Obfuscation

ECE/CS 407

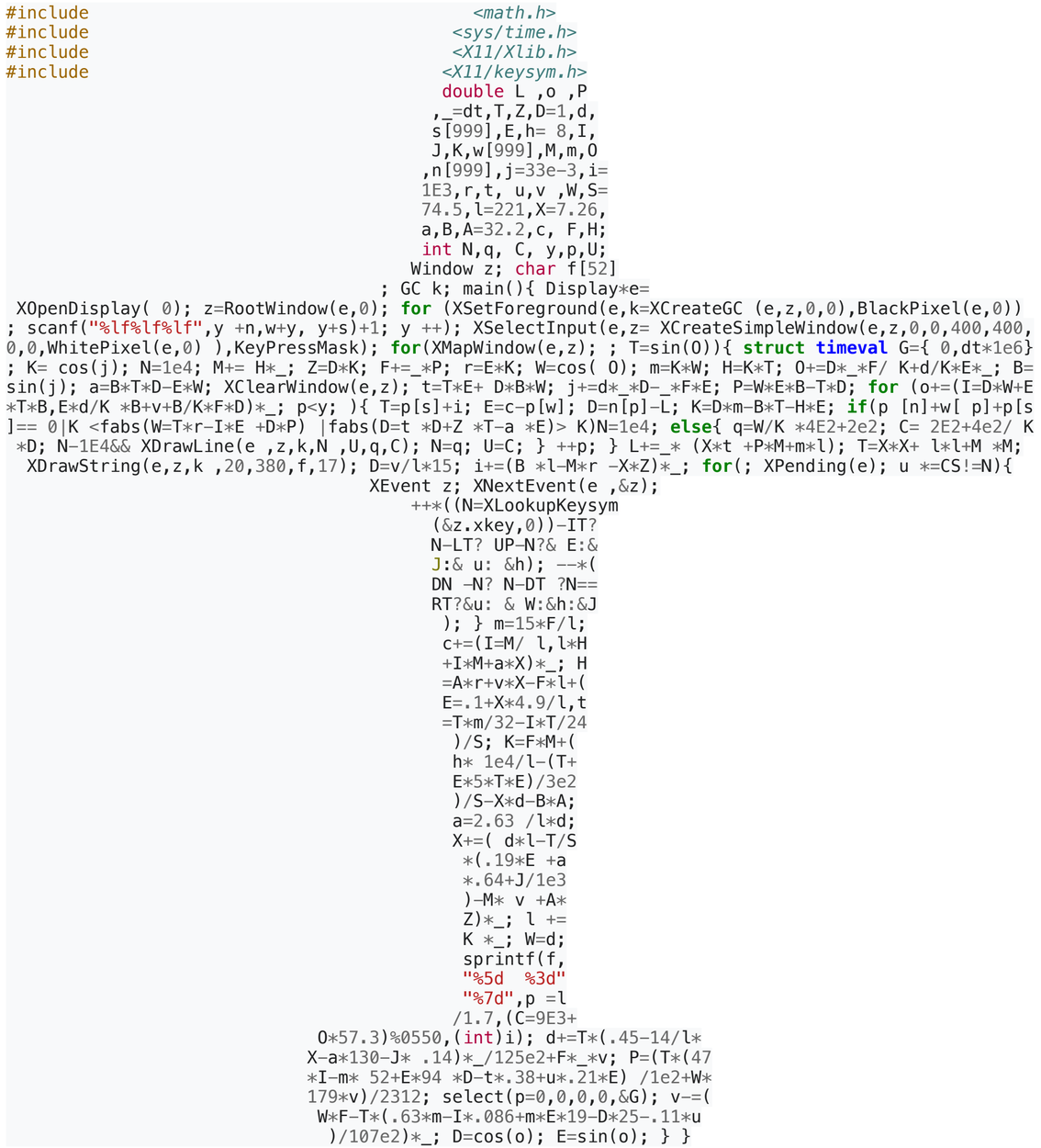
Today’s objectives

Understand the notion of an obfuscation

Define indistinguishability obfuscation

Applications of iO, how does iO fit into cryptography

1998 Winner, International Obfuscated C Code Contest



```
#include                                    <math.h>
#include                                   <sys/time.h>
#include                                   <X11/Xlib.h>
#include                                  <X11/keysym.h>
                                           double L ,o ,P
                                          ,_=dt,T,Z,D=1,d,
                                          s[999],E,h= 8,I,
                                          J,K,w[999],M,m,O
                                         ,n[999],j=33e-3,i=
                                         1E3,r,t, u,v ,W,S=
                                         74.5,l=221,X=7.26,
                                         a,B,A=32.2,c, F,H;
                                         int N,q, C, y,p,U;
                                       Window z; char f[52]
                                    ; GC k; main(){ Display*e=
 XOpenDisplay( 0); z=RootWindow(e,0); for (XSetForeground(e,k=XCreateGC (e,z,0,0),BlackPixel(e,0))
; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400,
0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=sin(O)){ struct timeval G={ 0,dt*1e6}
; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_*P; r=E*K; W=cos( O); m=K*W; H=K*T; O+=D*_*F/ K+d/K*E*_; B=
sin(j); a=B*T*D-E*W; XClearWindow(e,z); t=T*E+ D*B*W; j+=d*_*D-_*F*E; P=W*E*B-T*D; for (o+=(I=D*W+E
*T*B,E*d/K *B+v+B/K*F*D)*_; p<y; ){ T=p[s]+i; E=c-p[w]; D=n[p]-L; K=D*m-B*T-H*E; if(p [n]+w[ p]+p[s
]== 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T-a *E)> K)N=1e4; else{ q=W/K *4E2+2e2; C= 2E2+4e2/ K
 *D; N-1E4&& XDrawLine(e ,z,k,N ,U,q,C); N=q; U=C; } ++p; } L+=_* (X*t +P*M+m*l); T=X*X+ l*l+M *M;
  XDrawString(e,z,k ,20,380,f,17); D=v/l*15; i+=(B *l-M*r -X*Z)*_; for(; XPending(e); u *=CS!=N){
