PRACTICAL ZERO-KNOWLEDGE PROOFS FOR CIRCUIT EVALUATION

E. Ghadafi    N.P. Smart    B. Warinschi

Department of Computer Science,
University of Bristol

Twelfth IMA International Conference on Cryptography and Coding 15th – 17th December 2009
OUTLINE

1 ROM PROOFS

2 Groth-Sahai Proofs

3 Implementation

4 Batch Verification

5 Results

6 Summary
OUTLINE

1 ROM PROOFS

2 Groth-Sahai Proofs

3 IMPLEMENTATION

4 Batch Verification

5 RESULTS

6 SUMMARY
OUTLINE

1. ROM PROOFS
2. Groth-Sahai Proofs
3. Implementation
4. Batch Verification
5. Results
6. Summary
OUTLINE

1 ROM PROOFS
2 Groth-Sahai Proofs
3 Implementation
4 Batch Verification
5 Results
6 Summary
"A proof is whatever convinces me.", Shimon Even.

I know \( w \) st.
\[(x, w) \in R_L \]

**WI**: If Verifier cannot tell which witness was used

**ZK**: If Verifier learns nothing at all about the witness

\[ PROOF \ x \in L \]

OK \( \smile \), \( x \in L \)
Example applications:

- **Anonymous Credentials:** Client proves he possesses the required credentials without revealing them.

- **Online Voting:** Voter proves to the server that he has voted correctly without revealing his actual vote.

- **Signature Schemes, Oblivious Transfer, CCA-2 Encryption Schemes,** ...
History of NIZK Proofs

- De Santis-Di Crescenzo-Persiano, 2002.
**Our Contribution**

- Efficient implementations of NIZK proofs for Circuit SAT in the **ROM model** using Sigma-Protocols and other optimizations (e.g. Computing shared monomials, etc.).

- Efficient implementations of NIZK proofs for Circuit SAT in the **CRS model** using Groth-Sahai proofs.
Why Circuits ???

- Every $NP$ problem could be reduced to Circuit SAT.
  
  **Problem:** Circuit Size ???
  
  **Solution:** Efficient implementations would help solve some of this problem.

- Other techniques that do not require reduction to $NP$ are applicable to limited languages (i.e. You cannot prove much with them).
The interactive proof could be made non-interactive using the Fiat-Shamir transformation. The challenge is now:

$$H(Public\ parameters\ ||\ Commitment)$$

**Prover**

Public Parameters,

$$(w, x)$$

**Verifier**

Public Parameters,

$$(x)$$
The interactive proof could be made non-interactive using the Fiat-Shamir transformation. The challenge is now:

\[ H(\text{Public parameters} \ || \ \text{Commitment}) \]
The interactive proof could be made non-interactive using the Fiat-Shamir transformation. The challenge is now:

\[ H(\text{Public parameters} \| \text{Commitment}) \]
The interactive proof could be made non-interactive using the Fiat-Shamir transformation. The challenge is now:
\[ H(\text{Public parameters} \ || \ \text{Commitment}) \]
The interactive proof could be made non-interactive using the Fiat-Shamir transformation. The challenge is now: \( H(\text{Public parameters } || \text{Commitment}) \).
The interactive proof could be made non-interactive using the Fiat-Shamir transformation. The challenge is now:

\[ H(\text{Public parameters} \ || \ \text{Commitment}) \]
Symmetric External Diffie-Hellman Assumption Proofs:

**Setup:**

\[ A_1 \times A_2 \xrightarrow{f} A_T \]
Symmetric External Diffie-Hellman Assumption Proofs:

Setup:

\[ \mathbb{A}_1 \times \mathbb{A}_2 \xrightarrow{f} \mathbb{A}_T \]

\[ \mathbb{B}_1 \times \mathbb{B}_2 \xrightarrow{F} \mathbb{B}_T \]
Groth-Sahai Proofs

Symmetric External Diffie-Hellman Assumption Proofs:

Setup:

\[ \mathbb{A}_1 \times \mathbb{A}_2 \xrightarrow{f} \mathbb{A}_T \]
\[ \iota_1 \uparrow \rho_1 \quad \iota_2 \uparrow \rho_2 \quad \iota_T \uparrow \rho_T \]
\[ \mathbb{B}_1 \times \mathbb{B}_2 \xrightarrow{F} \mathbb{B}_T \]

Properties:

\[ \forall x \in \mathbb{A}_1, \forall y \in \mathbb{A}_2 : F(\iota_1(x), \iota_2(y)) = \iota_T(f(x, y)), \]
\[ \forall \mathcal{X} \in \mathbb{B}_1, \forall \mathcal{Y} \in \mathbb{B}_2 : f(p_1(\mathcal{X}), p_2(\mathcal{Y})) = p_T(F(\mathcal{X}, \mathcal{Y})). \]

Proof:

Consists of \( \Theta \in \mathbb{B}_1 \) and \( \Pi \in \mathbb{B}_2 \)
**Groth-Sahai Proofs**

- **Product Proof:** Prove that one value is the product of other two values.
  
  **Equation:** \( x_1^{(1)} \cdot x_2^{(1)} - x_1^{(2)} = 0 \).

- **Bit Proof:** Prove that a commitment hides 0 or 1.
  
  **Equation:** \( x_1^{(1)} \cdot x_2^{(1)} - x_1^{(1)} = 0 \).

- **Equality Proof:** Prove that two different commitments hide the same value.
  
  **Equation:** \( x_2^{(1)} - x_1^{(1)} = 0 \).
**IMPLEMENTATION**

The circuit input wires \( \{w_1, \ldots, w_7\} \)

The circuit final output wires \( \{w_{13}\} \)

The set of gates \( \{g_1, \ldots, g_6\} \)

The set of monomials (i.e. products needed in the QEq Method)

The set of proof wires (i.e. wires shared between monomials)
LEq-Method

- **LEq Method (Groth et al.):**
  Each gate is represented by linear equation as follows:
  
  \[ \text{out} = a \cdot x + b \cdot y + c \cdot z + d, \text{ where } \text{out} \in \{0, 1\} \]
  
  For each 2-to-1 gate, there exists unique values for \(a, b, c\) and \(d\) that makes the above equation hold.

  **OR gate as an example:** we have \(a = -1, b = -1, c = 2\) and \(d = 0\).

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>out</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-2</td>
</tr>
</tbody>
</table>
PROVER FOR LEQ-METHOD

- Evaluate every wire in the circuit given the input.
IMPLEMENTATION OF LEq-METHOD

PROVER FOR LEq-METHOD

- Evaluate every wire in the circuit given the input.
- $\forall w_i \in \mathcal{W}$ compute $comm_i = \text{comm}(w_i, r_i)$. 
**Prover for LEq-Method**

- Evaluate every wire in the circuit given the input.
- $\forall w_i \in \mathcal{W}$ compute $comm_i = comm(w_i, r_i)$.
- $\forall i \in \mathcal{W}$, Prove $comm_i \in \{0, 1\}$. 
**Implementaiton of LEq-Method**

**Prover for LEq-Method**

- Evaluate every wire in the circuit given the input.
- \( \forall w_i \in \mathcal{W} \) compute \( \text{comm}_i = \text{comm}(w_i, r_i) \).
- \( \forall i \in \mathcal{W}, \) Prove \( \text{comm}_i \in \{0, 1\} \).
- \( \forall i \in \mathcal{G}, \) prove that the linear equation value \( \in \{0, 1\} \).
PROVER FOR LEq-METHOD

- Evaluate every wire in the circuit given the input.
- $\forall w_i \in \mathcal{W}$ compute $comm_i = comm(w_i, r_i)$.
- $\forall i \in \mathcal{W}$, Prove $comm_i \in \{0, 1\}$.
- $\forall i \in \mathcal{G}$, prove that the linear equation value $\in \{0, 1\}$.
- Output the decommitment (i.e. Wire values and the randomness used in the commitment) of the circuit’s final output wires (i.e. the set $\mathcal{O}$).
For all wires, verify that $comm_i \in \{0, 1\}$. 
**VERIFIER FOR LEQ-METHOD**

