze data while it is still encrypted, in what could be a boon for both spam-filtering applications and cloud computing environments.

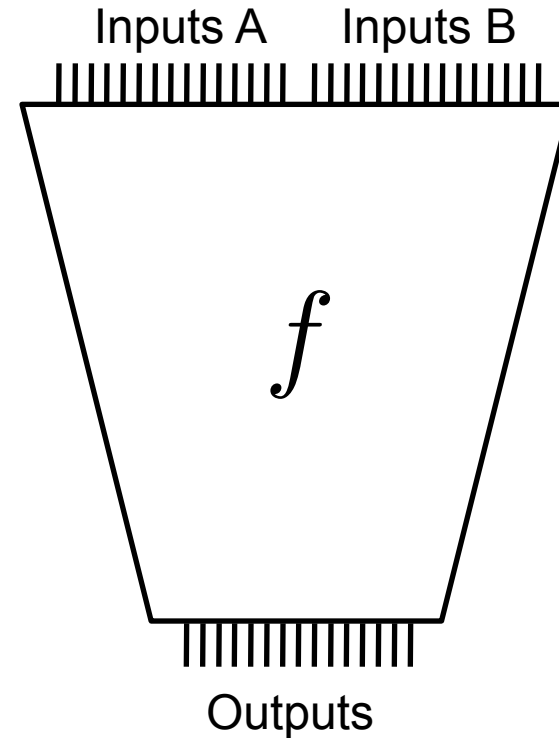
The challenge of manipulating data without exposing it has bugged cryptographers for decades. But in a breakthrough, IBM researcher and Stanford University Ph.D. candidate Craig Gentry has developed a "fully homomorphic encryption" scheme that keeps data protected.

Secure Two-Party Computation

Yao's Garbled Circuits (1)

“Garbled” circuit construction:

1. Represent function to be computed as Boolean circuit

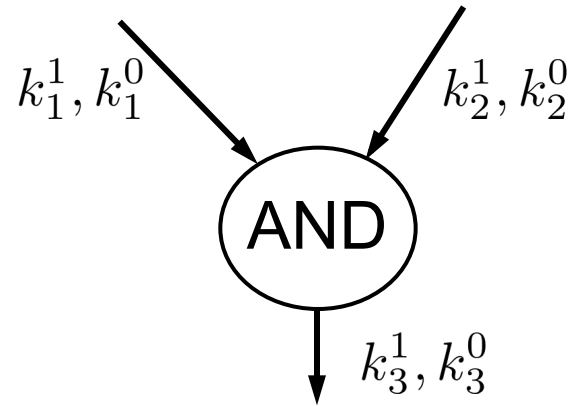


Secure Two-Party Computation

Yao's Garbled Circuits (2)

“Garbled” circuit construction:

1. Represent function to be computed as Boolean circuit
2. Choose 2 keys for each “wire” in the circuit
3. For each gate in the circuit, construct a “garbled” version (encrypted & permuted)
4. Output wires from output gates are not garbled



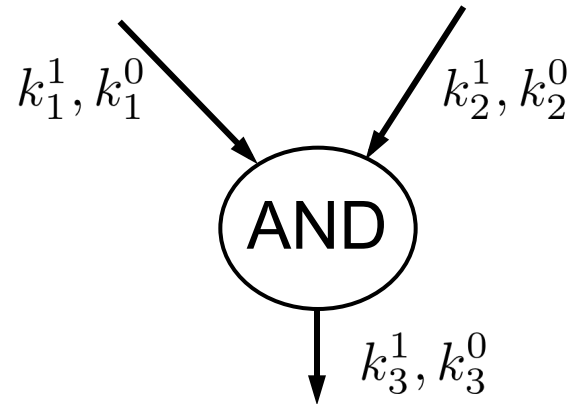
\wedge	$x = 1$	$x = 0$
$y = 1$	1	0
$y = 0$	0	0

	$x = 1$	$x = 0$
$y = 1$	$E(k_1^1, E(k_2^1, k_3^1))$	$E(k_1^1, E(k_2^0, k_3^0))$
$y = 0$	$E(k_1^0, E(k_2^1, k_3^0))$	$E(k_1^0, E(k_2^0, k_3^0))$

Secure Two-Party Computation

Yao's Garbled Circuits (3)