                                   XEvent z; XNextEvent(e ,&z);
                                       ++*((N=XLookupKeysym
                                         (&z.xkey,0))-IT?
                                         N-LT? UP-N?& E:&
                                         J:& u: &h); --*(
                                         DN -N? N-DT ?N==
                                         RT?&u: & W:&h:&J
                                          ); } m=15*F/l;
                                          c+=(I=M/ l,l*H
                                          +I*M+a*X)*_; H
                                          =A*r+v*X-F*l+(
                                          E=.1+X*4.9/l,t
                                          =T*m/32-I*T/24
                                           )/S; K=F*M+(
                                           h* 1e4/l-(T+
                                           E*5*T*E)/3e2
                                           )/S-X*d-B*A;
                                           a=2.63 /l*d;
                                           X+=( d*l-T/S
                                            *(.19*E +a
                                            *.64+J/1e3
                                            )-M* v +A*
                                            Z)*_; l +=
                                            K *_; W=d;
                                            sprintf(f,
                                            "%5d  %3d"
                                            "%7d",p =l
                                           /1.7,(C=9E3+
                                  O*57.3)%0550,(int)i); d+=T*(.45-14/l*
                                 X-a*130-J* .14)*_/125e2+F*_*v; P=(T*(47
                                 *I-m* 52+E*94 *D-t*.38+u*.21*E) /1e2+W*
                                 179*v)/2312; select(p=0,0,0,0,&G); v-=(
                                  W*F-T*(.63*m-I*.086+m*E*19-D*25-.11*u
                                   )/107e2)*_; D=cos(o); E=sin(o); } }
```