- For all wires, verify that $comm_i \in \{0, 1\}$.
- For each gate, verify that the linear equation value $\in \{0, 1\}$.
IMPLEMENTATION OF LEq-METHOD

**Verifier for LEq-Method**

- For all wires, verify that $comm_i \in \{0, 1\}$.
- For each gate, verify that the linear equation value $\in \{0, 1\}$.
- For each gate, verify that the linear equation was formed correctly.

For all wires, verify that $comm_i \in \{0, 1\}$.
For each gate, verify that the linear equation value $\in \{0, 1\}$.
For each gate, verify that the linear equation was formed correctly.
**Verifier for LEq-Method**

- For all wires, verify that $comm_i \in \{0, 1\}$.
- For each gate, verify that the linear equation value $\in \{0, 1\}$.
- For each gate, verify that the linear equation was formed correctly.
- Compare the final output commitments of the circuit with those of the prover and Accept if they are identical, or Reject otherwise.
**QEq-Method**

- **QEq Method:**
  Each gate is represented by a quadratic equation as follows:
  
  \[ z = a_0 + a_1 \cdot y + a_2 \cdot x + a_3 \cdot x \cdot y \]

  **OR gate as an example:**

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

  \[ a_0 = z_0 \]
  \[ a_1 = z_1 - a_0 \]
  \[ a_2 = z_2 - a_0 \]
  \[ a_3 = z_3 - a_0 - a_1 - a_2 \]
IMPLEMENTATION OF QEq-METHOD

PROVER FOR QEq-METHOD

- Evaluate the circuit given the input.
IMPLEMENTATION OF QEq-METHOD

PROVER FOR QEq-METHOD

- Evaluate the circuit given the input.
- Compute a commitment to each input wire $comm_i = comm(w_i, r_i)$ where $w_i \in \mathcal{I}$.
Implementation of QEq-Method

Prover for QEq-Method

- Evaluate the circuit given the input.
- Compute a commitment to each input wire $\text{comm}_i = \text{comm}(w_i, r_i)$ where $w_i \in \mathcal{I}$.
- Generate a proof that $\text{comm}_i$ will open to an element $\in \{0, 1\}$ for $i = 1, ..., |\mathcal{I}|$. 

For every element of $\mathcal{M}$ on, compute a commitment to the product $\text{comm}_i, j = \text{comm}(w_i \cdot w_j, r_{i, j})$.

For each gate $g_i$, compute a commitment $\text{comm}_k$ of the output wire $w_k$ via $\text{comm}_k = \text{comm}(w_k, r_k) = a_0 + a_2 \cdot \text{comm}_i + a_1 \cdot \text{comm}_j + a_3 \cdot \text{comm}_i \cdot j$.

For all monomials, generate a proof that the commitments $\text{comm}_i \cdot j$ are consistent with the wire commitments (i.e. do product proofs together).

Output the decommitment values of the final output wires.
IMPLEMENTATION OF QEq-METHOD

PROVER FOR QEq-METHOD

- Evaluate the circuit given the input.
- Compute a commitment to each input wire $comm_i = comm(w_i, r_i)$ where $w_i \in \mathcal{I}$.
- Generate a proof that $comm_i$ will open to an element $\in \{0, 1\}$ for $i = 1, \ldots, |\mathcal{I}|$.
- For every element of $\mathcal{Mon}$, compute a commitment to the product $comm_{i,j} = comm(w_i \cdot w_j, r_{i,j})$. 
Evaluate the circuit given the input.

Compute a commitment to each input wire $comm_i = comm(w_i, r_i)$ where $w_i \in \mathcal{I}$.

Generate a proof that $comm_i$ will open to an element $\in \{0, 1\}$ for $i = 1, \ldots, |\mathcal{I}|$.

