Polynomial reduction:
If there exists an algorithm breaking my cryptosystem in polynomial time than it is also possible to solve some well-known hard problem in polynomial time.
Polynomial reduction:
  If there exists an algorithm breaking my cryptosystem in polynomial time than it is also possible to solve some well-known hard problem in polynomial time.

Cryptographic assumption in this case states that this well-known hard problem has no polynomial solution.
Assumptions in Cryptography

Polynomial reduction:
If there exists an algorithm breaking my cryptosystem in polynomial time than it is also possible to solve some well-known hard problem in polynomial time.

Cryptographic assumption in this case states that this well-known hard problem has no polynomial solution.

Some classical assumptions:
- Factoring Integers is hard (no polynomial algorithm)
- Discrete Logarithm is hard
- One-Way Functions exist
Today we discuss obfuscations for several function families, which are black-box secure up to various cryptographic assumptions:

Today we discuss obfuscations for several function families, which are black-box secure up to various cryptographic assumptions:


2. Obfuscating Access Control System — Linn, Prabhakaran, Sahai, 2004
Today we discuss obfuscations for several function families, which are black-box secure up to various cryptographic assumptions:

2. Obfuscating Access Control System — Linn, Prabhakaran, Sahai, 2004
3. Point Functions on a Different Assumption — Wee, 2005
Today we discuss obfuscations for several function families, which are black-box secure up to various cryptographic assumptions:

2. Obfuscating Access Control System — Linn, Prabhakaran, Sahai, 2004
3. Point Functions on a Different Assumption — Wee, 2005
Outline


2. Obfuscating Access Control System — Linn, Prabhakaran, Sahai, 2004

3. Point Functions on a Different Assumption — Wee, 2005

Property Hiding

Property hiding informally:
- Function family $F$
- Property $\pi : F \rightarrow \{0, 1\}$
- Given $O(f)$ it is hard to find $\pi(f)$

More formally:
$$\forall A: |\Pr\{A(O(f)) = \pi(f)\} - 1/2| = \nu(|f|)$$

Question: differences with black-box security?
Property Hiding

Property hiding informally:

- Function family $F$
- Property $\pi : F \rightarrow \{0, 1\}$
- Given $O(f)$ it is hard to find $\pi(f)$

More formally:

$$\forall A : |Pr\{A(O(f)) = \pi(f)\} - 1/2| = \nu(|f|)$$
Property Hiding

Property hiding informally:

- Function family $F$
- Property $\pi : F \rightarrow \{0, 1\}$
- Given $O(f)$ it is hard to find $\pi(f)$

More formally:

$$\forall A : |Pr\{A(O(f)) = \pi(f)\} - 1/2| = \nu(|f|)$$

**Question:** differences with black-box security?
Is There Hidden Functionality in the Program?

prog $\pi_1^w$;
var $x$:string, $y$:bit;
input($x$);
if $x = w$ then $y := 1$ else $y := 0$;
output($y$);
end of prog;
Is There Hidden Functionality in the Program?

```
prog π₁\(^w\);
var x:string, y:bit;
input(x);
if x = w then y:=1 else
 y:=0;
output(y);
end of prog;
```

```
prog π₀;
var x:string, y:bit;
input(x);
y:=0; output(y);
end of prog;
```

**Task:** Make this families indistinguishable.
Some Theoretical Background

One-Way Permutation is bijection from the set of all binary strings of length $k$ to itself which is easy to compute and difficult to inverse.

$$F : B^k \rightarrow B^k$$
Some Theoretical Background

One-Way Permutation is bijection from the set of all binary strings of length $k$ to itself which is easy to compute and difficult to inverse.

$$F : B^k \rightarrow B^k$$

Hardcore Predicate for one way permutation $F$ is a predicate (i.e. boolean function) $h$ such that given $F(x)$ it's difficult to predict $h(x)$ better than just guess it.
Some Theoretical Background

One-Way Permutation is bijection from the set of all binary strings of length $k$ to itself which is easy to compute and difficult to inverse.

$$F : B^k \rightarrow B^k$$

Hardcore Predicate for one way permutation $F$ is a predicate (i.e. boolean function) $h$ such that given $F(x)$ its difficult to predict $h(x)$ better than just guess it.