Obfuscation makes a piece of code *unintelligible*

Basic idea: you can’t observe anything about the internal workings of the program, it is a **black box**

```
#include                                    <math.h>
#include                                   <sys/time.h>
#include                                   <X11/Xlib.h>
#include                                  <X11/keysym.h>
                                          double L ,o ,P
                                         ,_=dt,T,Z,D=1,d,
                                         s[999],E,h= 8,I,
                                         J,K,w[999],M,m,O
                                        ,n[999],j=33e-3,i=
                                        1E3,r,t, u,v ,W,S=
                                        74.5,l=221,X=7.26,
                                        a,B,A=32.2,c, F,H;
                                        int N,q, C, y,p,U;
                                      Window z; char f[52]
                                    ; GC k; main(){ Display*e=
 XOpenDisplay( 0); z=RootWindow(e,0); for (XSetForeground(e,k=XCreateGC (e,z,0,0),BlackPixel(e,0))
; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400,
0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=sin(O)){ struct timeval G={ 0,dt*1e6}
; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_*P; r=E*K; W=cos( O); m=K*W; H=K*T; O+=D*_*F/ K+d/K*E*_; B=
sin(j); a=B*T*D-E*W; XClearWindow(e,z); t=T*E+ D*B*W; j+=d*_*D-_*F*E; P=W*E*B-T*D; for (o+=(I=D*W+E
*T*B,E*d/K *B+v+B/K*F*D)*_; p<y; ){ T=p[s]+i; E=c-p[w]; D=n[p]-L; K=D*m-B*T-H*E; if(p [n]+w[ p]+p[s
]== 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T-a *E)> K)N=1e4; else{ q=W/K *4E2+2e2; C= 2E2+4e2/ K
 *D; N-1E4&& XDrawLine(e ,z,k,N ,U,q,C); N=q; U=C; } ++p; } L+=_* (X*t +P*M+m*l); T=X*X+ l*l+M *M;
  XDrawString(e,z,k ,20,380,f,17); D=v/l*15; i+=(B *l-M*r -X*Z)*_; for(; XPending(e); u *=CS!=N){
                                    XEvent z; XNextEvent(e ,&z);
                                        +++((N=XLookupKeysym
                                          (&z.xkey,0))-IT?
                                          N-LT? UP-N?& E:&
                                          J:& u: &h); --*(
                                          DN -N? N-DT ?N==
                                          RT?&u: & W:&h:&J
                                           ); } m=15*F/l;
                                           c+=(I=M/ l,l*H
                                           +I*M+a*X)*_; H
                                           =A*r+v*X-F*l+(
                                           E=.1+X*4.9/l,t
                                           =T*m/32-I*T/24
                                            )/S; K=F*M+(
                                            h* 1e4/l-(T+
                                            E*5*T*E)/3e2
                                            )/S-X*d-B*A;
                                            a=2.63 /l*d;
                                            X+=( d*l-T/S
                                             *(.19*E +a
                                             *.64+J/1e3
                                             )-M* v +A*
                                             Z)*_; l +=
                                             K *_; W=d;
                                             sprintf(f,
                                             "%5d  %3d"
                                             "%7d",p =l
                                            /1.7,(C=9E3+
                                  O*57.3)%0550,(int)i); d+=T*(.45-14/l*
                                 X-a*130-J* .14)*_/125e2+F*_*v; P=(T*(47
                                 *I-m* 52+E*94 *D-t*.38+u*.21*E) /1e2+W*
                                 179*v)/2312; select(p=0,0,0,0,&G); v-=(
                                  W*F-T*(.63*m-I*.086+m*E*19-D*25-.11*u
                                   )/107e2)*_; D=cos(o); E=sin(o); } }
```

Obfuscation makes a piece of code *unintelligible*

Basic idea: you can't observe anything about the internal workings of the program, it is a **black box**