For every element of $\mathcal{M}on$, compute a commitment to the product $comm_{i,j} = comm(w_i \ast w_j, r_{i,j})$.

For each gate $g_i$, compute a commitment $comm_k$ of the output wire $w_k$ via $comm(w_k, r_k) = comm_{a_0} + a_2 \cdot comm_i + a_1 \cdot comm_j + a_3 \cdot comm_{i \ast j}$.
PROVER FOR QEq-METHOD

- Evaluate the circuit given the input.
- Compute a commitment to each input wire $comm_i = comm(w_i, r_i)$ where $w_i \in \mathcal{I}$.
- Generate a proof that $comm_i$ will open to an element $\in \{0, 1\}$ for $i = 1, ..., |\mathcal{I}|$.
- For every element of $Mon$, compute a commitment to the product $comm_{i,j} = comm(w_i \ast w_j, r_{i,j})$.
- For each gate $g_i$, compute a commitment $comm_k$ of the output wire $w_k$ via $comm(w_k, r_k) = comm_{a_0} + a_2 \cdot comm_i + a_1 \cdot comm_j + a_3 \cdot comm_{i \ast j}$.
- For all monomials, generate a proof that the commitments $comm_{i \ast j}$ are consistent with the wire commitments (i.e. do product proofs together).


**Implementation of QE q-Method**

**Prover for QE q-Method**

- Evaluate the circuit given the input.
- Compute a commitment to each input wire $comm_i = comm(w_i, r_i)$ where $w_i \in I$.
- Generate a proof that $comm_i$ will open to an element $\in \{0, 1\}$ for $i = 1, \ldots, |I|$.
- For every element of $Mon$, compute a commitment to the product $comm_{i,j} = comm(w_i \cdot w_j, r_{i,j})$.
- For each gate $g_i$, compute a commitment $comm_k$ of the output wire $w_k$ via $comm(w_k, r_k) = comm_{a_0} + a_2 \cdot comm_i + a_1 \cdot comm_j + a_3 \cdot comm_{i \cdot j}$
- For all monomials, generate a proof that the commitments $comm_{i \cdot j}$ are consistent with the wire commitments(i.e. do product proofs together).
- Output the decommitment values of the final output wires.
\( \forall i \in \mathcal{I}, \text{ verify that } \text{comm}_i \text{ will open to an element } \in \{0, 1\}. \)
VERIFIER FOR QEq-METHOD

- $\forall i \in \mathcal{I}$, verify that $comm_i$ will open to an element $\in \{0, 1\}$.
- Compute the rest of wires’ commitments (Taking advantage of the homomorphic property of the commitment scheme).
IMPLEMENTATION OF QEq-Method

**Verifier for QEq-Method**

- $\forall i \in \mathcal{I}$, verify that $\text{comm}_i$ will open to an element $\in \{0, 1\}$.
- Compute the rest of wires’ commitments (Taking advantage of the homomorphic property of the commitment scheme).
- Verify all product proofs.
Verifier for QEq-Method

- $\forall i \in I$, verify that $comm_i$ will open to an element $\in \{0, 1\}$.
- Compute the rest of wires’ commitments (Taking advantage of the homomorphic property of the commitment scheme).
- Verify all product proofs.
- Compare the final output commitments of the circuit with those of the prover and Accept if they are identical, or Reject otherwise.
Motivation:
Verification of individual proofs takes a lot of time, so we use batch verification to save some time.
**Motivation:**
Verification of individual proofs takes a lot of time, so we use batch verification to save some time.

**Batch verification in the ROM model:**
- Small Exponent Test (Bellare et al.):
  To check that $y_1 = g^{x_1}, \ldots, y_n = g^{x_n}$
Motivation:
Verification of individual proofs takes a lot of time, so we use batch verification to save some time.

