Lecture 11: From Practice to Theory

Guest Lecturer: Alex Lombardi

Zero Knowledge Proofs

Instructors: Dan Boneh, Shafi Goldwasser, Dawn Song, Justin Thaler, Yupeng Zhang
Authentication  Blockchains and cryptocurrencies  secure multiparty computation

Cryptographic Proofs
What does theoretical research on proof systems look like?
Theoretical Research on Cryptographic Proofs

Feasibility (do they exist in principle?)

- SNAR(G/K)s, other protocols (ZK, WI, WH, etc.)
- Strong attack models (Concurrent? Quantum?)
Theoretical Research on Cryptographic Proofs

Feasibility (do they exist in principle?)

- SNAR(G/K)s, other protocols (ZK, WI, WH, etc.)
- Strong attack models (Concurrent? Quantum?)

Minimize Assumptions (to the extent possible)

- Trusted setup (CRS/URS/plain model)
- Security reduction based on simple, well-studied, falsifiable assumptions.
Theoretical Research on Cryptographic Proofs

Feasibility (do they exist in principle?)
- SNAR(G/K)s, other protocols (ZK, WI, WH, etc.)
- Strong attack models (Concurrent? Quantum?)

Minimize Assumptions (to the extent possible)
- Trusted setup (CRS/URS/plain model)
- Security reduction based on simple, well-studied, falsifiable assumptions.

Improve efficiency
- Amount of communication, number of rounds
- Prover/verifier efficiency
Theoretical Research on Cryptographic Proofs

Feasibility (do they exist in principle?)
- SNAR(G/K)s, other protocols (ZK, WI, WH, etc.)
- Strong attack models (Concurrent? Quantum?)

Minimize Assumptions (to the extent possible)
- Trusted setup (CRS/URS/plain model)
- Security reduction based on simple, well-studied, falsifiable assumptions.

Improve efficiency
- Amount of communication, number of rounds
- Prover/verifier efficiency

+Applications
Example: Interactive ZK
Interactive Zero-Knowledge Protocols

- No trusted setup allowed.
  - Security against Malicious verifier hard to guarantee.
Interactive Zero-Knowledge Protocols

- No trusted setup allowed.
  - Security against Malicious verifier hard to guarantee.

- Lecture 1: ZK for NP [GMW86] with inverse poly soundness error. How do we reduce the error?
Interactive Zero-Knowledge Protocols

- No trusted setup allowed.
  - Security against Malicious verifier hard to guarantee.

- Lecture 1: ZK for NP [GMW86] with inverse poly soundness error. How do we reduce the error?
  - Sequential repetition works (but very inefficient).
  - Parallel repetition reduces soundness error but *may not* preserve ZK! Let’s see why:
Zero Knowledge Proofs for NP

Claim: This graph has a 3-coloring.
Zero Knowledge Proofs for NP

1) Randomize colors

13
Zero Knowledge Proofs for NP

1) Randomize colors
2) Commit
Zero Knowledge Proofs for NP

1) Randomize colors
2) Commit

1) Sample a challenge edge.
Zero Knowledge Proofs for NP

2) Commit

3) Reveal edge colors

1) Sample a challenge edge.

2) Accept if colors are different.
Zero Knowledge Proofs for NP

ZK Simulator: guess Verifier’s challenge in advance, and **rewind** if the guess was wrong.

1) Guess \((x, y)\)
2) Pick two random bits
3) Commit

If \((x, y) \neq (x', y')\)
Zero Knowledge Proofs for NP

If there are $t$ repetitions, over $2^t$ possible challenges to guess from!

Would take exponential time.
In fact, it turns out that this protocol really shouldn’t be ZK!

[DNRS99]: If you can do Fiat-Shamir for $\Pi$, then $\Pi$ wasn’t malicious-verifier ZK.
Interactive Zero-Knowledge Protocols

- No trusted setup allowed.
  - Security against Malicious verifier hard to guarantee.
- Many lines of research devoted to understanding the feasibility of interactive ZK.