Usual construction of hard-core predicate: choose $r$ by random and take any one way permutation $F$ than given a pair $(F(x), r)$ its difficult to uncover $x \cdot r$. 
Obfuscation for Hidden Functionality

prog Π
var x: string, y: bit;
const u, v: string, σ: bit;
input(x);
if \text{ONEWAY}(x) = v \text{ then}
\text{if } x \cdot u = σ \text{ then } y := 1 \text{ else } y := 0;
\text{else } y := 0;
output(y);
end of prog;
Outline


2. Obfuscating Access Control System — Linn, Prabhakaran, Sahai, 2004

3. Point Functions on a Different Assumption — Wee, 2005

The family of Point function

\[ P_a(x) = 1 \text{ iff } x = a \]
Point Functions

The family of Point function

\[ P_a(x) = 1 \text{ iff } x = a \]

Point functions with output

\[ P_{a,b}(x) = b \text{ iff } x = a \]
Point Functions

The family of Point function

\[ P_a(x) = 1 \text{ iff } x = a \]

Point functions with output

\[ P_{a,b}(x) = b \text{ iff } x = a \]

Multi-point functions with output

\[ P_{A,B}(x) = B_i \text{ iff } x = A_i \]
Random Oracle Model

Random function $R : B^n \rightarrow B^m$ is just a random element from the set of all functions from $B^n$ to $B^m$
Random Oracle Model

Random function $R : B^n \rightarrow B^m$ is just a random element from the set of all functions from $B^n$ to $B^m$

In Random Oracle Model all participants (obfuscator, obfuscated program and adversary) have oracle access to a random function
Point functions: store $R(a)$
Obfuscating Point Functions

- Point functions: store $R(a)$
- Point functions with output: choose random $r$, store $R_1(a, r)$ and $R_2(a, r) \oplus b$
Obfuscating Point Functions

- Point functions: store $R(a)$
- Point functions with output: choose random $r$, store $R_1(a, r)$ and $R_2(a, r) \oplus b$
- Multiple points: repeat above for each point with different $r$
Obfuscating Point Functions

- Point functions: store $R(a)$
- Point functions with output: choose random $r$, store $R_1(a, r)$ and $R_2(a, r) \oplus b$
- Multiple points: repeat above for each point with different $r$

The black-box security is proven for this obfuscation. But some other property of obfuscation is broken. Which one?
Obfuscating Point Functions

- Point functions: store $R(a)$
- Point functions with output: choose random $r$, store $R_1(a, r)$ and $R_2(a, r) \oplus b$
- Multiple points: repeat above for each point with different $r$

The black-box security is proven for this obfuscation. But some other property of obfuscation is broken.
Obfuscating Point Functions

- Point functions: store $R(a)$
- Point functions with output: choose random $r$, store $R_1(a, r)$ and $R_2(a, r) \oplus b$
- Multiple points: repeat above for each point with different $r$

The black-box security is proven for this obfuscation. But some other property of obfuscation is broken.

Which one?
Informally:
- An unknown graph defining access to nodes
- Each edge has a password
- Start at start node
- Exponentially many access patterns
Access Control Mechanism (2)

- A directed multi-graph $G$ on $k$ vertices.
  \[ E = \{(u, v, i) : v = \mu_u^{(i)}\} \]
- A set of passwords $\{\pi_e | e \in E\}$
- A set of secrets at the nodes $\{\sigma_v | v \in [k]\}$
Access Control Mechanism (2)

- A directed multi-graph $G$ on $k$ vertices.
  
