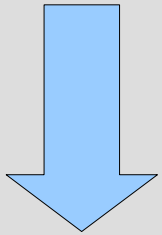
A cryptographic view on obfuscation

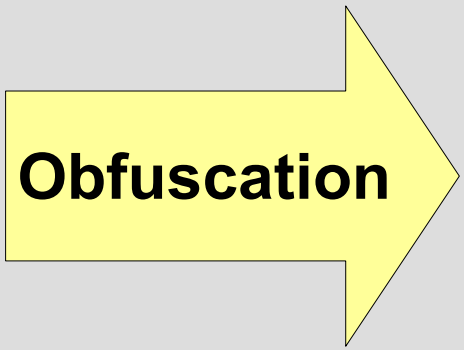
Dennis Hofheinz (CWI, Amsterdam)

Intuition behind obfuscation

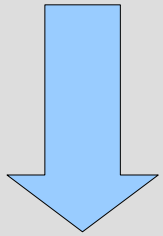
```
int main() {
 printf("Hi world");
}
```



Hi world!




```
          ...
strcpy(&x,&y,12);
for(z=0;z<--x;z++){
          ...
```



Hi world!

- Make code unintelligible
- But: preserve functionality

Applications of obfuscation

- Practical: **protect know-how**
- “In between example”: password check
- Theoretical:
  - Turn secret key crypto into public key crypto
  - (Fully) homomorphic encryption → MPC
  - Implement ideal assumptions
    - Random oracles, generic groups, ideal ciphers
  - ...

Negative results

- [B+01]: All-purpose obfuscators do not exist
  - Some program classes cannot be obfuscated
  - Example artificial, composition of point functions
  - Does not rule out **specific** obfuscators
- [GR07]: Relaxing the notion does not help
- [Hada00,Wee05]: obfuscation ↔ learnability
  - *f* obfuscatable ⇔ *f* learnable from oracle access
- [B+01,GT05]: Unobfuscatability examples

Positive results

- [LPS04]: “Practical” obfuscations using ROs
  - Control flow of programs obfuscated
- [C97,CMR98,Wee05,HMS07]: Point functions
  - Point function: $f_s(x)=1$ iff $s=x$
  - Use case: password check: s=pwd, x=user input
- [HRSV07]: Obfuscation of re-encryption
  - Decrypt and encrypt under different public key

Contradictory results?

- [Wee05]: Obfuscatable iff learnable
- [Wee05]: Obfuscation of point functions
- But: point functions not learnable from oracle!

- Major problem: No common definition
  - Results depend on definition
  - Exception: [B+01] (no all-purpose obfuscators)

This talk

- [HMS07] obfuscation definition
  - “Remix” of previous definitions
  - Useful, not too weak:
    - Secret key crypto → public key crypto possible
    - Password check possible
  - **At the same time** meaningful, not too strong:
    - Point functions easily obfuscatable → pwd check
    - Secret key encryption **in principle** obfuscatable
- Note: this is **not** the only reasonable definition

Obfuscation definition(s)

- General idea:

A good obfuscation yields nothing but functionality

Obfuscation definition(s)

- ... just a little more formal:

O(*f*) is a good obfuscation of *f*

$\Leftrightarrow$

O(*f*) provides the functionality of *f*, but nothing else

Obfuscation definition(s)

- The problematic part:

O(*f*) is a good obfuscation of *f*

⇔

O(*f*) provides the functionality of *f*, but nothing else

- How to formalize “but nothing else”?

Obfuscation definition(s)

- Cryptographic version:

O(*f*) is a good obfuscation of *f*

⇔

1. O(*f*) provides the functionality of *f*
2. Everything that can be learned from O(*f*)...
   ... can also be learned from oracle access to *f*

Obfuscation definition(s)

- Cryptographic version, simulation-based:

O(*f*) is a good obfuscation of *f*

⇔

1. O(*f*) provides the functionality of *f*
2. There is a simulator S such that:
   O(*f*) computationally^f indistinguishable from S^f

even with oracle access to *f*

S^f denotes S's output when run with oracle access to f
understood: S has to be efficient

Our definition

- More realistic: class of functions

O(f) is a good obfuscation of class F={f}

$\Leftrightarrow$ on average (uniform f), it holds that

1. O(f) provides the functionality of f
2. There is a simulator S such that:
   O(f) computationally^f indistinguishable from S^f

- Why? f indexed, e.g., by secret pwd, key, ...
  - Why uniform distribution? Use cases!

Our definition

- Complete definition, explicit formulation:

O(f) is a good obfuscation of class F={f}

$\Leftrightarrow$ on average (uniform f), it holds that

1. O(f) provides the functionality of f
2. There is a simulator S such that for all D,
   $Pr[D^f(O(f)) = 1] - Pr[D^f(S^f) = 1]$
   is negligible

- Understood: complexity (O, S, D PPT)

Application: password check

- Scenario: user tries to log into system
- Authentication through password query
- Goal 1: only correct password gives access
- First solution: store password on terminal
- Problem: what if a terminal gets corrupted?
- Goal 2: verification **functionality**, but not more

Application: password check

- Goal 1: only correct password gives access
- Goal 2: verification functionality, but not more
- Solution 2: obfuscate point function f_{pwd}
  - Definition: $f_{pwd}(x)=1$ iff $pwd=x$
  - Access iff $f_{pwd}(user\_input)$ evaluates to 1
  - Goal 1 is 1st requirement on good obfuscation
  - Goal 2 is 2nd requirement on good obfuscation

Point functions

- But: how to obfuscate point functions?

O(f) is a good obfuscation of class F={f_{pwd}}

$\Leftrightarrow$

on average (uniform *pwd*), it holds that

1. O(f_{pwd}) provides the functionality of f_{pwd}
2. There is a simulator S such that:
   O(f_{pwd}) computationally^{f} indistinguishable from $S^{f_{pwd}}$

Point functions

O(f) is a good obfuscation of class F={f_{pwd}}

$\Leftrightarrow$

on average (uniform *pwd*), it holds that

1. O(f_{pwd}) provides the functionality of f_{pwd}
2. There is a simulator S such that:
   O(f_{pwd}) computationally^{f} indistinguishable from $S^{f_{pwd}}$

- Solution: O(f_{pwd}) = g(*pwd*) for one-way perm. g
  - S outputs g(*pwd'*) for uniform, independent *pwd'*
  - Looks like O(f_{pwd})=g(*pwd*) since f-oracle useless

Other properties

- Secret key crypto → public key crypto:

  If you obfuscate the encryption algorithm
  (with hardwired secret key) of a symmetric encryption scheme

  ... you get a secure public key encryption scheme

- In principle, there are obfuscatable schemes
  - But not very interesting: PKE interpreted as SKE
- Works **only** for passive adversaries (CPA)
- Result of [B+01] holds: no generic obfuscator

Conclusion

- Obfuscation from a cryptographic viewpoint
  - No all-purpose obfuscator
  - Lots of definitions, not many positive results
  - But: no reason to be overly pessimistic!
- Not mentioned: **composition** of obfuscators
  - Core of impossibilities: composition defects
  - Essential for modular analysis
  - Open: **reasonable** composable results (RO?)