Last Lecture

- NIZK wrap-up
- Course wrap-up
- Evaluations

BDMP construction for 3-SAT: input \( \Phi \) (3cnf)

CRS: \( 6 \leftarrow \{0,3\}^{\text{long enough}}, 6 = 6_1 \cdot 6_2 \)

Prover: pick \( m \) RSA \( n, y \in QR_n \), use \( 6 \) to compute \( \Pi_x \) that \( (n, y) \) is correct

\( \Phi \): \( n \) vars, \( M \) clauses: encode sat. assignment \( \omega \): \( \omega_i \leftarrow QR_n \) if \( \omega_i = 0 \)

\( \omega_i \leftarrow QR_n \) if \( \omega_i = 1 \)

\((X_1 \lor \overline{X}_2 \lor X_5)\)

encoding of literals: \(( \omega_1, \omega_2, \omega_3, \omega_4)\)

\( X_1 \sim \omega_1 \)
\( \overline{X}_2 \sim \omega_2 \cdot y \)
\( X_3 \sim \omega_3 \)

For every clause must prove that not every literal is encoded using a QR.

Parse \( 6_2 = 6_1 0 6_1 0 \ldots 0 6_1 \)

Use \( 6_i \) for \( i^{th} \) clause as follows
- interpret it as a list of elements from \( Z_n^2 \)
- delete all \( \omega_i \)'s not in \( QR_n \) \cup \( QR_n \)
- organize in triples: \((t_1, t_2, t_3), (t_4, t_5, t_6), \ldots\)
- for triples of type \((0,0,0)\) reveal square roots
- for triples of the same type on clause \( i \), multiply with
- clause \( i \) and reveal square roots.

Output: \((n, y), \Pi_x, \omega_1, \ldots, \omega_\mu\) square roots for triples that are discarded/matched

\( \Rightarrow \) We need to make it \( \Sigma K \) for more than one statements

Multi-theorem NIZK (FLS): for a randomly chosen \( 6, \forall x, w_1, w_2 \)
- Witness-indistinguishability \((6, \text{Prover} (6, x, w_1)) \approx (6, \text{Prover} (6, x, w_2))\)

NIZK is also NIWI:
\((6, \text{Prover} (6, x, w_3)) \approx \text{Simulator} \approx (6, \text{Prover} (6, x, w_3))\)
Trick

\[ G = G_{R \circ \sigma} \land \exists \mathbf{k} \text{ security parameter} \]

\[ 2^{\mathbf{k}} \text{ log enough} \]

- that \((x, G_{R}) \in L'\) using witness \(w\)

Prover:

Use [BDMP] to prove the following statement:

\[ \exists \mathbf{g} \]

\[ x \in L \text{ (using witness } w), \text{ or } \exists \mathbf{k} \text{ a } k\text{-bit } s \text{ s.t. } G_{R} = G(s) \]

\[ L' = \{ (x, G_{R}) : \exists w \text{ s.t. either } w \text{ is witness that } x \in L \text{ or } G(w) = G_{R} \} \]

Verifier:

Just verifies the proof.

Simulator:

Use [BDMP] to prove \((x, G_{R}) \in L'\) using witness \(s\), s.t. \(G_{R} = G(s)\)

Security

- Real Prover
- Hybrid 0:
  \[ G = G_{L \circ R} \land w \text{ using } w \]

- Hybrid 1:
  \[ G = G_{L \circ R} \land w \text{ using } w \]

- Simulator
- Hybrid 2:
  \[ G = G_{L \circ R} \land w \text{ uses } S \]

Reference:

Simulatable \(\text{VRFs} - \text{crypto 018}\)

Course Wrap-Up

- Crypto toolkit: OWFs, OWFs, TDB, CRHFs
- Examples under specific number-theoretic assumptions
- Complexity theory - comp. number theory
- Algorithms

- Application:
  - Encryption, Authentication, Pseudorandomness
  - Secure communication
  - Secure computation: 2 PC, ZK proofs, NIZK

- Advanced cryptography / crypto reading group
- Policy - dialogue
- Applied cryptography
- Implementation

Methodology:

- Rigorous def.
- State assumptions
- Give proofs