Basic tricks of rewriting code don't actually seem to work

"Software Licensing"

```
#include                                    <math.h>
#include                                   <sys/time.h>
#include                                   <X11/Xlib.h>
#include                                  <X11/keysym.h>
                                           double L ,o ,P
                                          ,_=dt,T,Z,D=1,d,
                                          s[999],E,h= 8,I,
                                          J,K,w[999],M,m,O
                                         ,n[999],j=33e-3,i=
                                         1E3,r,t, u,v ,W,S=
                                         74.5,l=221,X=7.26,
                                         a,B,A=32.2,c, F,H;
                                         int N,q, C, y,p,U;
                                       Window z; char f[52]
                                     ; GC k; main(){ Display*e=
 XOpenDisplay( 0); z=RootWindow(e,0); for (XSetForeground(e,k=XCreateGC (e,z,0,0),BlackPixel(e,0))
; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400,
0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=sin(O)){ struct timeval G={ 0,dt*1e6}
; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_*P; r=E*K; W=cos( O); m=K*W; H=K*T; O+=D*_*F/ K+d/K*E*_; B=
sin(j); a=B*T*D-E*W; XClearWindow(e,z); t=T*E+ D*B*W; j+=d*_*D-_*F*E; P=W*E*B-T*D; for (o+=(I=D*W+E
*T*B,E*d/K *B+v+B/K*F*D)*_; p<y; ){ T=p[s]+i; E=c-p[w]; D=n[p]-L; K=D*m-B*T-H*E; if(p [n]+w[ p]+p[s
]== 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T-a *E)> K)N=1e4; else{ q=W/K *4E2+2e2; C= 2E2+4e2/ K
 *D; N-1E4&& XDrawLine(e ,z,k,N ,U,q,C); N=q; U=C; } ++p; } L+=_* (X*t +P*M+m*l); T=X*X+ l*l+M *M;
  XDrawString(e,z,k ,20,380,f,17); D=v/l*15; i+=(B *l-M*r -X*Z)*_; for(; XPending(e); u *=CS!=N){
                                     XEvent z; XNextEvent(e ,&z);
                                         ++*((N=XLookupKeysym
                                           (&z.xkey,0))-IT?
                                           N-LT? UP-N?& E:&
                                           J:& u: &h); --*(
                                           DN -N? N-DT ?N==
                                           RT?&u: & W:&h:&J
                                            ); } m=15*F/l;
                                            c+=(I=M/ l,l*H
                                            +I*M+a*X)*_; H
                                            =A*r+v*X-F*l+(
                                            E=.1+X*4.9/l,t
                                            =T*m/32-I*T/24
                                             )/S; K=F*M+(
                                             h* 1e4/l-(T+
                                             E*5*T*E)/3e2
                                             )/S-X*d-B*A;
                                             a=2.63 /l*d;
                                             X+=( d*l-T/S
                                              *(.19*E +a
                                              *.64+J/1e3
                                              )-M* v +A*
                                              Z)*_; l +=
                                              K *_; W=d;
                                              sprintf(f,
                                              "%5d  %3d"
                                              "%7d",p =l
                                             /1.7,(C=9E3+
                                  O*57.3)%0550,(int)i); d+=T*(.45-14/l*
                                 X-a*130-J* .14)*_/125e2+F*_*v; P=(T*(47
                                 *I-m* 52+E*94 *D-t*.38+u*.21*E) /1e2+W*
                                 179*v)/2312; select(p=0,0,0,0,&G); v-=(
                                  W*F-T*(.63*m-I*.086+m*E*19-D*25-.11*u
                                   )/107e2)*_; D=cos(o); E=sin(o); } }
```

Cryptographic Obfuscation

"I want to send you a program that is **provably** unintelligible"

A "virtual black box"

$\mathcal{O}(P)$

$$\mathcal{O}(P)(x) = P(x)$$

Cryptographic Obfuscation

"I want to send you a program that is **provably** unintelligible"

