Implementing Cryptographic Program Obfuscation

Daniel Apon\textsuperscript{1} Yan Huang\textsuperscript{1} Jonathan Katz\textsuperscript{1}
\textbf{Alex J. Malozemoff}\textsuperscript{1}

\textsuperscript{1}University of Maryland

Presented at CRYPTO Rump Session, Santa Barbara, California, USA, August 17–21, 2014.
Everybody loves (virtual black-box / indistinguishability) obfuscation...
Everybody loves (virtual black-box / indistinguishability) obfuscation...so we implemented it!

Implementation combines ideas from various obfuscation papers and uses CLT multilinear map scheme. It is slow...but not as slow as you might think.

Example: To obfuscate a 16-bit point function (i.e., 16 OR gates) with 52 bits of security using an Amazon EC2 machine with 32 cores:

- Obfuscation time: \( \approx 7 \) hours
- Evaluation time: \( \approx 3 \) hours
- Obfuscation size: 31 GB

⇒ it's almost nearly practical
Everybody loves (virtual black-box / indistinguishability) obfuscation... so we implemented it!

Implementation combines ideas from various obfuscation papers and uses CLT multilinear map scheme
Everybody loves (virtual black-box / indistinguishability) obfuscation... so we implemented it!

Implementation combines ideas from various obfuscation papers and uses CLT multilinear map scheme

It is slow...
Everybody loves (virtual black-box / indistinguishability) obfuscation... so we implemented it!

Implementation combines ideas from various obfuscation papers and uses CLT multilinear map scheme

It is slow... but not as slow as you might think
Everybody loves (virtual black-box / indistinguishability) obfuscation... so we implemented it!

Implementation combines ideas from various obfuscation papers and uses CLT multilinear map scheme

It is slow... but not as slow as you might think

**Example:** To obfuscate a 16-bit point function (i.e., 16 OR gates) with 52 bits of security using an Amazon EC2 machine with 32 cores:
Everybody loves (virtual black-box / indistinguishability) obfuscation... so we implemented it!

Implementation combines ideas from various obfuscation papers and uses CLT multilinear map scheme

It is slow... but not as slow as you might think

**Example:** To obfuscate a 16-bit point function (i.e., 16 OR gates) with 52 bits of security using an Amazon EC2 machine with 32 cores:

- Obfuscation time: \( \approx 7 \) hours
Everybody loves (virtual black-box / indistinguishability) obfuscation... so we implemented it!

Implementation combines ideas from various obfuscation papers and uses CLT multilinear map scheme

It is slow... but not as slow as you might think

Example: To obfuscate a 16-bit point function (i.e., 16 OR gates) with 52 bits of security using an Amazon EC2 machine with 32 cores:

- Obfuscation time: $\approx 7$ hours
- Evaluation time: $\approx 3$ hours
Everybody loves (virtual black-box / indistinguishability) obfuscation... so we implemented it!

Implementation combines ideas from various obfuscation papers and uses CLT multilinear map scheme

It is slow... but not as slow as you might think

Example: To obfuscate a 16-bit point function (i.e., 16 OR gates) with 52 bits of security using an Amazon EC2 machine with 32 cores:

- Obfuscation time: $\approx 7$ hours
- Evaluation time: $\approx 3$ hours
- Obfuscation size: 31 GB
Everybody loves (virtual black-box / indistinguishability) obfuscation...so we implemented it!

Implementation combines ideas from various obfuscation papers and uses CLT multilinear map scheme

It is slow...but not as slow as you might think

Example: To obfuscate a 16-bit point function (i.e., 16 OR gates) with 52 bits of security using an Amazon EC2 machine with 32 cores:

- Obfuscation time: $\approx 7$ hours
- Evaluation time: $\approx 3$ hours
- Obfuscation size: 31 GB

$\Rightarrow$ it’s almost nearly practical
Code is available: https://github.com/amaloz/ind-obfuscation
Code is available: https://github.com/amaloz/ind-obfuscation

ePrint version should be up at some point
Code is available: https://github.com/amaloz/ind-obfuscation

ePrint version should be up at some point

For the cryptanalysts in the audience: We have an obfuscated 14-bit point function on Dropbox$^1$ — learn the point and you win!

$^1$https://www.dropbox.com/s/85d03o0ny3b1c0c/point-14.circ.obf.60.zip
Code is available: https://github.com/amaloz/ind-obfuscation

ePrint version should be up at some point

For the cryptanalysts in the audience: We have an obfuscated 14-bit point function on Dropbox¹ — learn the point and you win!

Contact info: {dapon,yhuang,jkatz,amaloz}@cs.umd.edu

Thank you

¹https://www.dropbox.com/s/85d03o0ny3b1c0c/point-14.circ.obf.60.zip