Techniques For Obfuscation:
Relaxations and New Notions

by

Amitabh Saxena
DIT, UNITN, Trento, Italy 38100
Layout of Talk

1. Relaxed definitions of obfuscation
   - Special purpose obfuscators
   - Predicate obfuscators

2. Some positive results
   - Point function obfuscation (Random oracles)
   - Point function obfuscation (w/o ROs)
   - Some predicate obfuscators

3. Some more notions of obfuscation
For any two programs \( (P_1, P_2) \), we say \( P_1 \sim P_2 \) if both programs have identical functionality.

- Also extends to “approximate” functionality (For all but a negligible fractions of inputs, both programs have identical outputs).

We say any Boolean predicate \( \pi \) of the programs \( (P_1, P_2) \) is a semantic predicate if

\[
P_1 \sim P_2 \text{ implies } \pi(P_1) = \pi(P_2)
\]
Obfuscation [Barak et al.]

Obfuscator is an algorithm $O$ s.t. for all programs $P$:

- **Functionality**: $O(P) \sim P$
- **Poly Slowdown**:
  - $\text{Size}(O(P)) < \text{Poly}(\text{Size}(P))$
  - $\text{Time}(O(P)) < \text{Poly}(\text{Time}(P))$
- **Virtual Black-Box**: For all semantic predicates $\pi$, and for all algorithms $A$, there exists a simulator $S$ such that:

$$\Pr[A(O(P)) = \pi(P)] - \Pr[S^{P} = \pi(P)] \approx 0$$

- Such an obfuscator $O$ cannot exist! [Barak01]

By constructing a “cannibalistic” program that says:

“Feed me somebody that behaves like me, and I'll leak my secret!”
What can we hope to achieve?

We relax some of the requirements of [Barak01]

Obfuscator is an algorithm $O$ s.t. for certain programs $P$:

- **Functionality:** $O(P) \sim P$
- **Poly Slowdown:**
  - $\text{Size}(O(P)) < \text{Poly}(\text{Size}(P))$
  - $\text{Time}(O(P)) < \text{Poly}(\text{Time}(P))$
- **Virtual Black-Box:** For certain semantic predicates $\pi$, and for all algorithms $A$, there exists a simulator $S$ such that:
  
  $$\Pr[A(O(P)) = \pi(P)] - \Pr[S^P = \pi(P)] \approx 0$$

- “Special purpose obfuscator” and/or “Predicate obfuscator”
More Relaxations…. 

- Allow Random Oracles (RO) 
- RO is a type of “black-box” with true randomness
  - Infeasible to predict output on some input without making an explicit query to the RO
  - Infeasible to find collisions (2 inputs give same output)
- In the RO model, all participants (obfuscator, attacker and obfuscated program) have access to the random oracle.
- Lynn et al. use RO to securely obfuscate certain predicates of point functions. [Lynn04]
Point Function Obfuscation [Lynn04]

- A point function outputs 1 at only one input and 0 otherwise
- Password checking programs: (Password is “hello world!”)
  
  ```
  VERIFY_PASSWORD (Input X){
    If (X==“hello world!”) Then output 1;
    Else output 0;
  }
  ```

  Let Random_Oracle(“hello world!”) = 813841341

  ```
  VERIFY_PASSWORD_OBF (Input X){
    If (Random_Oracle(X) == 813841341) Then output 1;
    Else output 0;
  }
  ```

- Let $\pi_i(VERIFY_PASSWORD)$ denote $i^{th}$ bit of password.
- Provably secure obfuscation of predicate $\pi_i$ for all $i$.
- Obfuscation preserves approximate functionality!
Point Functions with output [Lynn 04]

- Instead of 1, the function on some input $a$ outputs some value $b > 1$
  - Obfuscation: Use two random oracles. Generate random $r$ and store
    \{ $r$, Random_Oracle1 $(a, r)$, Random_Oracle2 $(a, r)$ XOR $b$ \}

- Multi-point functions with output: Many point functions with output:
  \[
  F_{A, B}(x) = B_i \text{ if } x = A_i
  \]
  - Obfuscation: Repeat above for each input/output pair (with different $r$)

- Multi-point functions for “Access control” (via directed graphs)
  - Edge $i$ has “password” $A_i$ needed to access secret $B_i$ at head node
  - Can only access some node if we can prove a path from start node.

- The above method to obfuscate a multi-point function with output is
  secure assuming single point function obfuscation is secure [Lynn04]
  - Key idea is a “composition of obfuscations” Lemma
[Wee05] uses the basic idea of [Lynn04] for obfuscating point functions with random oracles.