- Party A generates garbled circuits and gives it to B , including the “input keys” corresponding to its own input
- Party B runs (together with A) “Oblivious Transfer” to obtain keys corresponding to B 's input
- Party B evaluates the circuit, announces the result



\wedge	$x = 1$	$x = 0$
$y = 1$	1	0
$y = 0$	0	0

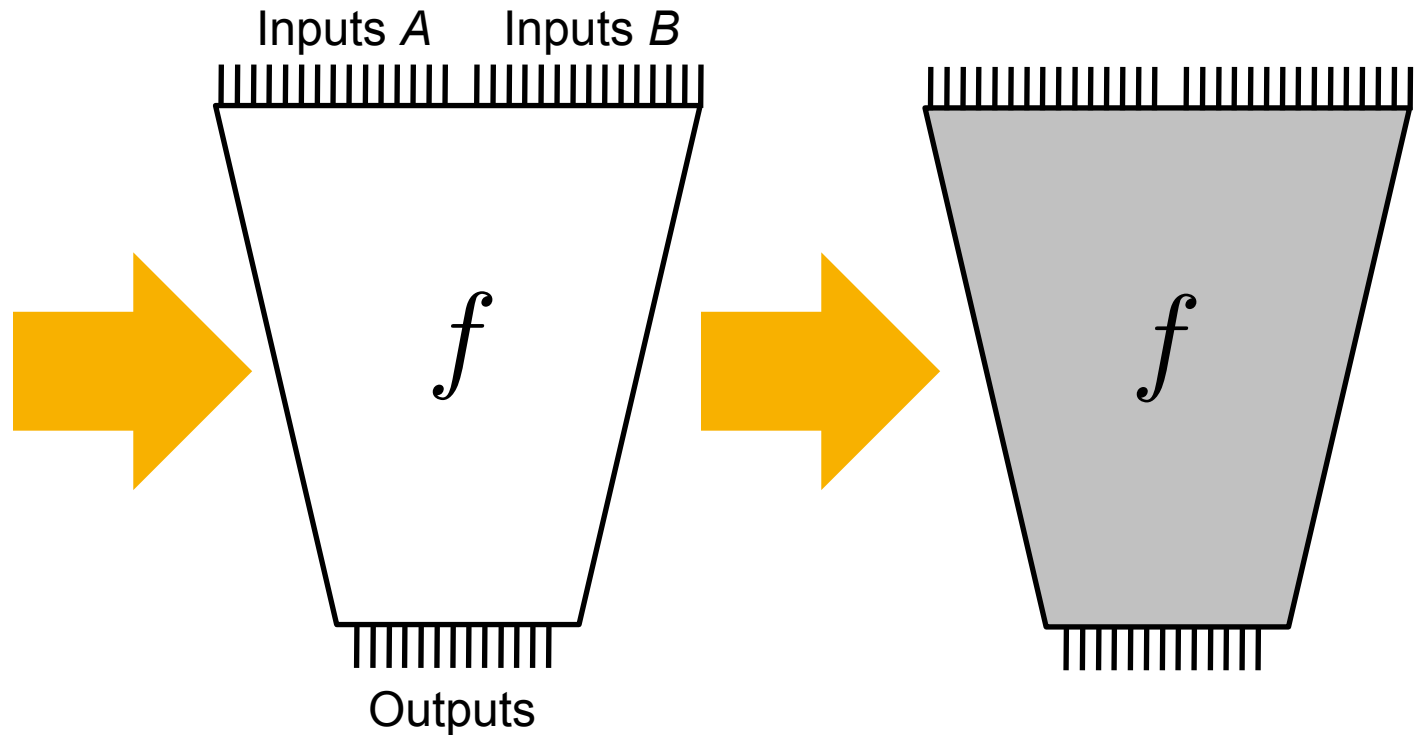
	$x = 1$	$x = 0$
$y = 1$	$E(k_1^1, E(k_2^1, k_3^1))$	$E(k_1^1, E(k_2^0, k_3^0))$
$y = 0$	$E(k_1^0, E(k_2^1, k_3^0))$	$E(k_1^0, E(k_2^0, k_3^0))$

Nice approach, but is it ready for practice?

- Cryptographic frameworks 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 secure computation!

