Quadratic Arithmetic Programs
Compiling C to Circuits

- Compiler understands a subset of C
  - Global, function, block-scoped variables
  - Arithmetic and bitwise operators
  - Functions, conditionals, bounded loops
  - Static initializers
  - Arrays, structs, pointers
  - Preprocessor syntax

- Outputs an arithmetic circuit with wire values $C_i \in \mathbb{F}_p$

\[
\begin{array}{|c|c|}
\hline
\text{Boolean} & \text{Arithmetic} \\
\hline
\text{And} & A \times B \\
\text{Or} & (A+B) - A \times B \\
\text{Not} & 1 - A \\
\text{Xor} & (A+B) - 2 \times A \times B \\
\hline
\end{array}
\]
Quadratic Programs

[GGPR – EuroCrypt 2013]

• An efficient encoding of computation
  – Lends itself well to cryptographic protocols

• Thm: Let $C$ be an arithmetic circuit that computes $F$. There is a Quadratic Arithmetic Program (QAP) of size $O(|C|)$ that computes $F$
  ⇒ Can verify any poly-time (or even NP) function

• Related theorem for Boolean circuits and Quadratic Span Programs (QSPs)
Quadratic Arithmetic Program Intuition

Construct polynomials $D(z)$ and $P(z)$ that encode gate equations and wire values $\{C_i\}$

$(c_1, ..., c_m)$ is a valid set of wire values iff:

\[
D(z) \text{ divides } P(z)
\]

\[
\equiv
\exists H(z): H(z) \cdot D(z) == P(z)
\]

\[
\equiv
\forall r_i : D(r_i) == 0 \Rightarrow P(r_i) == 0
\]

Crypto protocol checks divisibility at a random point, and hence cheaply checks correctness
Converting Arithmetic Circuit to QAPs

- Pick arbitrary root for each $X$: $r_5$, $r_6$ from $F$
- Define: $D(z) = (z - r_5)(z - r_6)$
- Define $P(z)$ via three sets of polynomials:
  \{v_1(z), \ldots, v_m(z)\}  \{w_1(z), \ldots, w_m(z)\}  \{y_1(z), \ldots, y_m(z)\}

<table>
<thead>
<tr>
<th>$z = r_5$</th>
<th>$z = r_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1(z)$</td>
<td>0</td>
</tr>
<tr>
<td>$v_2(z)$</td>
<td>0</td>
</tr>
<tr>
<td>$v_3(z)$</td>
<td>1</td>
</tr>
<tr>
<td>$v_4(z)$</td>
<td>0</td>
</tr>
<tr>
<td>$v_5(z)$</td>
<td>0</td>
</tr>
<tr>
<td>$v_6(z)$</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$r_5$</th>
<th>$r_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1(z)$</td>
<td>0</td>
</tr>
<tr>
<td>$w_2(z)$</td>
<td>0</td>
</tr>
<tr>
<td>$w_3(z)$</td>
<td>0</td>
</tr>
<tr>
<td>$w_4(z)$</td>
<td>1</td>
</tr>
<tr>
<td>$w_5(z)$</td>
<td>0</td>
</tr>
<tr>
<td>$w_6(z)$</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$r_5$</th>
<th>$r_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1(z)$</td>
<td>0</td>
</tr>
<tr>
<td>$y_2(z)$</td>
<td>0</td>
</tr>
<tr>
<td>$y_3(z)$</td>
<td>0</td>
</tr>
<tr>
<td>$y_4(z)$</td>
<td>0</td>
</tr>
<tr>
<td>$y_5(z)$</td>
<td>1</td>
</tr>
<tr>
<td>$y_6(z)$</td>
<td>0</td>
</tr>
</tbody>
</table>
Why It Works

- Define:
  \[ P(z) = (\sum c_i v_i(z)) (\sum c_i w_i(z)) - (\sum c_i y_i(z)) \]

- \( D(z) \) divides \( P(z) \) means:
  \[ \forall r_i : D(r_i) == 0 \Rightarrow P(r_i) == 0 \]

- \( D(r_5) = 0 \) \( P(r_5) = (c_3)(c_4) - (c_5) \)
- \( D(r_6) = 0 \) \( P(r_6) = (c_1+c_2)(c_5) - (c_6) \)
void main(){
    ...
    x = b[i] + d[j];
    y *= x;
    ...
    return y;
}

Pinocchio’s Verification Pipeline

Compile

Quadratic Program

\[ D(z) \]
\[ v_1(z) ... v_m(z) \]
\[ w_1(z) ... w_m(z) \]
\[ y_1(z) ... y_m(z) \]

Compile

\[ \Pi_y \]

Prove(\( EK_F, x, y \))

\[ EK_F, VK_F \]

GenKeys(F)

\{Yes, No\}

Verify(\( VK_F, x, y, \Pi_y \))
Cryptographic Protocol (simplified)

**GenKeys(F)** $\rightarrow$ **EK$_F$, VK$_F$**
Generate the QAP for F
Pick random $s$
Compute $EK_F = \{g^{v_1(s)}, ..., g^{v_m(s)}, g^{w_1(s)}, ..., g^{w_m(s)}, g^{y_1(s)}, ..., g^{y_m(s)}, g^{s^\lambda(s)}\}$
Compute $VK_F = \{g^{D(s)}\}$

**Prove(EK$_F$, x, y)** $\rightarrow$ **Π$_y$**
Evaluate circuit. Get wire values $c_1, ..., c_m$
Compute: $g^{v(s)} = \prod (g^{v_i(s)})^{c_i}$
$g^{w(s)} = \prod (g^{w_i(s)})^{c_i}$
$g^{y(s)} = \prod (g^{y_i(s)})^{c_i}$
Find $H(z)$ s.t. $H(z) \cdot D(z) = V(z) \cdot W(z) - Y(z)$
Compute $g^{H(s)} = \prod (g^{s^\lambda(s)})^{h_i}$
Proof is $(g^{v(s)}, g^{w(s)}, g^{y(s)}, g^{H(s)})$

**Verify(VK$_F$, x, y, Π$_y$)** $\rightarrow$ **{Yes, No}**
Check: $e(g^{v(s)}, g^{w(s)})/e(g^{y(s)}, g) =? e(g^{h(s)}, g^{D(s)})$
\[ e(\cdot, \cdot) \text{ is a pairing: } e(g^a, g^b) = e(g, g)^{ab} \]