A "virtual black box"

$\mathcal{O}(P)$

$$\mathcal{O}(P)(x) = P(x)$$

Crucial insight: Alice can put a cryptographic key inside the box

On the (Im)possibility of Obfuscating Programs*

Boaz Barak† Oded Goldreich‡ Russell Impagliazzo§ Steven Rudich¶
Amit Sahai‖ Salil Vadhan** Ke Yang††

July 29, 2010

Abstract

Informally, an *obfuscator* $\mathcal{O}$ is an (efficient, probabilistic) "compiler" that takes as input a program (or circuit) P and produces a new program $\mathcal{O}(P)$ that has the same functionality as P yet is "unintelligible" in some sense. Obfuscators, if they exist, would have a wide variety of cryptographic and complexity-theoretic applications, ranging from software protection to homomorphic encryption to complexity-theoretic analogues of Rice's theorem. Most of these applications are based on an interpretation of the "unintelligibility" condition in obfuscation as meaning that $\mathcal{O}(P)$ is a "virtual black box," in the sense that anything one can efficiently compute given $\mathcal{O}(P)$, one could also efficiently compute given oracle access to P.

In this work, we initiate a theoretical investigation of obfuscation. Our main result is that, even under very weak formalizations of the above intuition, obfuscation is impossible. We prove this by constructing a family of efficient programs $\mathcal{P}$ that are *unobfuscatable* in the sense that (a) given *any* efficient program P' that computes the same function as a program $P \in \mathcal{P}$, the "source code" P can be efficiently reconstructed, yet (b) given *oracle access* to a (randomly selected) program $P \in \mathcal{P}$, no efficient algorithm can reconstruct P (or even distinguish a certain bit in the code from random) except with negligible probability.

We extend our impossibility result in a number of ways, including even obfuscators that (a) are not necessarily computable in polynomial time, (b) only approximately preserve the functionality, and (c) only need to work for very restricted models of computation ($\mathbf{TC^0}$). We also rule out several potential applications of obfuscators, by constructing "unobfuscatable" signature schemes, encryption schemes, and pseudorandom function families.

Keywords: cryptography, complexity theory, software protection, homomorphic encryption, Rice's Theorem, software watermarking, pseudorandom functions, statistical zero knowledge

*A preliminary version of this paper appeared in *CRYPTO'01* [BGI+].

†Department of Computer Science, Princeton University, NJ 08540. E-mail: boaz@cs.princeton.edu

‡Department of Computer Science, Weizmann Institute of Science, Rehovot, ISRAEL. E-mail: oded.goldreich@weizmann.ac.il

§Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093-0114. E-mail: russell@cs.ucsd.edu

¶Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave. Pittsburgh, PA 15213. E-mail: rudich@cs.cmu.edu

‖Department of Computer Science, UCLA, Los Angeles, CA 90095. Email: sahai@cs.ucla.edu

**School of Engineering and Applied Sciences and Center for Research on Computation and Society, Harvard University, 33 Oxford Street, Cambridge, MA 02138. E-mail: salil@seas.harvard.edu

††Google Inc., Moutain View, CA 94043. E-mail: yangke@gmail.com

Virtual black-box obfuscation does not exist in general (in the standard model)

Indistinguishability Obfuscation (iO)

$$i\mathcal{O}(P)(x) = P(x)$$

For any two *equivalent* programs P_0, P_1

$$\{P_0, P_1, i\mathcal{O}(P_0)\} \approx \{P_0, P_1, i\mathcal{O}(P_1)\}$$

Intuition: iO is a “pseudo-canonicalizer”