[BKP18] suggests that you can do it in 3.
Interactive Zero-Knowledge Protocols

- No trusted setup allowed.
  - Security against **Malicious verifier** hard to guarantee.
- Many lines of research devoted to understanding the feasibility of interactive ZK.
  - **How many communication rounds?** [BKP18] suggests that you can do it in 3.
  - **How efficient can you make the prover?** [IKOS07, ...]
  - **Stronger forms of security:** quantum attacks, concurrency
Main Topics: Fiat-Shamir and SNARGs
Succinct Non-Interactive Arguments (SNARGs)

\[ x, \text{crs} \]

\[ P(w) \quad \pi \quad V \]

- Completeness: if \( x \in L \), \( V \) accepts honest \( P \) with probability \( 1 - \text{negl} \)
- Computational Soundness: if \( x \notin L \), for all efficient \( P^* \), \( V \) rejects w.p. \( 1 - \text{negl} \)
- Succinctness: proof has length \( \text{poly}(\lambda, \log(|x| + |w|)) \) and verification is fast.
Succinct Non-Interactive Arguments (SNARGs)

- Completeness: if $x \in L$, $V$ accepts honest $P$ with probability $1 - \text{negl}$
- Computational Soundness: if $x \notin L$, for all efficient $P^*$, $V$ rejects w.p. $1 - \text{negl}$
- Succinctness: proof has length $\text{poly}(\lambda, \log(|x| + |w|))$ and verification is fast.

This class so far: constructions of SNARGs using IOPs and a random oracle.
The Fiat-Shamir Transform

**Powerful, general** proposal for removing interaction.

Interactive

\[
P \xrightarrow{\alpha} V \xleftarrow{\beta, \gamma} P
\]

Non-Interactive

\[
P \xrightarrow{\alpha, \beta, \gamma} V
\]

If \( h \) is modeled as a random oracle, securely compiles any constant-round public coin protocol.
The Fiat-Shamir Transform

What does that mean?

If $h$ is modeled as a random oracle, securely compiles any constant-round public coin protocol.
The Random Oracle Model [BR93]

Assumption about the structure of an attack on a hash function $h$:

“The best you can do is treat $h$ as a black box in your attack.”

Under such an assumption, $h(\cdot)$ can be thought of as a random function.
Claim: Fiat-Shamir for constant-round protocols is secure in the ROM

Proof (3 message case):

\[ P^* \rightarrow h(\cdot) \rightarrow \alpha, \beta, \gamma \rightarrow V \]

\( \alpha \) must come from one of the oracle queries
Claim: Fiat-Shamir for constant-round protocols is secure in the ROM

Proof (3 message case):

\[ P^* \xrightarrow{\alpha, \beta, \gamma} h(\cdot) \]

\[ V \]

\[ V \]

\[ Q - 1 \] queries

\[ h(\cdot) \]

\[ P^* \xrightarrow{\alpha \text{ (ith query)}} \]

\[ \beta \]

\[ \gamma \]

\[ \alpha \text{ must come from one of the oracle queries} \]
Claim: Fiat-Shamir for constant-round protocols is secure in the ROM

Proof (3 message case):

$\alpha, \beta, \gamma$ must come from one of the oracle queries

Sample $i \leftarrow [Q]$ (number of queries)

$1/Q$ security loss
The Random Oracle Model [BR93]

Assumption about the structure of an attack on a hash function $h$:

“The best you can do is treat $h$ as a black box in your attack.”

Under such an assumption, $h(\cdot)$ can be thought of as a random function.
The Random Oracle Model [BR93]

Assumption about the structure of an attack on a hash function $h$:

“The best you can do is treat $h$ as a black box in your attack.”

In practice, $h(\cdot)$ is instantiated with (e.g.) SHA256, possibly salted.
The Random Oracle Model [BR93]

Assumption about the structure of an attack on a hash function $h$:

“The best you can do is treat $h$ as a black box in your attack.”

No matter what, $h(\cdot)$ is instantiated with a public efficient algorithm.
Obvious (theoretical) problem:

Public efficient algorithms can’t compute random functions
Next: example of an uninstantiable random oracle property [CGH98]
Random Oracles Do Not Exist

Fix a function $f: \{0,1\}^* \rightarrow \{0,1\}^\lambda$

We say that a hash function $h$ is Correlation Intractable (CI) for $f$ if it is hard to find $x$ such that $h(x) = f(x)$

$\forall$ PPT $A$,  

$$\Pr_{h \leftarrow H, x \leftarrow A(h)} [h(x) = f(x)] = \text{negl}$$
Random Oracles Do Not Exist

For any fixed $f$, a RO is CI for $f$.

Why? Each query $x$ to the RO produces a random output $y$, which is equal to $f(x)$ with probability $2^{-\lambda}$. 
Random Oracles Do Not Exist

Claim [CGH98]: \( \exists f \) such that for any (efficient) hash family \( H \), \( H \) fails to be CI for \( f \)!
Random Oracles Do Not Exist

Claim [CGH98]: \( \exists f \) such that for any (efficient) hash family \( H \), \( H \) fails to be CI for \( f \)!

\( f(x) \): interpret \( x \) as a program \( P \) and output \( P(x) \).
Random Oracles Do Not Exist

Claim [CGH98]: \( \exists f \) such that for any (efficient) hash family \( H \), \( H \) fails to be CI for \( f \! \). 