Batch verification in the ROM model:

- **Small Exponent Test (Bellare et al.):**
  To check that \( y_1 = g^{x_1}, \ldots, y_n = g^{x_n} \)
  - Choose \( \gamma_1, \ldots, \gamma_n \) at random where \( |\gamma_i| = l \).
  - Compute \( X = \sum_{i=1}^{n} (x_i \cdot \gamma_i) \) and \( Y = \prod_{i=1}^{n} y_i^{\gamma_i} \).
  - The verification is done by checking that \( g^X = Y \).

- There are different ways to efficiently compute product of powers (i.e. \( Y \)).
Batch verification in the **CRS model:**

- **Product Proof:** To verify a single Product Proof, one checks:

\[
F\left(\vec{C}_1^{(2)}, -W_2\right) \cdot F\left(\vec{C}_1^{(1)}, \vec{C}_2^{(1)}\right) \cdot F(-U_1, \Pi) \cdot F(\Theta, -U_2) = 1
\]
Batch Verification

Batch verification in the **CRS model:**

- **Product Proof:** To verify a single Product Proof, one checks:
  \[ F \left( \vec{C}_1^{(2)}, -W_2 \right) \cdot F \left( \vec{C}_1^{(1)}, \vec{C}_2^{(1)} \right) \cdot F(-U_1, \Pi) \cdot F(\Theta, -U_2) = 1 \]

  Only need \( n + 3 \) products of *Four lots* of pairings compared to \( 4n \) products of *Four lots* of pairings.
Batch Verification in the **CRS model**: 

- **Product Proof:** To verify a single Product Proof, one checks:
  \[
  F \left( \vec{C}_1^{(2)}, -W_2 \right) \cdot F \left( \vec{C}_1^{(1)}, \vec{C}_2^{(1)} \right) \cdot F(-U_1, \Pi) \cdot F(\Theta, -U_2) = 1
  \]

  Only need \( n + 3 \) products of *Four lots* of pairings compared to \( 4n \) products of *Four lots* of pairings.

- **Bit Proof:** To verify a single Bit Proof, one checks:
  \[
  F \left( \vec{C}_1^{(1)}, -W_2 \right) \cdot F \left( \vec{C}_1^{(1)}, \vec{C}_2^{(1)} \right) \cdot F(-U_1, \Pi) \cdot F(\Theta, -U_2) = 1
  \]

  Only need \( n + 3 \) products of *Four lots* of pairings compared to \( 4n \) products of *Four lots* of pairings.
Batch verification in the **CRS model**:

- **Product Proof:** To verify a single Product Proof, one checks:
  \[
  F\left(\overrightarrow{C}_1^{(2)}, -W_2\right) \cdot F\left(\overrightarrow{C}_1^{(1)}, \overrightarrow{C}_2^{(1)}\right) \cdot F(-U_1, \Pi) \cdot F(\Theta, -U_2) = 1
  \]
  Only need \(n + 3\) products of *Four lots* of pairings compared to \(4n\) products of *Four lots* of pairings.

- **Bit Proof:** To verify a single Bit Proof, one checks:
  \[
  F\left(\overrightarrow{C}_1^{(1)}, -W_2\right) \cdot F\left(\overrightarrow{C}_1^{(1)}, \overrightarrow{C}_2^{(1)}\right) \cdot F(-U_1, \Pi) \cdot F(\Theta, -U_2) = 1
  \]
  Only need \(n + 3\) products of *Four lots* of pairings compared to \(4n\) products of *Four lots* of pairings.