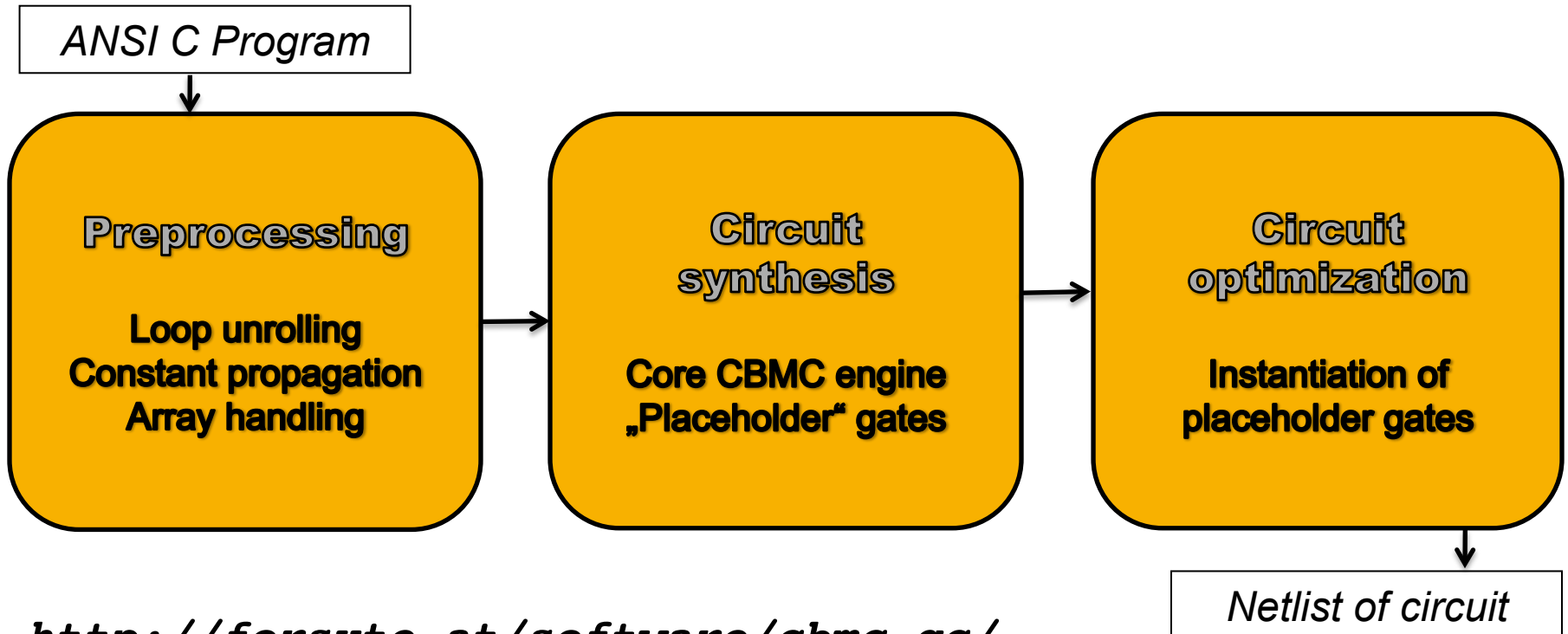
Required Toolchain

```
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;  
}
```



CBMC-GC

Central idea: transforms ANSI C code to circuit useable for secure computation using Yao's Garbled Circuits



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

Basis for CBMC-GC: Bit-precise Model Checker CBMC

- Checks violations of assertions in ANSI C programs



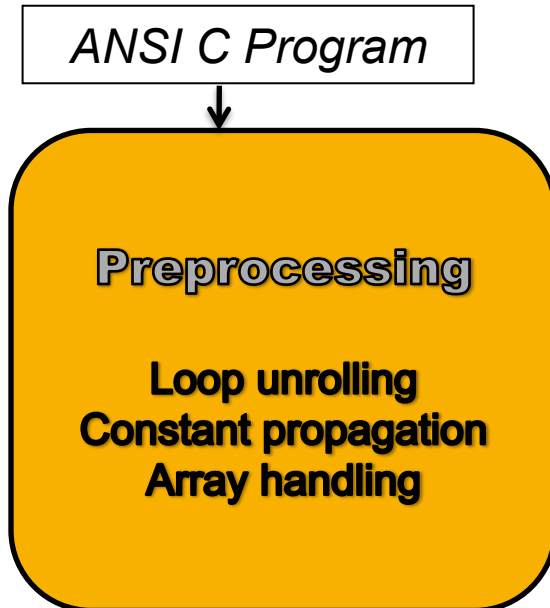
<http://www.cprover.org>

Bounded Model Checker:

