Obfuscation
Cryptography

• Theoretical cryptography provided theoretic foundations for most classical problems
  ▶ encryption, authentication, protocols

• Still cryptography challenges little theory

• One such problem program obfuscation
Cryptography
Cryptography

• Theoretical cryptography provided theoretic foundations for most *classical problems*
  ▶️ *encryption, authentication, protocols*

• Still cryptography *challenges* little theory

• One such problem *program obfuscation*
Theoretical cryptography provided theoretic foundations for most classical problems.
- encryption, authentication, protocols

Still cryptography challenges little theory.

One such problem program obfuscation

What is obfuscation?
Obfuscation
Obfuscation

• Goal program obfuscation: make a program unintelligible while preserving functionality
Obfuscation

• Goal program obfuscation: make a program unintelligible while preserving functionality

• Ideally a virtual black box: anything one can compute from it also computed from input-output behaviour
Situation 1

Alice

Input

Black Box

Output
Situation 2

Bob

Input → Obfuscated Program → Output
Obfuscation

• An obfuscator $O$ is an (efficient, probabilistic) “compiler” that takes as input a program $P$ and produces a new program $O(P)$ satisfying the following two conditions:
Obfuscation

• An obfuscator $O$ is an (efficient, probabilistic) “compiler” that takes as input a program $P$ and produces a new program $O(P)$ satisfying the following two conditions:

1. $O(P)$ computes the same function as $P$
Obfuscation

• An obfuscator \( O \) is an (efficient, probabilistic) “compiler” that takes as input a program \( P \) and produces a new program \( O(P) \) satisfying the following two conditions:

1. \( O(P) \) computes the same function as \( P \)

2. Anything that can be efficiently computed from \( O(P) \) can be efficiently computed given oracle access to \( P \)
Obfuscation

- An obfuscator $O$ is an (efficient, probabilistic) “compiler” that takes as input a program $P$ and produces a new program $O(P)$ satisfying the following two conditions:

1. $O(P)$ computes the same function as $P$

2. Anything that can be efficiently computed from $O(P)$ can also be efficiently computed given oracle access to $P$.

Why obfuscation?
Windows Security Update
Windows Security Update
Windows Security Update

Microsoft

UPDATE

Mallory
Windows Security Update

Mallory
Windows Security Update

Reverse engineering patches

Mallory
Windows Security Update

Mallory
Windows Security Update

Alice → Mallory

Bob
Windows Security Update

Should we be worried?
Windows Security Update

- Less than 30 minutes, flaw fixed by a Microsoft update to the Secure Sockets Layer (SSL)

- A reliable exploit for the flaw was created in less than 10 hours

- Less than three hours, Microsoft corrected vulnerability in the Internet Security and Acceleration (ISA) Server, missed same vulnerability other parts of the system
Windows Security Update
Windows Security Update
Windows Security Update

Is this possible?
In Practice

• This is done a lot

• Not standard

• Relies on human ingenuity, security-via-obscurity

• “At best, obfuscation merely makes it time-consuming, but not impossible, to reverse engineer a program”
In Practice

• This is done a lot

• Not standard

• Relies on human ingenuity, security-via-obscurity

• “At best, obfuscation merely makes it time-consuming, but not impossible, to reverse engineer a program.”

Can we formalise it?
Barak

On the (Im)possibility of Obfuscating Programs

• Intuition: can’t be, it’s too strong

• Black Box Paradigm: any function $f(\cdot)$ should be obfuscated $O(f(\cdot))$

• Barak constructed functions that are unobfuscatable
• Virtual’ black box paradigm, even weak formalisations, impossible

• Families of efficient programs \( \mathcal{P} \) unobfuscatable

• \textbf{Given} any efficient program \( P' \) computing same function as \( P \in \mathcal{P} \), the “source code” of \( P \) can be efficiently reconstructed

• \textbf{Only oracle access} to \( P \in \mathcal{P} \), no efficient algorithm reconstruct \( P \), except negligible probability
• Remained impossible even when including obfuscators that
  1. Are not necessarily computable in polynomial time
  2. Only approximately preserve the functionality
  3. Only need to work for very restricted models of computation.
Indistinguishability obfuscators not ruled out by impossibility result
Garg

Candidate Indistinguishability Obfuscation and Functional Encryption for all circuits

- Indistinguishable obfuscation
  - Barak showed this is best possible obfuscation
  - Suffices for many applications
Indistinguishable obfuscation

- An indistinguishability obfuscator $iO$ for a class of circuits $C$

- Equivalent circuits $C_1$ and $C_2$ from same class, compute same function and $|C_1| = |C_2|$

- Distribution of obfuscations $iO(C_1)$ and $iO(C_2)$ be computationally indistinguishable
Multilinear Jigsaw Puzzle
Multilinear Jigsaw Puzzle
Multilinear Jigsaw Puzzle

Indistinguishably obfuscated program
Multilinear Jigsaw Puzzle

- Variant of multilinear maps
- Offers only a strict subset of functionality
- Similar to encoding scheme (GGH) from 1997
Multilinear Jigsaw Puzzle

