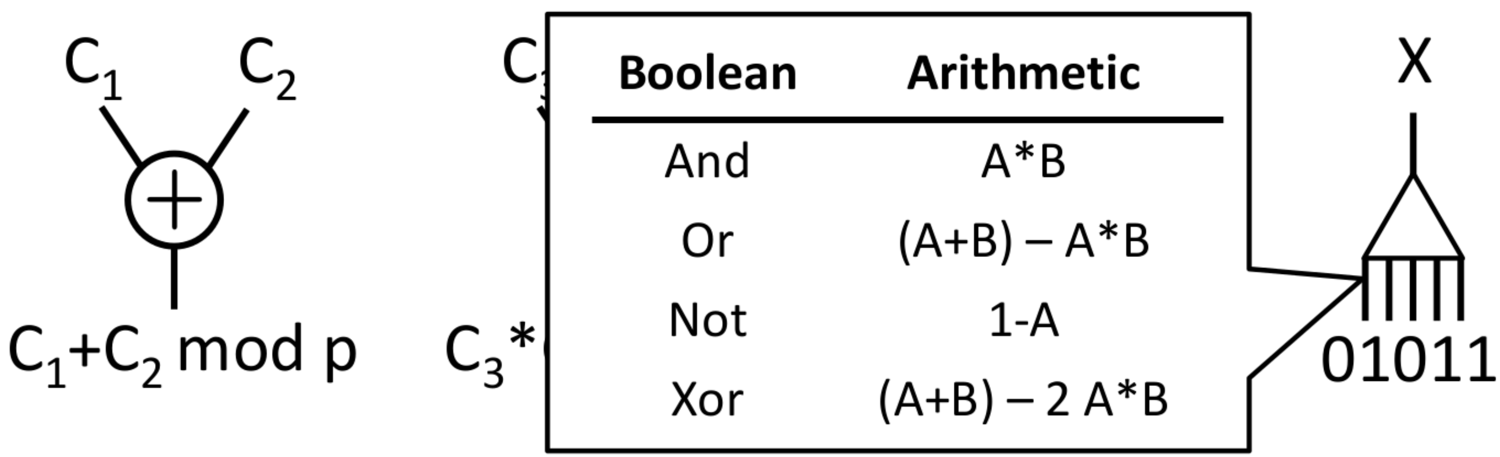
Quadratic Arithmetic Programs

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 $C_i \in \mathbb{F}_p$



Boolean	**Arithmetic**
And	A*B
Or	(A+B) – A*B
Not	1-A
Xor	(A+B) – 2 A*B

Quadratic Programs

[GGPR – EuroCrypt 2013]

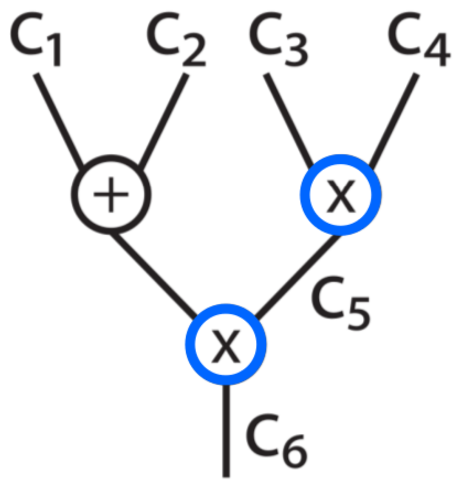
- An efficient encoding of computation
  - Lends itself well to cryptographic protocols

- Thm: Let C be an arithmetic circuit that computes F.
  There is a Quadratic Arithmetic Program (QAP)
  of size O(|C|) that computes F

  $\Rightarrow$ Can verify any poly-time (or even NP) function

- Related theorem for Boolean circuits and Quadratic Span Programs (QSPs)

Quadratic Arithmetic Program Intuition



$$C_3 * C_4 == C_5$$
$$(C_1 + C_2) * C_5 == C_6$$
$$\vdots$$

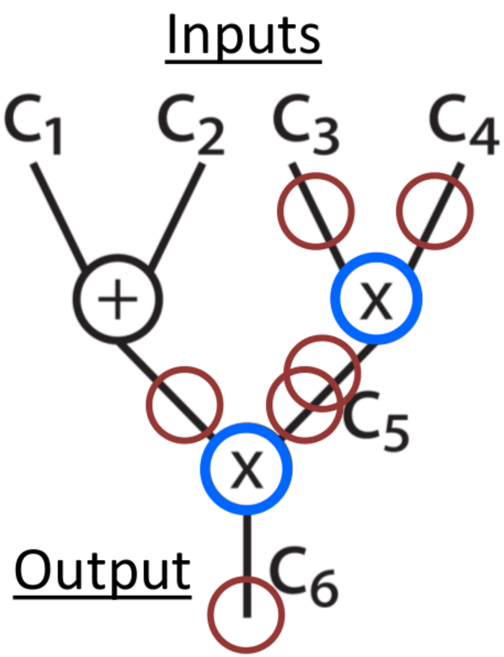
Construct polynomials $D(z)$ and $P(z)$ that encode gate equations and wire values $\{C_i\}$