- “Bit-precise” transformation of C program into a Boolean formula
 - Boolean formula consists of program model and negated assertions
 - SAT solver checks for solution indicating a violated assertion
-
- We use in CBMC-GC the transformation of an ANSI C program to a Boolean formula
-

CBMC-GC

Preprocessing



- Loop unrolling to
 - ➔ remove all for/while loops
 - ➔ create a flat program
- Bounds for loops need to be determined
 - ➔ constant propagation
 - ➔ manual specification if necessary
- Array handling requires MUX circuits
 - ➔ optimizations for performance

CBMC-GC

Circuit Synthesis



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Circuit synthesis

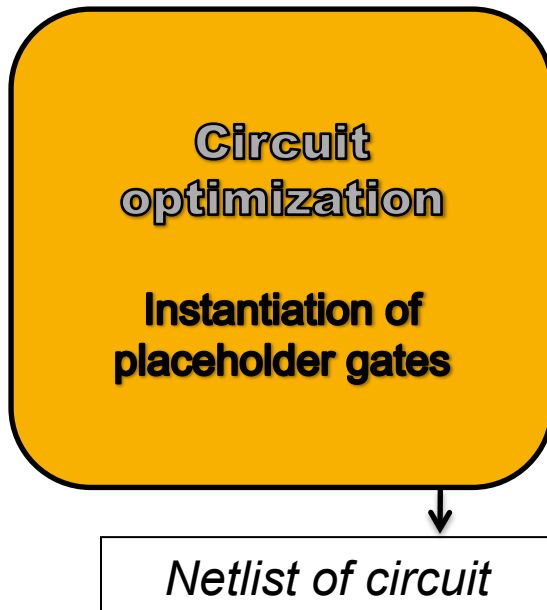
Core CBMC engine
„Placeholder“ gates

- Use core CBMC engine to generate Boolean circuit from unrolled program
 - ➔ code “evolution” of memory
 - ➔ yields “sparse” formula
- Rewriting formula as circuit
- Placeholder gates: high-level representation of certain basic operations
 - ➔ to be optimized at later stage

CBMC-GC

Circuit Optimization


- Instantiation of placeholder gates
 - ➔ choose “most optimal” circuits
 - ➔ many XOR gates good for performance
- Output in the form of netlist
- Circuit can be executed in any framework implementing Yao’s Garbled Circuits



CBMC-GC: Example, Yao's Millionaires

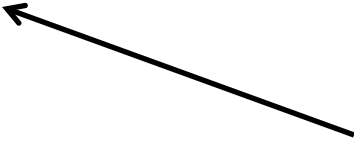
```
void millionaires() {  
    int INPUT_A_mila;  
    int INPUT_B_milb;  
    int OUTPUT_res;
```

Local variables code
inputs and outputs



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

Computations
specified as C program



CBMC-GC: A bigger example

Matrix multiplication

```
#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++)
```

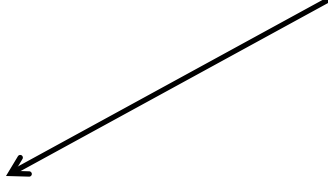
```
                OUTPUT_c[i][j] += INPUT_A_a[i][k] * INPUT_B_b[k][j];
```

```
}
```

More complex data types
like arrays, structs, enums



(Limited) support for
pointer arithmetic



Current Limitations

CBMC-GC inherits limits from model checker CBMC:


- **Bounded programs:** bounds for all loops must be known
 - in practice no problem, as we use terminating programs anyway
- 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