Indistinguishability Obfuscation (iO)

$$i\mathcal{O}(P)(x) = P(x)$$

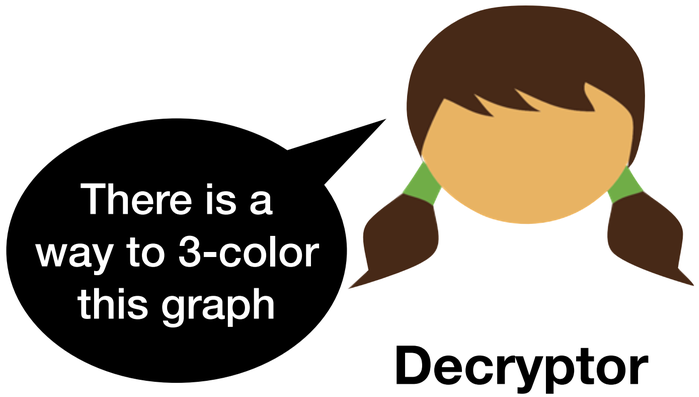
For any two *equivalent* programs P_0, P_1

$$\{P_0, P_1, i\mathcal{O}(P_0)\} \approx \{P_0, P_1, i\mathcal{O}(P_1)\}$$

Intuition: iO is a “pseudo-canonicalizer”