\( f(x) \): interpret \( x \) as a program \( P \) and output \( P(x) \).

Given \( h \leftarrow H \), attack sets \( x = \langle h \rangle \) to be a description of \( h \). Then,

\[
f(x) = P(x) = P(\langle h \rangle) = h(\langle h \rangle) = h(x).
\]
Random Oracles Do Not Exist

Is this a reasonable counterexample?

- Hash function/random oracle must be able to hash inputs of arbitrary length. CI with bounded inputs might exist!
Random Oracles Do Not Exist

Is this a reasonable counterexample?

- Hash function/random oracle must be able to hash inputs of arbitrary length. CI with bounded inputs might exist!
  - [Barak01,GK03] apply to fixed-input length hash functions.
Random Oracles Do Not Exist

Is this a reasonable counterexample?

- Hash function/random oracle must be able to hash inputs of arbitrary length. CI with bounded inputs might exist!
  - [Barak01,GK03] apply to fixed-input length hash functions.

Theorem [Barak ‘01, Goldwasser-Kalai ‘03]: ∃ interactive protocol Π such that Π_{FS} is ROM-secure but insecure for any efficiently computable H (e.g. SHA-3).
Random Oracles Do Not Exist

Is this a reasonable counterexample?

- Hash function/random oracle must be able to hash inputs of arbitrary length. CI with bounded inputs might exist!
  - [Barak01,GK03] apply to fixed-input length hash functions.
- Security property broken by running the hash function on its own description. Is this practically relevant?
Random Oracles Do Not Exist

Is this a reasonable counterexample?

- Hash function/random oracle must be able to hash inputs of arbitrary length. CI with bounded inputs might exist!
  - [Barak01,GK03] apply to fixed-input length hash functions.

- Security property broken by running the hash function on its own description. Is this practically relevant?
  - Recursive SNARKs do something of this flavor.
Random Oracles Do Not Exist

Is this a reasonable counterexample?

- Hash function/random oracle must be able to hash inputs of arbitrary length. CI with bounded inputs might exist!
  - [Barak01, GK03] apply to **fixed-input length** hash functions.
- Does **NOT** imply RO-based SNARKs are broken in practice.
  - But it does imply a lack of theoretical understanding.
What can we do without random oracles?
Falsifiable Assumptions

Prove security assuming that some concrete algorithmic task is infeasible:

- Computing discrete logarithms is hard.
- Solving random noisy linear equations (LWE) is hard.
- SHA256 is collision-resistant.
Falsifiable Assumptions

Many cryptographic constructions use random oracles to get better efficiency, but *can* be based on falsifiable assumptions.

- CCA-secure public key encryption.
- Identity-based encryption.
- Non-interactive zero knowledge.
Falsifiable Assumptions

Can (ZK-)SNARKs for NP be built based on falsifiable assumptions?

- (minor caveats but) No!
- No way to extract a long witness from a short proof. Need assumption (RO, “knowledge assumption”) that guarantees adversary “knows” a long string given a short commitment.
Can (ZK-)SNARGs for NP be built based on falsifiable assumptions?

- It’s complicated. (We don’t know)
- Significant barriers [Gentry-Wichs ‘11]
- The community is still trying to understand this.
Rest of today: SNARGs for limited computations from falsifiable assumptions (LWE)
Two tools/techniques

- **Correlation-intractable hash functions** [CCHLRRW19,PS19,HLR21]
  - Used to instantiate Fiat-Shamir without random oracles, for “nice enough” interactive protocols.

- **Somewhere extractable commitments** [HW15]
  - Used to make a “nice enough” interactive protocol
    - Special variant of the typical IOP-based approach.
Correlation Intractability

A hash family $H$ is CI for $f$ if $\forall$ PPT $A$,

$$\Pr_{h \leftarrow H, x \leftarrow A(h)} [h(x) = f(x)] = \text{negl}$$
Correlation Intractability

A hash family $H$ is CI for binary relation $R$ if $\forall$ PPT $A$,

$$\Pr_{h \leftarrow H, x \leftarrow A(h)} [(x, h(x)) \in R] = \text{negl}$$
Correlation Intractability

A hash family $H$ is CI for $f$ if $\forall$ PPT $A$,

$$\Pr_{h \leftarrow H, x \leftarrow A(h)} [h(x) = f(x)] = \text{negl}$$

- Weren’t these impossible to build?
  - Restrict to fixed input length (necessary)
  - Restrict to fixed running time on $f$ (unclear if necessary)
CI Construction

