Provable Security for Program Obfuscation
1. Idea of Provable Security
   - Ways to Achieve Security

2. Basic Results
   - Impossibility of obfuscation
   - Property Hiding
   - Encrypted computation

3. Overview of Further Research
   - Mobile cryptography
   - Black-box Security
   - Practical Approach
What do we want to get?

We want to be sure that our system is safe to use.

Today: what does it mean to be sure about safety?

Usual approach: to build some proof of safety.
How are we going to prove security?

- **Theoretic security**: obfuscated program doesn’t provide enough information to successful attack.
  Example: exact reverse engineering. Solution: delete comments.

- **Computational (cryptographic) security**: attack required too much computation.

Necessary hardness of attack: average superpolynomial complexity.
Now: no problems with such **proved** complexity.
So what can we accept as enough hard problem?

⇒ NP-hard problems. Disadvantage: worst case complexity

⇒ NP-hard problems with average complexity results. Example: SUBSET SUM

⇒ Problems with wide-believed hardness: Examples: FACTORING, DISCRETE LOG
What are the best results to the moment?

⇒ Specific attacks on specific programs are computationally hard
⇒ For some classes of programs we can hide most of internal information
⇒ Some program analysis is proved to be hard
⇒ And obfuscation in general is impossible!
A nondeterministic algorithm $O$ is a **TM obfuscator** if three following conditions hold:

- **(functionality)** For every TM $M$, the string $O(M)$ describes the same function as $M$.

- **(polynomial slowdown)** The description length and running time of $O(M)$ are at most polynomially larger than that of $M$.

- **("virtual black box" property)** For any PPT $A$, there is a PPT $S$ and a negligible function $\alpha$ such that for all TMs $M$

\[
\left| Pr[A(O(M)) = 1] - Pr[S^M(1^{|M|}) = 1] \right| \leq \alpha(|M|).
\]
A 2-TM obfuscator is defined in the same way as a TM-obfuscator, except the “virtual black box” property is changed as follows:

\[ Pr[A(O(M), O(N)) = 1] - Pr[S^{M,N}(1^{|M|+|N|}) = 1] \leq \alpha(\min(|M|, |N|)). \]

**“virtual black box” property** For any PPT A, there is a PPT S and a negligible function \(\alpha\) such that for all TMs M and N.
A 2-TM obfuscator is defined in the same way as a TM-obfuscator, except the "virtual black box" property is changed as follows:

\[ \Pr[A(O(M), O(N)) = 1] - \Pr[S^{M,N}(1 |M|+| N| ) = 1] \leq \alpha(\min(|M|, |N|)). \]

What obfuscator is more powerful?
Two Programs Lemma

2-TM obfuscators do not exist.

\[ C_{\alpha,\beta}(x) = \begin{cases} \beta, & x = \alpha \\ 0, & \text{otherwise} \end{cases} \]

\[ D_{\alpha,\beta}(C) = \begin{cases} 1, & C(\alpha) = \beta \\ 0, & \text{otherwise} \end{cases} \]

\[ Z_k(x) = 0^k \]

**Intuition:** it is difficult to distinguish pairs \( C_{\alpha,\beta}, D_{\alpha,\beta} \) from pair \( Z_k, D_{\alpha,\beta} \) given only black box access to these programs.
Suppose $O$ is 2-TM obfuscator. Let’s check its “black box” property on pairs $C_{\alpha,\beta}, D_{\alpha,\beta}$ and $Z_k, D_{\alpha,\beta}$ for every $\alpha, \beta$ where $A = N(M)$.

\[
\Pr[A(O(C_{\alpha,\beta}), O(D_{\alpha,\beta})) = 1] = 1
\]

\[
\Pr[A(O(Z_k), O(D_{\alpha,\beta})) = 1] = 2^{-k}
\]

\[
\left| \Pr[S_{C_{\alpha,\beta}, D_{\alpha,\beta}} = 1] - \Pr[S_{Z_k, D_{\alpha,\beta}} = 1] \right| \leq 2^{-\Omega(k)}
\]

So we get a contradiction! But...

**There is a flaw in the proof. Do you see?**
Impossibility Theorem

TM obfuscators do not exist.

\[ F_{\alpha,\beta}(b, x) = C_{\alpha,\beta} \# D_{\alpha,\beta} \]

\[ G_{\alpha,\beta}(b, x) = Z_k \# D_{\alpha,\beta} \]

Algorithm \( A \) is the following: to decompose \( M \) into two parts and evaluate the second part on the code (encoding) of the first.

Argument is similar to the Lemma’s proof.
Instance: two families of programs $\Pi_1$ and $\Pi_2$

**Adversary task:** given a program $P \in \Pi_1 \cup \Pi_2$ to decide whether $P \in \Pi_1$ or $P \in \Pi_2$.

**Desirable protection:** make adversary task as difficult as well-known computationally hard problem is.
**Password Checking Hiding**

**Task:** Make this families indistinguishable.

```
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;
```
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 $x \cdot r$. 
Program with hidden password checking

```
prog \( \Pi \)
var \( x \): string, \( y \): bit;
const \( u, v \): string, \( \sigma \): bit;
input(\( x \));
if \( \text{ONE\_WAY}(x) = v \) then
    if \( x \cdot u = \sigma \) then \( y := 1 \) else \( y := 0 \);
else \( y := 0 \);
output(\( y \));
end of prog;
```
Slide from Lecture 1 — your turn to explain.

Basic task: keep $F$ unknown to Bob.
Homomorphic Encryption

General idea: to design an encoding such that it is possible to evaluate various operations over encrypted messages (and getting encrypted results) without decrypting them.

In particular encoding is called

- **Additively homomorphic** if it is possible to compute $E(x + y)$ from $E(x)$ and $E(y)$

- **Multiplicatively homomorphic** if it is possible to compute $E(xy)$ from $E(x)$ and $E(y)$

- **Mixed multiplicatively homomorphic** if it is possible to compute $E(xy)$ from $E(x)$ and $y$. 
**Fact:** there exists additively homomorphic encryption schemes over the rings $\mathbb{Z}/N\mathbb{Z}$.

**Corollary:** there exists additively & mixed multiplicatively homomorphic encryption schemes over the rings $\mathbb{Z}/N\mathbb{Z}$.

**Proof:** Mixed multiplication could be done by polynomial number of additions.
Let $P$ be polynomial over $\mathbb{Z}/N\mathbb{Z}$ ring.

$$P = \sum a_{i_1 \ldots i_s} X_1^{i_1} \ldots X_s^{i_s}$$

Then we can encrypt $P$ by just sending encrypted coefficients (using MM-A homomorphic encryption). Bob is able to compute $E(P(X))$ and return it back to Alice.

What we reveal to Bob? Only set of nonzero coefficients of $P$. 
What are further results for encrypted computation?

⇒ Other presentations of function.
   - [Loreiro, Molva] – function as a matrix.
What is hard to get from programs after obfuscating transformations?

⇒ Alias analysis is NP-hard!
⇒ Average hardness is proved only for several fixed analysis algorithms
We can prove property extracting to be hard in some cases.

We can use cryptographic constructions to hide some internal constants.

Obfuscation in general is impossible.