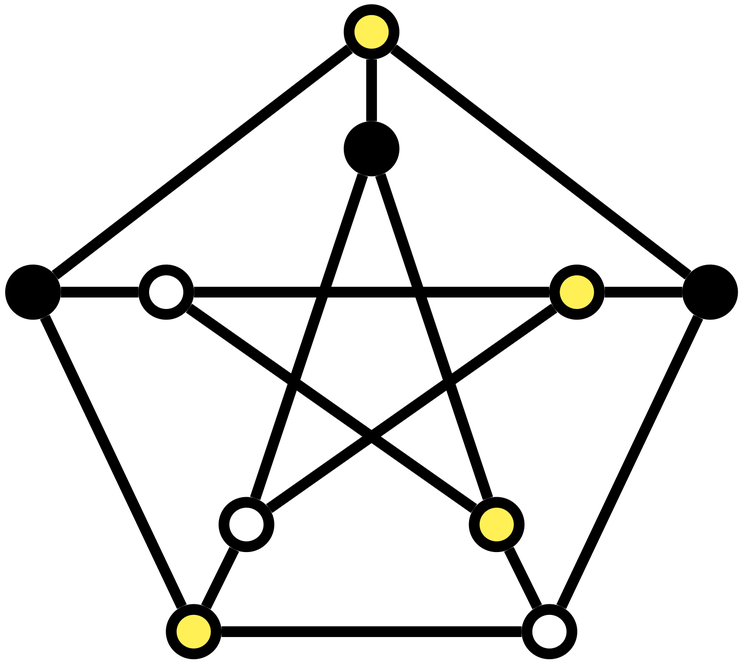
Fun Fact: if P = NP, then iO exists

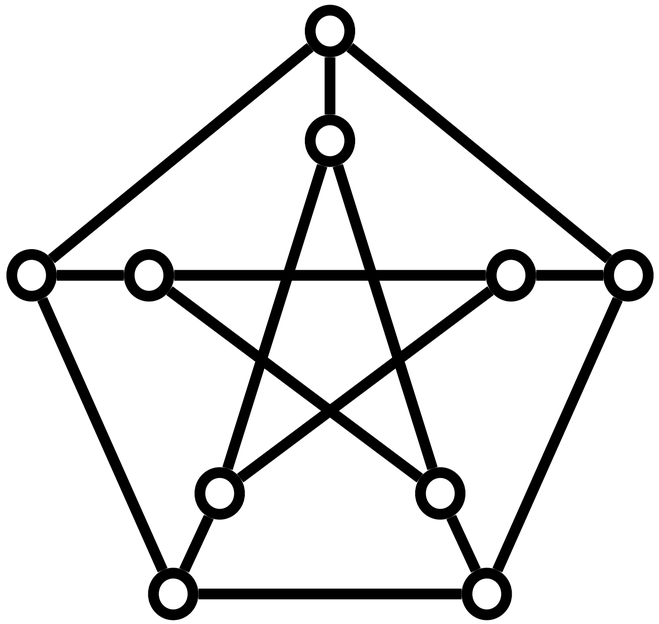
Witness Encryption



Decryptor

Encryptor





If the instance has a witness, the witness allows to decrypt

If the instance has no witness, secrecy holds

Applications of Indistinguishability Obfuscation (iO)
plus one-way functions

- Public Key Cryptography
- Digital Signatures
- Oblivious Transfer
- Short Zero Knowledge Proofs
- Fully Homomorphic Encryption
- Witness encryption
- Deniable encryption
- Functional encryption
- … and pretty much everything we have learned how to do in this course

Applications of Indistinguishability Obfuscation (iO)

plus one-way functions

- Public Key Cryptography
- Digital Signatures
- Oblivious Transfer
- Short Zero Knowledge Proofs
- Fully Homomorphic Encryption
- Witness encryption
- Deniable encryption
- Functional encryption
- … and pretty much everything we have learned how to do in this course
- … and pretty much everything anyone knows how to do with cryptography

Indistinguishability Obfuscation from Well-Founded Assumptions

Aayush Jain* Huijia Lin† Amit Sahai‡

November 12, 2020

Abstract

Indistinguishability obfuscation, introduced by [Barak et. al. Crypto'2001], aims to compile programs into unintelligible ones while preserving functionality. It is a fascinating and powerful object that has been shown to enable a host of new cryptographic goals and beyond. However, constructions of indistinguishability obfuscation have remained elusive, with all other proposals relying on heuristics or newly conjectured hardness assumptions.

In this work, we show how to construct indistinguishability obfuscation from subexponential hardness of four well-founded assumptions. We prove:

Theorem (Informal)**.** Let $\tau \in (0,\infty), \delta \in (0,1), \epsilon \in (0,1)$ be arbitrary constants. Assume sub-exponential security of the following assumptions, where λ is a security parameter, p is a λ-bit prime, and the parameters ℓ, k, n are large enough polynomials in λ:

- the Learning With Errors (LWE) assumption over $\mathbb{Z}_p$ with subexponential modulus-to-noise ratio 2^{k^ϵ}, where k is the dimension of the LWE secret,
- the Learning Parity with Noise (LPN) assumption over $\mathbb{Z}_p$ with polynomially many LPN samples and error rate $1/\ell^\delta$, where ℓ is the dimension of the LPN secret,
- the existence of a Boolean Pseudo-Random Generator (PRG) in NC^0 with stretch $n^{1+\tau}$, where n is the length of the PRG seed,
- the Symmetric eXternal Diffie-Hellman (SXDH) assumption on asymmetric bilinear groups of order p.

Then, (subexponentially secure) indistinguishability obfuscation for all polynomial-size circuits exists.

Further, assuming only polynomial security of the aforementioned assumptions, there exists collusion resistant public-key functional encryption for all polynomial-size circuits.