Here’s a simple construction [CLW18] using Fully Homomorphic Encryption (FHE)

\[ x \xrightarrow{\text{pk}} x \xrightarrow{\text{pk}, f} f(x) \xrightarrow{\text{sk}} f(x) \]
CI Construction

\[ \langle h \rangle = (pk, Enc(g)) \]

Real hash key: \( g \equiv 0 \) (or a uniform random string – nobody can tell)

\[ h(x) = \text{Eval}(x, Enc(g)) = Enc(g(x)) \]

Key point: \( g \) is hidden to everyone! We consider different \( g \) to prove security.
Suppose an attacker, given \( \langle h \rangle \), finds \( x \) such that \( h(x) = f(x) \).

Key idea: let \( g^*(x) = \text{Dec}(f(x)) + 1 \). We know that \( \text{Enc}(g) \approx \text{Enc}(g^*) \) if the encryption scheme is (circular-)secure.

\[
h(x) = \text{Eval}(x, \text{Enc}(g^*)) = \text{Enc}(g^*(x))
\]

\[
\text{Dec}(f(x)) = \text{Dec}(h(x)) = g^*(x) = \text{Dec}(f(x)) + 1. \text{Impossible!}
\]
Correlation Intractability: what we know

\[ H \text{ is CI for } R \text{ if } \forall \text{ PPT } A, \quad \Pr_{\substack{h \leftarrow H \\ x \leftarrow A(h)}}[(x, h(x)) \in R] = \text{negl} \]

- Constructions for efficiently computable functions:
  - From LWE ([CLW18, PS19, LV22])
  - From DDH (JJ21)
- Construction [HLR21] for (efficient) relations with “product structure”
How do we use CI to instantiate Fiat-Shamir?
Avoid the “Bad Challenges”

Def: Given false claim $x$ and a first message $\alpha$, a challenge $\beta$ is “bad” if there exists a prover message $\gamma$ making $V$ accept.

We want to say: if the (3 message) interactive protocol is sound, then (for all $x, \alpha$) most $\beta$ are not bad. True for statistically sound IPs.
Avoid the “Bad Challenges”

Exactly what CI is good for! Define relation $R_x = \{(\alpha, \beta): \beta \text{ is bad}\}$. Then if $h$ is CI for $R_x$ (when $x \notin L$), $\Pi_{FS}$ is sound using $h$!

Protocols with more than 3 messages: round-by-round soundness (each round has a type of “bad challenge” to avoid).
Avoid the “Bad Challenges”

Main challenges:

1) Sometimes our IP doesn’t have statistical soundness.
2) We can only build CI for relations $R$ that can be decided efficiently.
Important example: SNARGs via IOPs (PCPs)
SNARGs from PCPs [Kilian, Micali]

\[ P(x, w) \]  
\[ V(x) \]

Compute (long) proof string \( \pi \) from \( (x, w) \)

\[ \text{Com}(\pi) \]

\( r \) (describes location set \( S \))

\[ \text{Open} \pi_S \]

Verify opening, check consistency of \( \pi_S \)

\[ r \leftarrow \{0,1\}^\lambda \]

Candidate SNARG: apply Fiat-Shamir to this protocol!

Simplified (less efficient) version of modern SNARKs you’ve learned about.
SNARGs from PCPs [Kilian, Micali]

\[ P(x, w) \]

Compute (long) proof string \( \pi \) from \((x, w)\)

\[ V(x) \]

\( \text{Com}(\pi) \)

\( r \) (describes location set \( S \))

\( \text{Open} \pi_S \)

\[ r \leftarrow \{0, 1\}^\lambda \]

Verify opening, check consistency of \( \pi_S \)

Not statistically sound, so it’s not clear how to analyze FS without random oracles.
SNARGs for Batch NP

\[ P(x_1, ..., x_k, w_1, ..., w_k) \quad \pi \quad V(x_1, ..., x_k) \]

- Completeness: if \( x_i \in L \) for all \( i \), \( V \) accepts honest \( P \)
- Computational Soundness: if \( x_i \notin L \) for some \( i \), for all efficient \( P^* \), \( V \) rejects.
- Succinctness: proof has length \( \text{poly}(\lambda, |w|, \log k) \)

Surprisingly powerful (implies SNARGs for P, etc.)
Interactive Batch Arguments from PCPs [CJJ21]

\[ P(x_1, ..., x_k, w_1, ..., w_k) \]

\[ \text{Com}(\pi_1, ..., \pi_k) \]

\[ r \text{ (describes location set } S) \]

\[ \text{Open } \pi_1, S, ..., \pi_k, S \]

\[ V(x_1, ..., x_k) \]

\[ r \leftarrow \{0,1\}^\lambda \]