$(c_1, \ldots, c_m)$ 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 $\mathbb{F}$
- Define: $D(z) = (z - r_5)(z - r_6)$
- Define $P(z)$ via three *sets* of polynomials:

  $\{v_1(z), \ldots, v_m(z)\}$ $\{w_1(z), \ldots, w_m(z)\}$ $\{y_1(z), \ldots, y_m(z)\}$

	$z=r_5$	$z=r_6$
$v_1(z)$	0	1
$v_2(z)$	0	1
$v_3(z)$	1	0
$v_4(z)$	0	0
$v_5(z)$	0	0
$v_6(z)$	0	0

Left Inputs

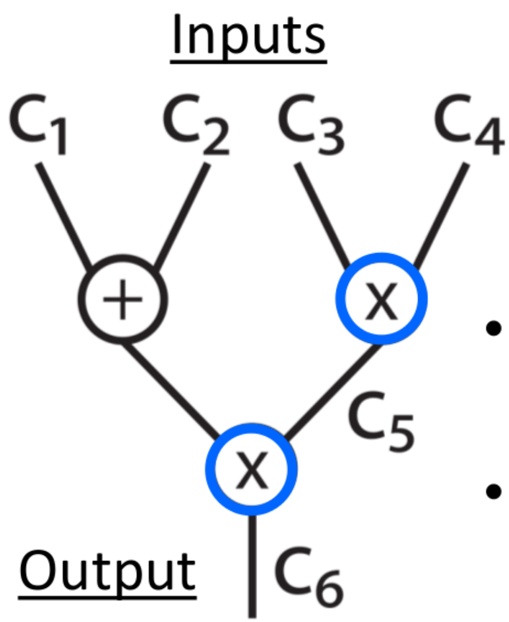
	r_5	r_6
$w_1(z)$	0	0
$w_2(z)$	0	0
$w_3(z)$	0	0
$w_4(z)$	1	0
$w_5(z)$	0	1
$w_6(z)$	0	0

Right Inputs

	r_5	r_6
$y_1(z)$	0	0
$y_2(z)$	0	0
$y_3(z)$	0	0
$y_4(z)$	0	0
$y_5(z)$	1	0
$y_6(z)$	0	1

Outputs

Why It Works



	x=r_5	x=r_6
$v_1(z)$	0	1
$v_2(z)$	0	1
$v_3(z)$	1	0
$v_4(z)$	0	0
$v_5(z)$	0	0
$v_6(z)$	0	0

	r_5	r_6
$w_1(z)$	0	0
$w_2(z)$	0	0
$w_3(z)$	0	0
$w_4(z)$	1	0
$w_5(z)$	0	1
$w_6(z)$	0	0

	r_5	r_6
$y_1(z)$	0	0
$y_2(z)$	0	0
$y_3(z)$	0	0
$y_4(z)$	0	0
$y_5(z)$	1	0
$y_6(z)$	0	1

- Define:

$$P(z) = \left(\sum c_i v_i(z)\right)\left(\sum c_i w_i(z)\right) - \left(\sum c_i y_i(z)\right)$$

