Quantum Resource Estimates for Computing Elliptic Curve Discrete Logarithms

Martin Roetteler, Michael Naehrig, Krysta Svore, Kristin Lauter
Microsoft Research

ASIACRYPT 2017
Hong Kong, 5 December 2017

Elliptic curves in cryptography

Widely used for key exchange and digital signatures

- TLS, SSH, Bitcoin, Tor, WhatsApp, Signal, …
- P-256, P-384, P-521, secp256k1, Curve25519, Curve448, …
- Security relies on hardness of ECDLP

Shor’s algorithm can solve the ECDLP in polynomial time!

What are the required resources for Shor?

Motivation

- Implement, simulate and test Shor's ECDLP algorithm on a classical machine
- Count all qubits and gates, compute depth
- Get precise resource estimates from implementation
- Compare with previous work [Proos-Zalka-04]

Elliptic curve discrete logarithm problem

Finite abelian group $(E(\mathbb{F}_p), +, \mathcal{O}),\ \#E(\mathbb{F}_p) = h \cdot r$

$P \in E(\mathbb{F}_p)$

$$[m]P = \underbrace{P + P + \cdots + P}_{m \text{ terms}}$$

ECDLP

$$\boxed{\text{Given } P, Q \text{ of order } r, \text{ such that } Q = [m]P, \text{ find } m.}$$

Shor's algorithm for the ECDLP [Shor-94]

- Let $n = \lceil \log(p) \rceil$. Prepare superposition

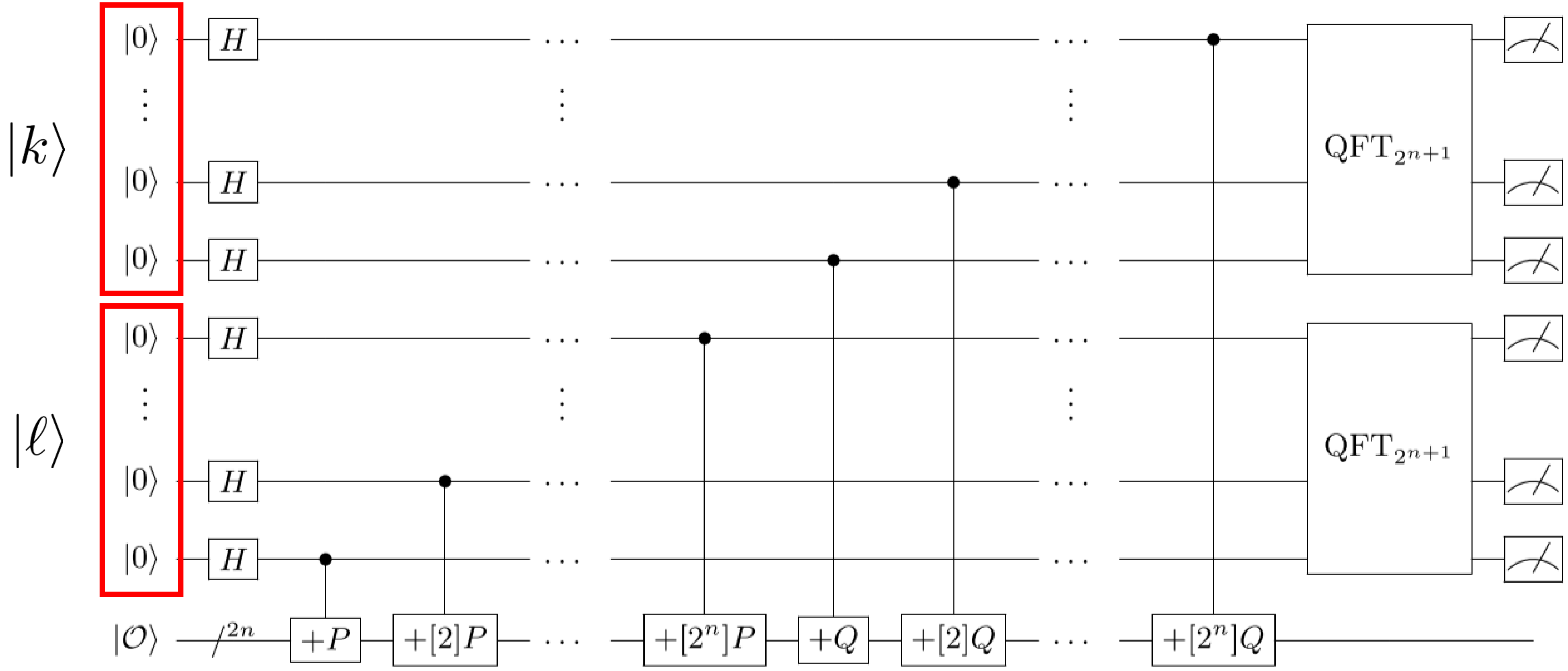
$$\frac{1}{2^{n+1}} \sum_{k,\ell=0}^{2^{n+1}-1} |k,\ell\rangle \, |\mathcal{O}\rangle$$