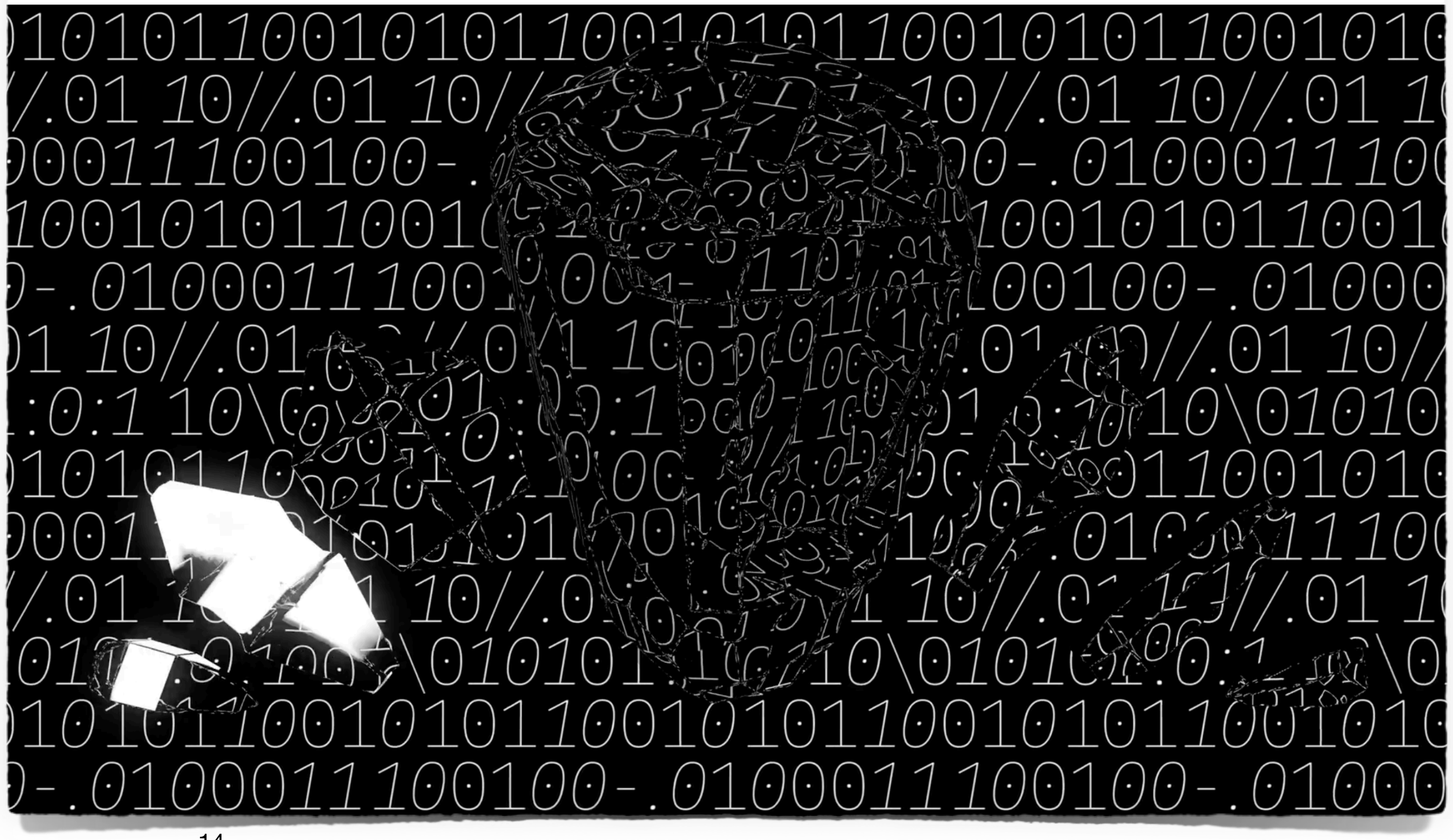
*UCLA, Center for Encrypted Functionalities, and NTT Research. Email: aayushjain@cs.ucla.edu.
†UW. Email: rachel@cs.washington.edu.
‡UCLA, Center for Encrypted Functionalities. Email: sahai@cs.ucla.edu.

COMPUTER SECURITY

Computer Scientists Achieve 'Crown Jewel' of Cryptography

18

A cryptographic master tool called indistinguishability obfuscation has for years seemed too good to be true. Three researchers have figured out that it can work.



Why do we care?

Current obfuscation techniques are not efficient

But they provide a tool for understanding complexity of a cryptographic problem

- If you can solve your problem with obfuscation, may mean an efficient solution exists
- If your definition implies obfuscation, you may be in trouble

Today’s objectives

Understand the notion of an obfuscation

Define indistinguishability obfuscation

Applications of iO, how does iO fit into cryptography