Compilers for Zero-Knowledge: An Overview

Yuval Ishai
Technion

9th BIU Winter School on Cryptography
Broad Motivation

• ZK research is a big party
  – Many motivating applications
  – Many challenging questions
  – Many exciting results

• Big party → Big mess?

• This talk: advocating a modular approach
  – Separate “information-theoretic” and “crypto” parts
  – General cryptographic compilers (IT → crypto)
  – General information-theoretic compilers (IT → IT)
NP relation $R(x,w)$

**Convenient Representation**

**Computational model**

**Information-Theoretic Proof System**

“ZK-PCP”

**Crypto assumptions / Generic models**

**crypto compiler**

**ZK Proof/Argument**

- Boolean circuit
- Arithmetic circuit
- RAM
- QSP, QAP, SSP
- R1CS
- TinyRAM

Different kinds (coming up)
NP relation $R(x,w)$

Representation

Computational model

Information-Theoretic Proof System

“ZK-PCP”

crypto compiler

ZK Proof/Argument

MPC protocols

IT Compilers

Carmit’s talk
Why?

• Simplicity
  – Break complex tasks into simpler components
  – Easier to analyze and optimize
  – Potential for proving lower bounds

• Generality
  – Apply same constructions in different settings
  – Research deduplication, less papers to read/write

• Efficiency
  – Port efficiency improvements between settings
  – Mix & match different components
  – Systematic exploration of design space
ZK Zoo
(ignoring assumptions for now…)

Qualitative features
• Interactive?
• Succinct?
• Fast verification?
• Public verification?
• Public input?
• NP vs. P?
• Trusted setup?
• Symmetric crypto only?
• Post quantum?

Quantitative features
• Communication
• Prover complexity
• Verifier complexity

Major commercialization efforts
Standardization process
zkproof.org
2nd workshop: April 10-12

Optimal ZKP protocol?
Food for thought…

• Which verifier is better?
  – V1: SHA256 hash
  – V2: PKE decryption

• V2 can be more obfuscation-friendly! [BISW17]
  – Relevant complexity measure: branching program size
  – Motivated “lattice-based” designated-verifier SNARKs
  – Promising avenue for practical general-purpose obfuscation

• Similar: MPC-friendly prover, etc.
Back to 20th Century
Theorem [GMW86]:
Bit-commitment $\rightarrow$ ZKP for all of NP

Theorem [GMW86+Naor89+HILL99]:
One-way function $\rightarrow$ ZKP for all of NP

Theorem [OW93]:
ZKP for “hard on average” $L$ in NP $\rightarrow$ i.o. one-way function

Are we done?
ZKP for 3-Colorability

[GMW86]

• Prover wants to prove that a given graph is 3-colorable
ZKP for 3-Colorability

• Prover wants to prove that a given graph is 3-colorable
  – $x=\text{graph } w=\text{coloring}$
ZKP for 3-Colorability

- Prover randomly permutes the 3 colors (6 possibilities)
  - Say,
ZKP for 3-Colorability

• Prover randomly permutes the 3 colors (6 possibilities)
  – Say, \[ \text{permuted colors: teal} \rightarrow \text{blue} \rightarrow \text{green} \rightarrow \text{red} \]
ZKP for 3-Colorability

- Prover separately commits to color of each node and sends commitments to Verifier
ZKP for 3-Colorability

- Verifier challenges Prover by selecting a random edge
ZKP for 3-Colorability

- Prover sends decommitments for opening the colors of the two nodes
ZKP for 3-Colorability

- Verifier accepts if both colors are valid and are distinct (otherwise it rejects).

- Repeat $O(|E|)$ times to amplify soundness.
Issues

• Security proof more subtle than it may seem
  – Need to redo analysis of Hamiltonicity-based ZK?
• Two sources of inefficiency
  – Karp reduction
  – Soundness amplification (+ many rounds)
Abstraction to the rescue…
Information-Theoretic Proof System: ZK-PCP

Prover: \((x,w) \rightarrow \pi\)

\[
\pi = \begin{bmatrix}
1 & 3 & 1 & 2 & 1 & 3 & 1 & 2 & 1 & 1 & 3 & 1 & 3 & 1 & 2 & 1
\end{bmatrix}
\]
Information-Theoretic Proof System: ZK-PCP