- Compute

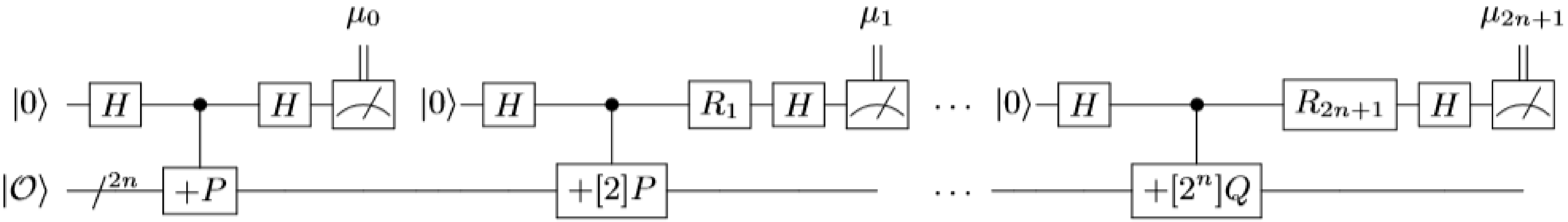
$$\boxed{\frac{1}{2^{n+1}} \sum_{k,\ell=0}^{2^{n+1}-1} |k,\ell\rangle \, |[k]P + [\ell]Q\rangle}$$

- Apply QFT to registers $|k,\ell\rangle$, measure them
- Compute DL in classical post-processing

Shor's algorithm for the ECDLP [Shor-94]

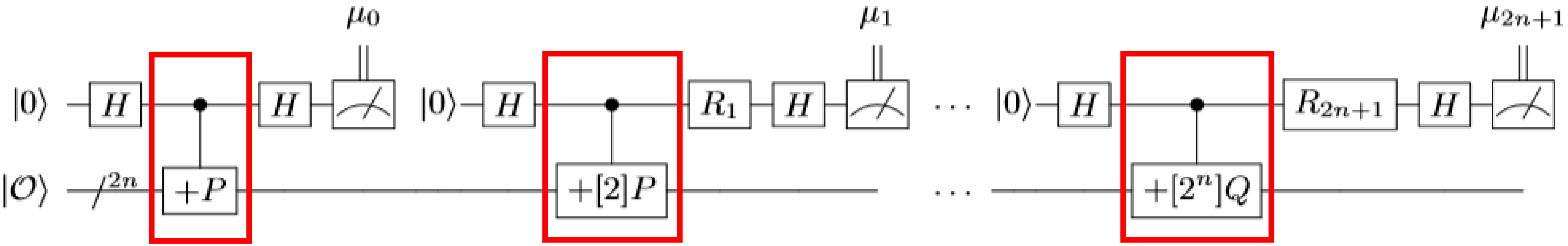


Shor's algorithm for the ECDLP



- Semiclassical Fourier transform reduces # of qubits
  [Griffiths-Niu-96], applied by [Beauregard-03, Haener-Roetteler-Svore-17] for factoring
- Measurements after every step
- Phase shift gates R_i depend on all previous measurements