Verify opening, check consistency of \( \pi_S \)
Interactive Batch Arguments from PCPs [CJJ21]

Choose $\text{Com}$ to be *statistically binding* on one out of $k$ proofs ($\pi_1$)

If $x_i$ is false, protocol is now statistically sound! ($\pi_1$ is fixed)
SSB Commitments

\[ k_1 = H(h_1, h_2) \]

\[ h_1 = H(m_1, m_2) \]

\[ m_1 = H(M, Y) \]

\[ m_2 = H(V, E) \]

\[ h_2 = H(m_3, m_4) \]

\[ m_3 = H(C, T) \]

\[ m_4 = H(O, R) \]

\[ h_1 = H(m_1, m_2) \]

\[ k_1 = H(h_1, h_2) \]

\[ h_2 = H(m_3, m_4) \]

\[ m_3 = H(C, T) \]

\[ m_4 = H(O, R) \]

\[ h_1 = H(m_1, m_2) \]

\[ k_1 = H(h_1, h_2) \]

\[ h_2 = H(m_3, m_4) \]

\[ m_3 = H(C, T) \]

\[ m_4 = H(O, R) \]
SSB Commitments

\[ H = H_3 \text{ (binding on 3^{rd} location)} \]

- \( h_1 = H(m_1, m_2) \)
- \( m_1 = H(M, Y) \)
- \( m_2 = H(V, E) \)
- \( h_2 = H(m_3, m_4) \)
- \( m_3 = H(C, T) \)
- \( m_4 = H(O, R) \)

\[ k_1 = H(h_1, h_2) \]

(securely encodes \( V \))
SSB Commitments

\[ H = H_3 \text{ (binding on 3rd location)} \]

\[ h_1 = H(m_1, m_2) \]

\[ k_1 = H(h_1, h_2) \]

(securely encodes V)

\[ H_1 \approx_c H_2 \approx_c \ldots \approx_c H_n \]

\[ m_1 = H(M, Y) \]

\[ m_2 = H(V, E) \]

\[ m_3 = H(C, T) \]

\[ m_4 = H(O, R) \]

\[ k_1 = H(h_1, h_2) \]

\[ h_2 = H(m_3, m_4) \]
Interactive Batch Arguments from PCPs [CJJ21]

\[ P(x_1, \ldots, x_k, w_1, \ldots, w_k) \quad \text{Com}(\pi_1, \ldots, \pi_k) \quad V(x_1, \ldots, x_k) \]

Choose \( \text{Com} \) to be \textit{statistically binding} on one out of \( k \) proofs (\( \pi_1 \))

If \( x_i \) is false, protocol is now statistically sound! (\( \pi_1 \) is fixed)
Interactive Batch Arguments from PCPs [CJJ21]

\[ P(x_1, ..., x_k, w_1, ..., w_k) \quad \text{Com}(\pi_1, ..., \pi_k) \quad V(x_1, ..., x_k) \]

Choose \text{Com} to be \textit{statistically binding} on one out of \( k \) proofs (\( \pi_k \))

If \( x_i \) is false, protocol is now statistically sound! (\( \pi_k \) is fixed)
Batch Arguments from PCPs [CJJ21]

\[ P(x_1, ..., x_k, w_1, ..., w_k) \]

\[ V(x_1, ..., x_k) \]

\[ \text{Com}(\pi_1, ..., \pi_k) \]

\[ r \text{ (describes location set } S) \]

\[ \text{Open} \pi_1,S, ..., \pi_k,S \]

\[ r \leftarrow \{0,1\}^\lambda \]

Verify opening, check consistency of \( \pi_S \)

With some work, can use CI hash functions to compile this protocol.

Succinctness: \(|w| \cdot \lambda + k \cdot \lambda\), but can be reduced to \(|w| \cdot \lambda\) by recursing.
Summary of Fiat-Shamir without RO

- Use hash functions that are CI for appropriate functions/relations
  - \[\text{[CCHLRRW19, PS19, BKM20, JJ21, HLR21]}\]
- Carefully show that FS-soundness for protocols of interest follows from compatible forms of CI
  - \[\text{[CCHLRRW19]: (non-succinct) NIZK}\]
  - \[\text{[JKKZ21]: non-interactive sumcheck protocol}\]
  - \[\text{[CJJ21]: batch NP arguments}\]
Summary of Fiat-Shamir without RO

Open problems:

- Characterize which protocols can be FS-compiled (we know it doesn’t work in general [Bar01, GK03])

- SNARGs for NP from falsifiable assumptions?
END OF LECTURE