Prover: \((x, w) \rightarrow \pi\)

\[ \pi = \begin{array}{cccccccccccccc}
1 & 3 & 1 & 2 & 1 & 3 & 1 & 2 & 1 & 1 & 3 & 1 & 3 & 1 & 2 & 1
\end{array} \]

Verifier

- Simple security definition
  - Completeness
  - Perfect (public-coin) ZK
  - Soundness error \(\epsilon\)
    (amplified via parallel repetition)

- Clean efficiency measures
  - Alphabet size
  - Query complexity
  - Prover computation
  - Verifier computation
Information-Theoretic Proof System: ZK-PCP

Prover: \((x, w) \rightarrow \pi\)

\[ \pi = 1 \ 3 \ 1 \ 2 \ 1 \ 3 \ 1 \ 2 \ 1 \ 1 \ 3 \ 1 \ 3 \ 1 \ 2 \ 1 \]

Verifier

Crypto compilers

ZK in plain model

NIZK in ROM

[GMW86, PW99] + Stat-binding commit

[FS86, Mic00] + Random oracle

[GMW86, PW99] + Stat-binding commitment

[GK96] + Stat-hiding commitment
Information-Theoretic Proof System: \textbf{ZK-PCP}

Prover: \((x, w) \rightarrow \pi\)

\[
\pi = [1, 3, 1, 2, 1, 3, 1, 2, 1, 1, 3, 1, 3, 1, 2, 1]
\]

Veriﬁer

\textbf{Crypto compilers}

\textbf{ZK in plain model}

\textbf{NIZK in CRS model}

Ron’s talk:

\textbf{NIZK in Hidden Bits Model}

+Stat-binding commit

\textbf{GMW86, PW99}

+Stat-hiding commit

\textbf{GK96}

+Trapdoor permutation

\textbf{FLS90}

\textbf{Stat-binding commit}
Information-Theoretic Proof System: ZK-PCP

Prover: \((x,w) \rightarrow \pi\)

\[\pi = 1 \ 3 \ 1 \ 2 \ 1 \ 3 \ 1 \ 2 \ 1 \ 1 \ 3 \ 1 \ 3 \ 1 \ 2 \ 1\]

Better parameters?
Simpler?
Less "magical"?
IT Compilers:
MPC $\rightarrow$ ZK-PCP
Given MPC protocol for $f(w_1, \ldots, w_n) = R(w_1 \oplus \ldots \oplus w_n)$
Applications

• Simple ZK proofs using:
  – (2,5) or (1,3) semi-honest MPC [BGW88, CCD88, Maurer02]
  – (2,3) or (1,2) semi-honest MPC$^{\text{OT}}$ [Yao86, GMW87, GV87, GHY87]
  – Practical! [GMO16, CDG+17, KKW18] ➔ post-quantum signatures!

• ZK proofs with $O(|R|) + \text{poly}(k)$ communication
  – MPC from AG codes [CC05, DI05]

• Many good ZK protocols implied by MPC literature
  – MPC for linear algebra [CD01,…]
  – MPC over rings [CFIK03] or groups [DPSW07, CDI+13]

• Going (somewhat) sublinear! [AHIV17] – Carmit’s talk
Going fully sublinear
Traditional PCPs

- $x \in L \implies \exists \pi \Pr[\text{Verifier accepts } \pi] = 1$
- $x \notin L \implies \forall \pi^* \Pr[\text{Verifier accepts } \pi^*] \leq 1/2$

PCP Theorem [AS, ALMSS, Dinur]:
NP statements have polynomial-size PCPs in which the verifier reads only $O(1)$ bits.
- Can be made ZK with small overhead [KPT97, IW04]
Still need crypto compiler…

Verifier

Input $x$

Prover

ZK-PCP $\pi \in \{0, 1\}^{\text{poly}(|x, w|)}$

$q_1, q_2, q_3$

$\pi_{q_1}, \pi_{q_2}, \pi_{q_3}$

ACC/REJ
Crypto Compiler
[Kil93, Mic94]

Merkle Tree construction

\[ H = \text{collision resistant hash function} \]
\[ H: \{0,1\}^* \rightarrow \{0,1\}^k \]

\[ \pi_1 \pi_2 \pi_3 \pi_4 \ldots \pi_m \]

\[ H \rightarrow H \rightarrow H \rightarrow H \rightarrow H \rightarrow H \rightarrow H \rightarrow \text{com} \]
Limitations

- PCP Encoding
  - Witness
  - Computationally Heavy!