Shor's algorithm for the ECDLP



- Except for Hadamard $\boldsymbol{H}$, phase shifts $\boldsymbol{R_i}$ and measurements, all gates can be implemented over the Toffoli gate set.
- Toffoli gate is universal, easily simulated classically
- Focus on elliptic curve point addition as a Toffoli network

What are the required resources for Shor?

~~Motivation~~ Plan

elliptic curve point addition

- Implement, simulate and test ~~Shor's ECDLP algorithm~~ on a classical machine

  Toffoli Toffoli

- Count all qubits and gates, compute depth
- Get precise resource estimates from implementation

  multiply by 2n (not the # of qubits though)

- Compare with previous work [Proos-Zalka-04]

The elliptic curve group law

Point addition in affine, short Weierstrass coordinates

$$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p \mid y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

$$P_1 \neq \mathcal{O} \neq P_2, P_1 \neq \pm P_2$$

$$P_1 = (x_1, y_1), P_2 = (x_2, y_2)$$

$$P_3 = (x_3, y_3) = P_1 + P_2$$

$$\boxed{\begin{aligned} \lambda &= \frac{y_2 - y_1}{x_2 - x_1} \\ x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= (x_1 - x_3)\lambda - y_1 \end{aligned}}$$

Reversible point addition

- Need to write point addition as a reversible computation

$$|R\rangle\,|00\ldots0\rangle \mapsto \left|R + [2^i]P\right\rangle |00\ldots0\rangle$$

- Clean up garbage, ancilla qubits

- Constant optimization: classical constant modulus, precomputed classical constant point multiples $[2^i]P, [2^j]Q$

- Optimize for small # of qubits first, then small # of gates and low depth

Reversible point addition

$$P_3 = P_1 + P_2$$

$$\boxed{\begin{aligned} \lambda &= \frac{y_2 - y_1}{x_2 - x_1} \\ x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= (x_1 - x_3)\lambda - y_1 \end{aligned}}$$

Reversible point addition

$$P_3 = P_1 + P_2$$

$$\boxed{\begin{aligned} \lambda &= \frac{y_1 - y_2}{x_1 - x_2} \\ x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= (x_2 - x_3)\lambda - y_2 \end{aligned}}$$

$$P_1 = P_3 + (-P_2)$$

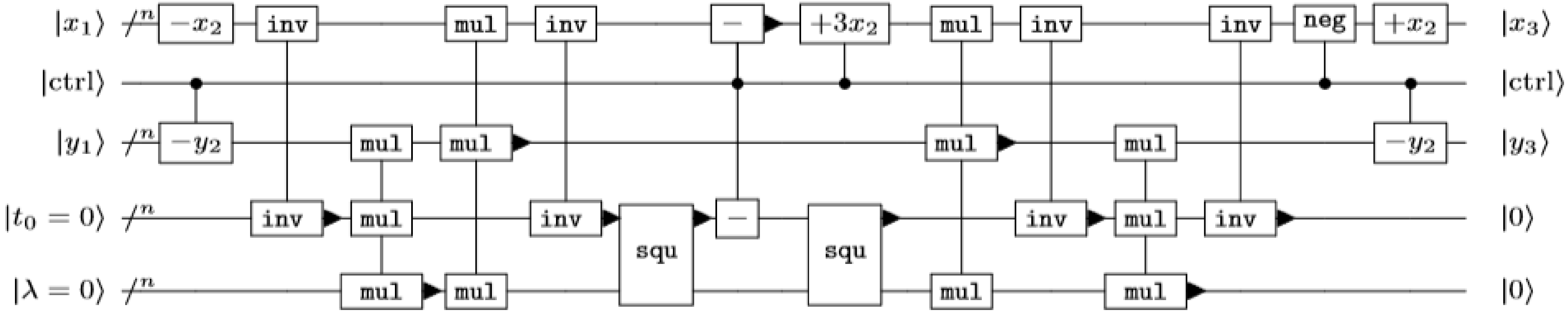
$$\boxed{\begin{aligned} \lambda' &= \frac{y_3 + y_2}{x_3 - x_2} \\ x_1 &= (\lambda')^2 - x_3 - x_2 \\ y_1 &= (x_2 - x_1)\lambda' - y_2 \end{aligned}}$$