  $E = \{(u, v, i) : v = \mu_u^{(i)}\}$

- A set of passwords $\{\pi_e | e \in E\}$

- A set of secrets at the nodes $\{\sigma_v | v \in [k]\}$

$$X_G((i_1, x_1), \ldots, (i_n, x_n)) = \begin{cases} 
  v_n, \sigma_{v_n}, & \text{if } \exists v_0, \ldots, v_n \in [k] \text{ and } \\
  e_0, \ldots, e_{n-1} \in E \text{ such that } \\
  v_0 = 1, e_j = (v_j, v_{j+1}, i_j), \\
  \text{and } x_j = \pi_{e_j} \\
  , & \text{otherwise}
\end{cases}$$
Access Control Mechanism (2)

- A directed multi-graph $G$ on $k$ vertices. 
  \[ E = \{ (u, v, i) : v = \mu_u^{(i)} \} \]
- A set of passwords $\{ \pi_e | e \in E \}$
- A set of secrets at the nodes $\{ \sigma_v | v \in [k] \}$

$$X_G((i_1, x_1), \ldots, (i_n, x_n)) = \begin{cases} 
  v_n, \sigma_{v_n}, & \text{if } \exists v_0, \ldots, v_n \in [k] \text{ and } \\
  e_0, \ldots, e_{n-1} \in E \text{ such that } \\
  v_0 = 1, e_j = (v_j, v_{j+1}, i_j), \\
  \text{and } x_j = \pi_{e_j}, & \text{otherwise}
\end{cases}$$

Claim: obfuscation of access functions can be reduced to that of multi-point functions
Outline

2. Obfuscating Access Control System — Linn, Prabhakaran, Sahai, 2004
3. Point Functions on a Different Assumption — Wee, 2005
New Assumption

There exists a polynomial-time computable permutation $\pi : B^n \rightarrow B^n$ and a constant $c$ such that for every polynomial $s(n)$ and every adversary $A$ of size $s$ for all sufficiently large $n$,

$$Pr[A(\pi(x)) = x] \leq s(n)^c / 2^n$$
New Construction

Instead of $R(a)$ we will store:

$$h(x, \tau_1, \ldots, \tau_{3n}) = (\tau_1, \ldots, \tau_{3n}, \langle \pi(x), \tau_1 \rangle, \ldots, \langle \pi^{3n}(x), \tau_{3n} \rangle)$$
Outline

2. Obfuscating Access Control System — Linn, Prabhakaran, Sahai, 2004
3. Point Functions on a Different Assumption — Wee, 2005
Quiz in a Newspaper

- We publish some problem in a newspaper
- We want publish some additional check-yourself information
- We want that this information will contain almost zero information about the answer
Let the answer is $x$
We will publish:

$$H(x) = (r^2, r^{2h(x)})$$

If hash function $h$ is collision-free, then $H$ is black-box secure about $x$. 

Let the answer is $x$
We will publish:

$$H(x) = (r^2, r^{2h(x)})$$

If hash function $h$ is collision-free, then $H$ is black-box secure about $x$.

Assumption: Let $p = 2q - 1$ and $a, b, c \in R Z_q^*$. Then distributions $\langle g^a, g^b, g^{ab} \rangle$ and $\langle g^a, g^b, g^c \rangle$ are computationally indistinguishable
Hash-based Construction

Let the answer is $x$
We will publish:

$$H(x) = (r, h(r, h(x)))$$
2. Prove that probability of changing functionality in obfuscating point functions by using random oracle from $B^n$ to $B^{3n}$ is negligible.

3. Prove that $s^c(n)/2^n$, where $s$ is a polynomial and $c$ is a constant, is a negligible function.
Summary

Main points:
- It is possible to obfuscate point functions with black-box security

Security proofs are based on various cryptographic assumptions:
- existence of one-way functions,
- Diffie-Hellman indistinguishability,
- random oracle model.

To be honest, all solutions obscure data rather than algorithm.
Main points:

- It is possible to obfuscate point functions with black-box security.
- Security proofs are based on various cryptographic assumptions: existence of one-way functions, Diffie-Hellman indistinguishability, random oracle model.
Main points:

- It is possible to obfuscate point functions with black-box security.
- Security proofs are based on various cryptographic assumptions: existence of one-way functions, Diffie-Hellman indistinguishability, random oracle model.
- To be honest, all solutions obfuscate data rather than algorithm.
R. Canetti

N. Varnovsky and V. Zakharov
On the possibility of provably secure obfuscating programs, 2003.
http://www.springerlink.com/index/5JG4QL4TR9RVD3NK.pdf.

B. Linn, M. Prabhakaran, A. Sahai

H. Wee
On obfuscating point functions, 2005.

Thanks for attention. Questions?