- Cryptographic Hashing
  - Potential workaround [LM18, BBF18]
  - Sub-optimally succinct
  - + opening PCP queries
Relaxing PCP model 1: Interaction

\[ \pi_1 = 1 \ 3 \ 1 \ 2 \ 1 \ 3 \ 1 \ 2 \ 1 \ 1 \ 3 \ 1 \ 3 \ 1 \ 2 \ 1 \]

Interactive PCP [KR08,GIMS10]
IOP [BCS16,RRR16]
Relaxing PCP model 2: Linear PCP

[ALMSS98,IKO07,BCIO13]

over a (large) finite field $F$

$\pi =$

\[
\begin{array}{cccccccccccccccc}
4 & 3 & 1 & 2 & 8 & 3 & 1 & 2 & 1 & 9 & 3 & 1 & 6 & 1 & 2 & 1 \\
\end{array}
\]

inner product

$q_1 =$

\[
\begin{array}{cccccccccccccccc}
5 & 3 & 6 & 2 & 1 & 3 & 1 & 2 & 1 & 1 & 6 & 1 & 3 & 1 & 8 & 1 \\
\end{array}
\]

$q_2 =$

\[
\begin{array}{cccccccccccccccc}
7 & 3 & 1 & 2 & 4 & 3 & 1 & 2 & 7 & 1 & 3 & 1 & 7 & 1 & 2 & 1 \\
\end{array}
\]

$q_3 =$

\[
\begin{array}{cccccccccccccccc}
1 & 2 & 1 & 2 & 1 & 9 & 1 & 2 & 5 & 1 & 4 & 1 & 3 & 1 & 3 & 1 \\
\end{array}
\]

Prover

Verifier

$a_1$

\[
\begin{array}{c}
\text{x}
\end{array}
\]

$a_2$

ACC / REJ

$a_3$
Advantages of Linear PCPs

• Simple!
  – Hadamard PCP: $\pi = (W, W \times W)$

• Short, efficiently computable
  – $O(|C|)$-size, quasi-linear time via QSP/QAP [GGPR13, …]

• Negligible soundness error with $O(1)$ queries
  – Reusable soundness
    $\Pr[\pi^* \text{ is accepted}]$ is either 1 or $O(1/|F|)$
  – Maximal succinctness
  – In fact, 1 query is enough! [BCIOP13]
Crypto Compilers for Linear PCPs

- **First generation** [IKO07, GI10, Gro10, SMBW12,…]
  - Standard assumptions
    - Linearly homomorphic encryption, discrete log
  - Interactive, one-way-succinct/somewhat succinct
  - **Idea**: use succinct vector-commitment with linear opening

- **Second generation** [Gro10, Lip12, GGPR13, BCIOP13,…]
  - Strong “knowledge” or “targeted malleability” assumptions
  - Non-interactive using a (long, structured) CRS
  - Publicly verifiable via pairings
  - **Idea**: include “encrypted queries” in CRS
Crypto Compiler: First Attempt

Prover

\[ \pi = 4 \ 3 \ 1 \ 2 \ 8 \ 3 \ 1 \ 2 \ 1 \ 9 \ 3 \ 1 \ 6 \ 1 \ 2 \ 1 \]

Verifier

\[ q_1 = 5 \ 3 \ 6 \ 2 \ 1 \ 3 \ 1 \ 2 \ 1 \ 1 \ 6 \ 1 \ 3 \ 1 \ 8 \ 1 \]
\[ q_2 = 7 \ 3 \ 1 \ 2 \ 4 \ 3 \ 1 \ 2 \ 7 \ 1 \ 3 \ 1 \ 7 \ 1 \ 2 \ 1 \]
\[ q_3 = 1 \ 2 \ 1 \ 2 \ 1 \ 9 \ 1 \ 2 \ 5 \ 1 \ 4 \ 1 \ 3 \ 1 \ 3 \ 1 \]

a_1  
\[ \text{x} \]

a_2  

a_3  

ACC / REJ
Crypto Compiler: First Attempt

CRS

\[ q_1 = \begin{bmatrix} 5 & 3 & 6 & 2 & 1 & 3 & 1 & 2 & 1 & 1 & 6 & 1 & 3 & 1 & 8 & 1 \\ 7 & 3 & 1 & 2 & 4 & 3 & 1 & 2 & 7 & 1 & 3 & 1 & 7 & 1 & 2 & 1 \\ 1 & 2 & 1 & 2 & 1 & 9 & 1 & 2 & 5 & 1 & 4 & 1 & 3 & 1 & 3 & 1 \end{bmatrix} \]