- **Equality Proof:** To verify a single Equality Proof, one checks:
  \[
  F\left(\overrightarrow{C}_1^{(1)}, -W_2\right) \cdot F\left(W_1, \overrightarrow{C}_2^{(1)}\right) \cdot F(-U_1, \Pi) \cdot F(\Theta, -U_2) = 1
  \]
  Only need 4 products of *Four lots* of pairings(16 pairings) compared to \(4n\) products of *Four lots* of pairings(16n Pairings)!!!
## Proof Sizes Comparison

<table>
<thead>
<tr>
<th>Parameter</th>
<th>LEq-Method</th>
<th>QEq-Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commitments</td>
<td>$</td>
<td>\mathcal{W}</td>
</tr>
<tr>
<td>Bit Proofs</td>
<td>$</td>
<td>\mathcal{W}</td>
</tr>
<tr>
<td>Product Proofs</td>
<td>$-$</td>
<td>$</td>
</tr>
<tr>
<td>Decommitments</td>
<td>$</td>
<td>\mathcal{O}</td>
</tr>
</tbody>
</table>

1 If we are using the Random Oracle Model.
2 If we are using the Common Reference String Model.
CIRCUITS’ DETAILS

Circuit-1: 32-bit integers comparison.
Circuit-2: AES-128 (Prove that the plain text was encrypted under the secret key).

**Table:** Details of the two circuits used in the experiments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Circuit-1</th>
<th>Circuit-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gates</td>
<td>184</td>
<td>33880</td>
</tr>
<tr>
<td>Input Wires</td>
<td>64</td>
<td>128</td>
</tr>
<tr>
<td>Output Wires</td>
<td>1</td>
<td>128</td>
</tr>
<tr>
<td>Total Wires</td>
<td>248</td>
<td>34136</td>
</tr>
<tr>
<td>$</td>
<td>\mathcal{PW}</td>
<td>$</td>
</tr>
<tr>
<td>$</td>
<td>\mathcal{Mon}</td>
<td>$</td>
</tr>
</tbody>
</table>

**Curves Used**

**ROM:** secp256r1 curve from the SECG standard.
**CRS:** 256–bit Barreto-Naehrig curve.
RESULTS AND TIMINGS

All our timings are in seconds and were tested on a Linux machine with Intel Core Duo 3.00GHz processor.

**Table:** Timings for our two circuits

<table>
<thead>
<tr>
<th>Model</th>
<th>Circuit</th>
<th>Proof Method</th>
<th>Prover Time</th>
<th>Verifier Time</th>
<th>Batch Time</th>
<th>Time Saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM</td>
<td>1</td>
<td>LEq</td>
<td>4.7</td>
<td>5.3</td>
<td>1.97</td>
<td>62.8%</td>
</tr>
<tr>
<td>ROM</td>
<td>1</td>
<td>QEq</td>
<td>1.95/2.25</td>
<td>2.5</td>
<td>2.01/1.28</td>
<td>19.6%/48.8%</td>
</tr>
<tr>
<td>ROM</td>
<td>2</td>
<td>LEq</td>
<td>729</td>
<td>839</td>
<td>321</td>
<td>61.7%</td>
</tr>
<tr>
<td>ROM</td>
<td>2</td>
<td>QEq</td>
<td>296/280</td>
<td>372</td>
<td>360/253</td>
<td>3.2%/31.9%</td>
</tr>
<tr>
<td>CRS</td>
<td>1</td>
<td>LEq</td>
<td>44</td>
<td>450</td>
<td>64</td>
<td>85.8%</td>
</tr>
<tr>
<td>CRS</td>
<td>1</td>
<td>QEq</td>
<td>15.23</td>
<td>163</td>
<td>29.5</td>
<td>81.9%</td>
</tr>
<tr>
<td>CRS</td>
<td>2</td>
<td>LEq</td>
<td>7174</td>
<td>70300</td>
<td>9431</td>
<td>86.6%</td>
</tr>
<tr>
<td>CRS</td>
<td>2</td>
<td>QEq</td>
<td>2406</td>
<td>24861</td>
<td>4200</td>
<td>83.1%</td>
</tr>
</tbody>
</table>
Summary

- **QEq method is faster than the LEq method.**
- **Computing the shared monomials saves time.**
- **GS proofs are slower than the ROM proofs.** This is no surprise as proofs in the standard model are usually less efficient than the ROM ones.
- **GS proof verification is faster when using the "pairing product" trick.**
- **Batch verification is very beneficial in Groth-Sahai proofs.**
The End. Questions?