CBMC-GC: More examples

Median computation using Bubblesort

```
#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

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

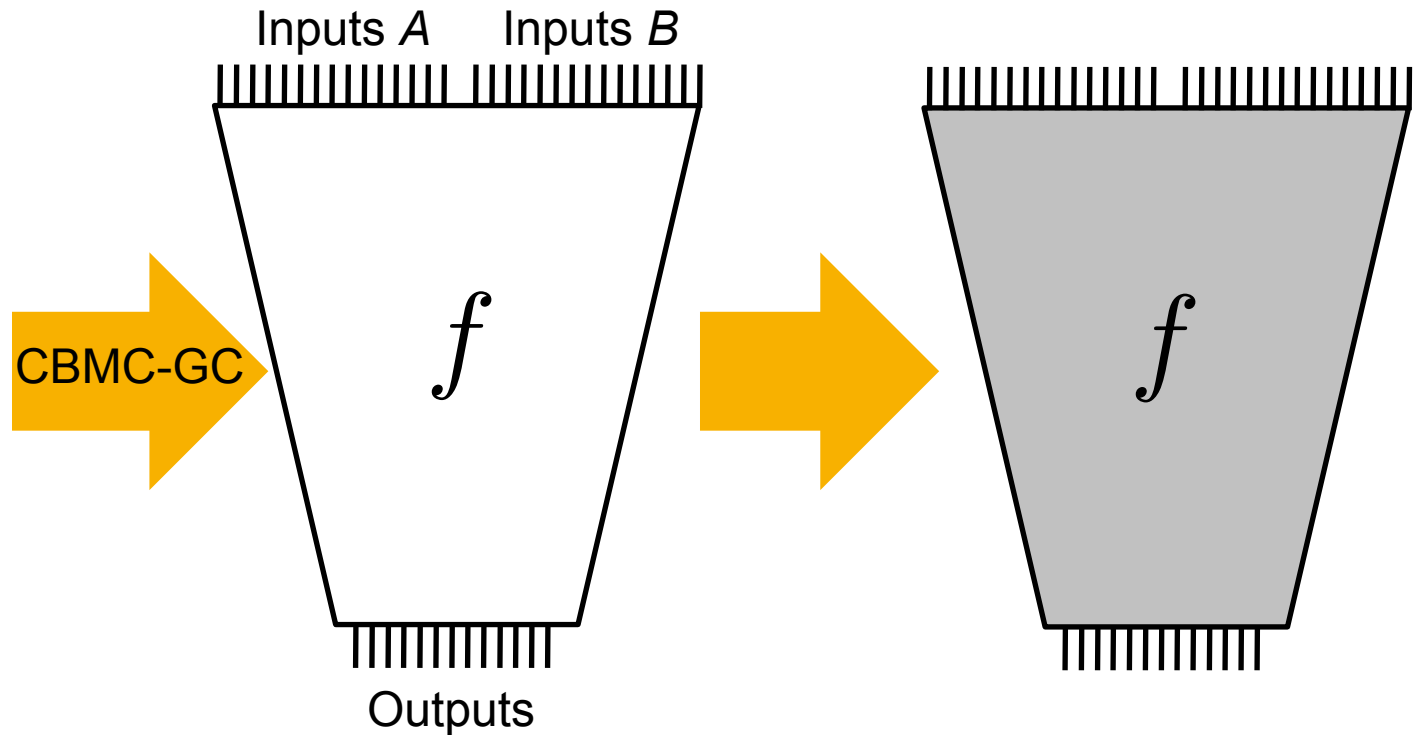
void mergesort(int l, int r) {
    int i, j, k, m;
    if (r > l) {
        m = (r + l)/2; mergesort(l, m); mergesort(m + 1, r);
        for (i = m + 1; i > l; i--)
            b[i - 1] = INPUT_A_a[i - 1];
        for (j = m; j < r; j++)
            b[r + m - j] = INPUT_A_a[j + 1];
        for (k = l;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



Toolchain

```
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;  
}
```



Experimental results

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

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

Future Release of CBMC-GC

Next release (under preparation) will feature:

- More precise heuristics for loop unrolling
- Optimized circuits for basic operations
- Automatic optimization engine for Boolean circuits
- Automatic bit-width selection for integers
- Test framework for circuits

Functionality	Factor
Addition	1
Multiplication	≈ 1
Millionaire's Problem	≈ 2
Hamming Distance	≈ 3.5
Bitwise AND	≈ 2
Bit shifts	≈ 32

Conclusions

- Protection of sensitive data requires **technical means**
- We should not rely (entirely) on administrative measures!
- **Secure computation can be practical already**, despite lack of efficient FHE!
- “Usable” compilers exist
- Integration in software engineering pipeline required...