Prover

\[ \pi = \begin{bmatrix} 4 & 3 & 1 & 2 & 8 & 3 & 1 & 2 & 1 & 9 & 3 & 1 & 6 & 1 & 2 & 1 \end{bmatrix} \]

Verifier

\[ a_1 \]
\[ a_2 \]
\[ a_3 \]

x

ACC / REJ
Crypto Compiler: First Attempt

**CRS**

\[ q_1 = \]
\[ q_2 = \text{linearly homomorphic encryption} \]
\[ q_3 = \]

**Prover**

\[ \pi = 4 3 1 2 8 3 1 2 1 9 3 1 6 1 2 1 \]

**Verifier**

\[ a_1 \]
\[ a_2 \]
\[ a_3 \]
\[ x \]
Crypto Compiler: First Attempt

CRS

\[ q_1 = \]
\[ q_2 = \text{linearly homomorphic encryption} \]
\[ q_3 = \]

Prover

\[ \pi = \]

Problem 1: May allow more than just linear functions!

Solution 1: Assume it away: “linear-only encryption”
- A natural instance of targeted malleability [BSW12]
- Plausible for most natural public-key encryption schemes … including post-quantum ones [Reg05,BISW17]
- Win-win flavor
Problem 2: Prover can apply different $\pi_i$ to each $q_i$ or even combine $q_i$

Solution 2: Compile LPCP into a proof system that resists this attack
- Linear Interactive Proof (LIP): 2-message IP with “linear-bounded” Prover
- IT compiler: LPCP $\rightarrow$ LIP via a random consistency check [BCIOP13]
Problem 3: Only works in a designated-verifier setting

Solutions 3:
- Look for designated verifiers around your neighborhood
- LPCP with deg-2 decision + “bilinear groups” $\rightarrow$ public verification [Gro00,BCIOP03]
Combining the Two Relaxations: Linear IOP

Prover

\( \pi_1 = 1 \ 3 \ 1 \ 2 \ 1 \ 3 \ 1 \ 2 \ 1 \ 1 \ 3 \ 1 \ 3 \ 1 \ 2 \ 1 \)

\( q_1 = 5 \ 3 \ 6 \ 2 \ 1 \ 3 \ 1 \ 2 \ 1 \ 1 \ 6 \ 1 \ 3 \ 1 \ 8 \ 1 \)

Verifier

Challenge

\( \pi_2 = 1 \ 3 \ 1 \ 2 \ 1 \ 3 \ 1 \ 2 \ 1 \ 1 \ 3 \ 1 \ 3 \ 1 \ 2 \ 1 \)

\( q_2 = 7 \ 3 \ 1 \ 2 \ 4 \ 3 \ 1 \ 2 \ 7 \ 1 \ 3 \ 1 \ 7 \ 1 \ 2 \ 1 \)

Challenge

Implicit in interactive proofs for P
[GKR08, RRR16]
Suppose statement $x$ is known to prover but is
- Secret-shared between two or more verifiers
- Partitioned between two or more verifiers

Goal: strong ZK, hiding $x$ as well

Tool: fully linear ZK proof systems
- Only allow linear access to $x$: $q_i$ applies jointly to $(x, \pi)$
- Can be naturally compiled to ZK in above settings
  - Also with linearly encrypted or committed input
  - Implicitly used in previous systems [BGI16,CB17]
Fully Linear PCP/IOP
[BBCGI19]

• Constructions: NP languages
  – Standard LPCPs for NP are fully linear, but big proofs
  – Meaningful also for “simple” languages in P!

• Sublinear-size proofs for “simple” languages
  – Implicit in interactive proofs [GKR08, RRR16, NPY18]
  – New constructions for low-degree polynomials
    • E.g., test that $x \in F^n$ is in $\{0,1\}^n$
Conclusions

- Modular approach to efficient ZKP design
  - Information-theoretic ZK-PCP + crypto compiler
    - point queries vs. linear queries
    - non-interactive vs. interactive
- Applies to most efficient ZKP from the literature
  - In a sense inherent to “black-box” constructions [RV09]
  - but not to non-bb constructions [Val09,BCCT13,BCTV14]
- Lots of room for further progress
  - Better PCPs (and lower bounds)
  - Better crypto compilers
  - Better IT compilers
The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Program under grant agreement no. 742754 – ERC – NTSC