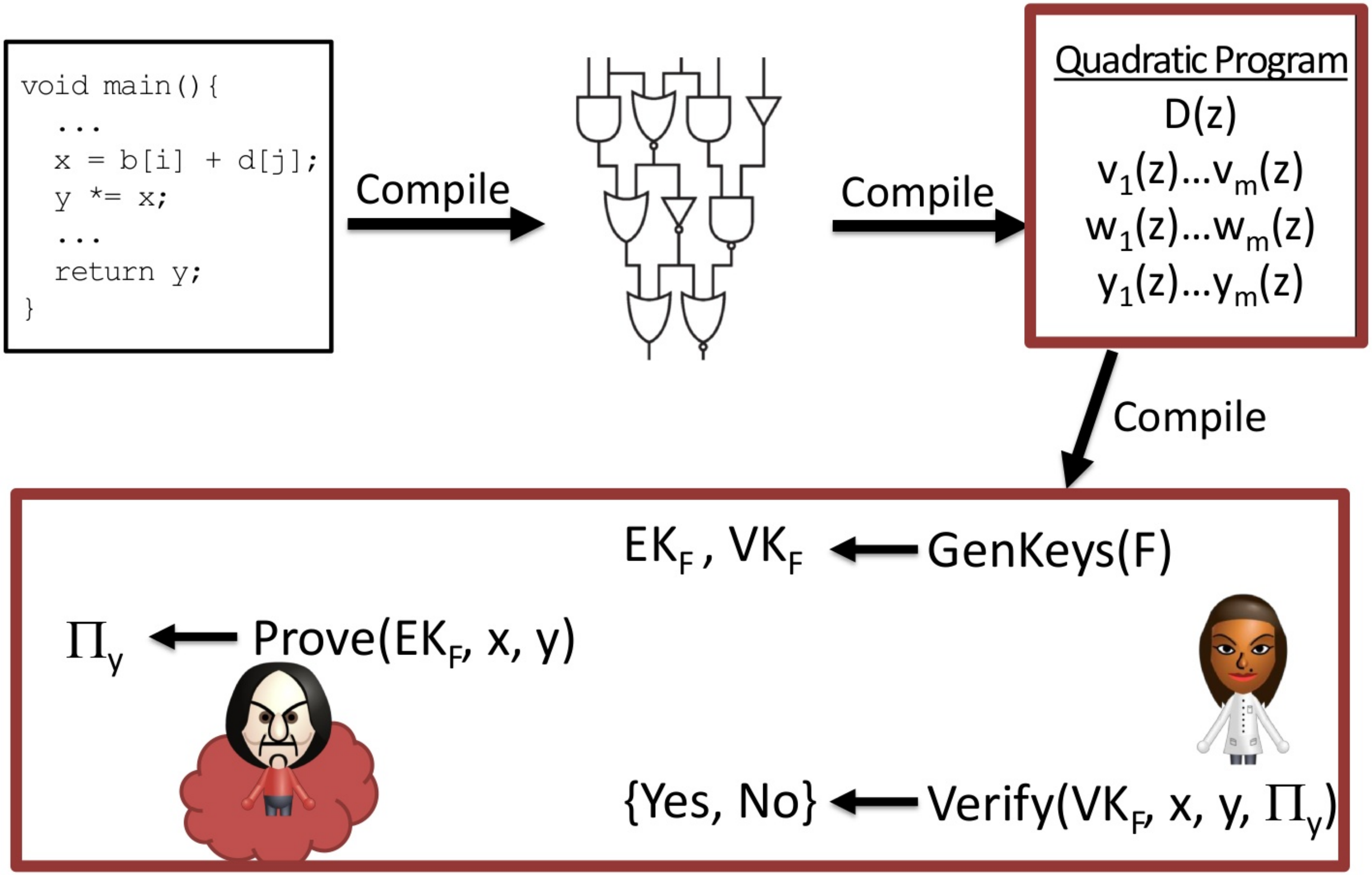
- $D(z)$ divides $P(z)$ means:

$$\forall\, r_i : D(r_i) == 0 \;\Rightarrow\; P(r_i) == 0$$

$$D(r_5) = 0 \quad P(r_5) = (c_3)(c_4) - (c_5)$$

$$D(r_6) = 0 \quad P(r_6) = (c_1 + c_2)(c_5) - (c_6)$$

Pinocchio's Verification Pipeline



Cryptographic Protocol (simplified)

GenKeys(F) ⟶ EK_F , VK_F

Generate the QAP for F

Pick random s

Compute $EK_F = \{g^{v1(s)}, \ldots, g^{vm(s)},$
$g^{w1(s)}, \ldots, g^{wm(s)},$
$g^{y1(s)}, \ldots, g^{ym(s)}, g^{s\hat{}i}\}$

Compute $VK_F = \{g^{D(s)}\}$

Prove(EK_F, x, y) ⟶ Π_y

Evaluate circuit. Get wire values $c_1, \ldots, c_m$

Compute: $g^{v(s)} = \Pi\ (g^{v\_i(s)})^{c\_i}$,
$g^{w(s)} = \Pi\ (g^{w\_i(s)})^{c\_i}$
$g^{y(s)} = \Pi\ (g^{y\_i(s)})^{c\_i}$

Find H(z) s.t. H(z)*D(z) = V(z)*W(z)-Y(z)

Compute $g^{H(s)} = \Pi\ (g^{s\hat{}i})^{h\_i}$

Proof is $(g^{v(s)}, g^{w(s)}, g^{y(s)}, g^{H(s)})$

Verify(VK_F, x, y, Π_y) ⟶ {Yes, No}

Check: $e(g^{v(s)}, g^{w(s)})/e(g^{y(s)}, g) =?= e(g^{h(s)}, g^{D(s)})$

$e(\cdot, \cdot)$ is a pairing:
$e(g^a, g^b) == e(g, g)^{ab}$