• Jigsaw generator

• Jigsaw verifier \( e : G_1 \times G_2 \times \cdots \times G_k \rightarrow G_T \)

• Perform group and multilinear operations

• Valid multilinear form anything can be computed using group operation in separate groups and multilinear map
Multilinear Jigsaw Puzzle

• Jigsaw generator outputs some system parameters prms and k+1 nonempty sets of elements $S_i = \{x_{1(i)}, \ldots, x_{n_i(i)}\} \subset G_i$ for $i = 1, \ldots, k, T$

• The Jigsaw verifier takes as input:
  $(\text{prms}, S_1, \ldots, S_k, S_T, \prod)$

• Where $\prod$ and is a valid multilinear form in these elements
  • “yes” if $\prod$ evaluates to the unit element in $G_T$ on the given elements
  • “no” otherwise
Shai Halevi

• We are in the midst of (yet another) “quantum leap” in our cryptographic capabilities

• Things that were science fiction just two years ago are now plausible

• Fuelled by new powerful building blocks
  
  • Combination of Homomorphic Encryption (HE)
  
  • Cryptographic Multilinear Maps (MMAPs)
We are in the midst of (yet another) “quantum leap” in our cryptographic capabilities

Things that were science fiction just two years ago are now plausible

Fuelled by new powerful building blocks

- Combination of Homomorphic Encryption (HE)
- Cryptographic Multilinear Maps (MMAPs)

Are we done?
Future research

• "... far from ready for commercial applications. The technique turns short, simple programs into giant, unwieldy albatrosses."
Future research

- Better underlying hardness assumptions
- Faster constructions (horrendous complexity)
- Better notions
  - Not capture intuition of what obfuscator is
Take Home Message

• Obfuscation of general programs in ‘virtual black box’ paradigm is impossible

• The weaker notion of indistinguishable obfuscation can be achieved for polynomial sized circuits
Sources


• Damien Stehlé and Ron Steinfeld. Making ntru as secure as worst-case problems over ideal lattices. p. 27–47, 2011.

• Shai Halevi – IBM Research, Multilinear Maps and Obfuscation
Thanks for your attention!

- Questions
- Discussion
- Appendix
Discussion

• What challenges remain in cryptography?
Discussion

• How is it possible that one can reverse engineer two unobfuscatable programs, but not distinguish between the two?
Functional encryption

- Send encrypted function instead of message
- Revealed information depends on receiver
Functional encryption

- Hospital treatment information
  - Share anonymised with researchers
  - Share with GP
  - Share with insurance
Software Protection

• By definition, obfuscating a program protects it against reverse engineering.

• For example, if one party, Alice, discovers a more efficient algorithm for factoring integers, she may wish to sell another party, Bob, a program for apparently weaker tasks (such as breaking RSA) that use the factoring algorithm as a subroutine without actually giving Bob a factoring algorithm. Alice could hope to achieve this by obfuscating the program she gives to Bob.

• Obfuscators would also be useful in watermarking software. A software vendor could modify a program’s behaviour in a way that uniquely identifies the person to whom it is sold, and then obfuscate the program to guarantee that this “watermark” is difficult to remove.
Software Patching

- If a new malware vulnerability is found in software, there is a risk that releasing a software patch will allow attackers to become aware of a vulnerability before the patch has a chance to fully circulate among users. Obfuscation offers a solution concept: an initial patch can be released in obfuscated form, and then transitioned to a more efficient un-obfuscated patch once large-scale adoption has occurred for the initial patch. Here, the assumption would be that the obfuscated patch would hide where the vulnerability in the software was (at least as well as the original vulnerable software did).
Private to Public

• Obfuscation can also be used to create new public-key encryption schemes by obfuscating a private-key encryption scheme. Given a secret key $K$ of a private-key encryption scheme, one can publish an obfuscation of the encryption algorithm $\text{Enc}K$. This allows everyone to encrypt, yet only one possessing the secret key $K$ should be able to decrypt.

• Interestingly, in the original paper of Diffie and Hellman [DH], the above was the reason given to believe that public-key cryptosystems might exist even though there were no candidates known yet. That is, they suggested that it might be possible to obfuscate a private-key encryption scheme
Homomorphic Encryption

• A long-standing open problem in cryptography is whether homomorphic encryption schemes exist (cf., [RAD, FM, DDN, BL, SYY]). That is, we seek a secure public-key cryptosystem for which, given encryptions of two bits (and the public key), one can compute an encryption of any binary Boolean operation of those bits.

• Obfuscators would allow one to convert any public-key cryptosystem into a homomorphic one, by using ideas as in the previous paragraph. Specifically, use the secret key to construct an algorithm that performs the required computations (by decrypting, applying the Boolean operation, and encrypting the result), and publish an obfuscation of this algorithm along with the public key.
Boolean Circuit

• Jigsaw is a mathematical construction
• Garg now only allows for AND, OR, NOT gates
• Not functioning as a Turing machine