Gives an instantiation of random oracle under assumption that a certain type of one-way permutation exists.

These types of one-way permutations are believed to exist (eg. RSA).

One of the few constructions where a random oracle can be instantiated by a real function.

*Caveat:* Technique of [Lynn04] to convert point function obfuscation to multi-point function obfuscation fails!

- “composition of obfuscations” Lemma does not work for [Wee05]
Predicate Obfuscation (some more)

(Learnable v/s non-learnable)

Two fundamental types of programs

- Learnable: (can re-create source code just from few I/O queries)
  
  ```
  Program_1 (input X)
  /* Ignore input */
  Output 0;
  ```

- Not learnable: (cannot re-create source code from few I/O queries)
  
  ```
  Program_2 (input X)
  If (X == 1668801023012013) Then
    Output 1;
  Else
    Output 0;
  ```

- Predicate $\pi (P)$: To decide if program $P$ is learnable or not.

- In other words, given Program$_i$ for unknown $i \leftarrow \{1, 2\}$, to decide:
  
  - Does there exist $X$ such that Program$_i(X) = 1$?
Predicate Obfuscation …
(learnable v/s non-learnable)

- From previous slide, Program_2 (non-learnable) contains some “hidden” functionality inside, while Program_1 (learnable) does not.

- **Applications:** (perhaps) watermarking [Varnovsky03] (watermarked program contains some hidden functionality)

- **Goal:** Want to hide the predicate $\pi$ that indicates if the program can ever output 1 or not.

- [Varnovsky03] give a method for hiding which program (from previous slide) is given.

- Their construction is based on any one-way function and information theoretic.

- We will give (for simplicity) a construction using a number-theoretic primitive. We assume that for a composite $n$, with unknown factorization,
  1. Computing square roots $\mod n$ is as hard as factoring $n$
  2. For any $1 < x < n-1$, such that Jacobi_Symbol $(x, n)=1$, deciding if $x$ is a quadratic residue $\mod n$ is hard.
learnable v/s non-learnable...

Let $k$ be the bit-length of $X$ such that $\text{Program} _2(X) = 1$

Generate $n = pq$ for large primes $p, q$ (assume $|k| = |n|$). Let $y = X^2 \mod n$ and let $w$ be a quadratic non-residue $\mod n$ such that $\text{Jacobi}_\text{symbol}(w, n)=1$.

Since $w$ is a quadratic non-residue, $\text{Obf}_\text{Program} _1$ will never output 1.

Without factors of $n$ we cannot decide if $w$ (or $y$) is a quadratic residue or not.

Hence provably secure obfuscation of the predicate $\pi$. 

---

Program_1 (input X){
  /* Ignore input */
  Output 0;
}

Program_2 (input X){
  If (X == 1668801023012013) Then
    Output 1;
  Else
    Output 0;
}

Obf_Program_1 (input X){
  const $w$ ;
  If (X$^2 \mod n == w$) Then
    Output 1;
  Else
    Output 0;
}

Obf_Program_2 (input X){
  const $y$ ;
  If (X$^2 \mod n == y$) Then
    Output 1;
  Else
    Output 0;
}
More notions of obfuscation

- Indistinguishability obfuscation [Barak01]
  - If $P_1 \sim P_2$, then the obfuscations $O(P_1)$ and $O(P_2)$ are indistinguishable.
  - Not clear how much information is “hidden”!

- Best-possible obfuscation [Goldwasser07]
  - The obfuscation $O(P)$ leaks as little information as possible, and is therefore the “best possible”.
  - Informally, any other program $P' \sim P$ with $|P'| \leq |O(P)|$ leaks more information than $O(P)$.
  - Formal definitions given for circuits (but we will skip this).
  - Mostly negative results! 😞
Obfuscating Re-Encryption

Re-encryption for asymmetric ciphers

- Given a ciphertext encrypted under Alice’s encryption key, transform it into a ciphertext under Bob’s encryption key (without knowing Alice’s decryption key). Thus, some sort of obfuscation is required.
- [Hohenberger07] gave an obfuscation for re-encryption using bilinear maps.
- Uses a slightly different notion of obfuscation
  - “Average case secure obfuscation” – Indistinguishability of the output of an adversary with access to obfuscated code and that of simulator with black-box access to code.
- **Drawback:** Can only re-encrypt once.
End of Talk!

- Questions?
References

[Barak01] “On the (im)possibility of obfuscating programs”, CRYPTO 2001

[Lynn04] “Positive results and techniques for obfuscation”, EUROCRYPT 2004

[Wee05] “On obfuscating point functions”, STOC 2005


[Hohenberger07] “Securely obfuscating re-encryption”, TCC 2007