$$\lambda = -\frac{y_3 + y_2}{x_3 - x_2} = -\lambda'$$

Controlled reversible point addition

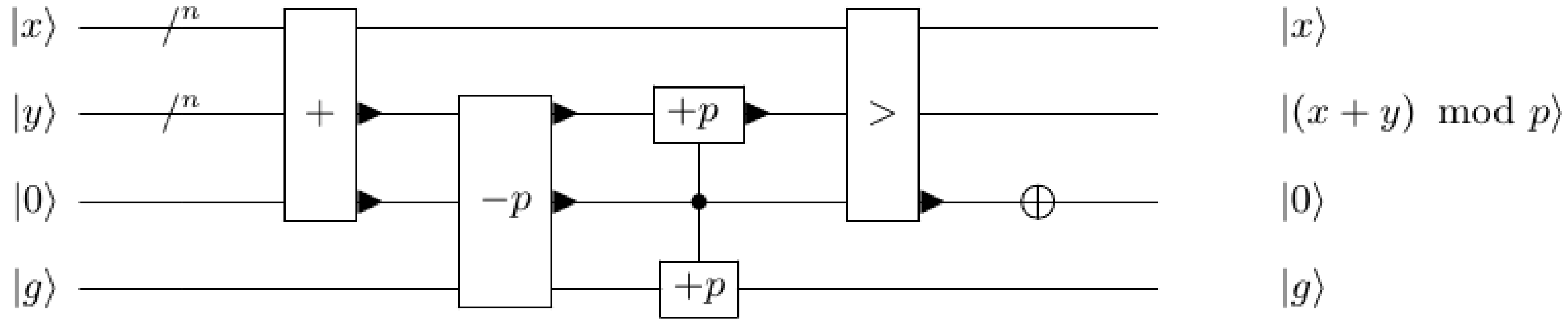
`sub_const_modp` x_1 x_2; // $x_1 \leftarrow x_1 - x_2$
`ctrl_sub_const_modp` y_1 y_2 ctrl; // $y_1 \leftarrow [y_1 - y_2]_1, [y_1]_0$
`inv_modp` x_1 t_0; // $t_0 \leftarrow 1/(x_1 - x_2)t$
`mul_modp` y_1 t_0 λ; // $\lambda \leftarrow [\frac{y_1 - y_2}{x_1 - x_2}]_1, [\frac{y_1}{x_1 - x_2}]_0$
`mul_modp` λ x_1 y_1; // $y_1 \leftarrow 0$
`inv_modp` x_1 t_0; // $t_0 \leftarrow 0$
`squ_modp` λ t_0; // $t_0 \leftarrow \lambda^2$
`ctrl_sub_modp` x_1 t_0 ctrl; // $x_1 \leftarrow [x_1 - x_2 - \lambda^2]_1, [x_1 - x_2]_0$
`ctrl_add_const_modp` x_1 $3x_2$ ctrl; // $x_1 \leftarrow [x_2 - x_3]_1, [x_1 - x_2]_0$
`squ_modp` λ t_0; // $t_0 \leftarrow 0$
`mul_modp` λ x_1 y_1; // $y_1 \leftarrow [y_3 + y_2]_1, [y_1]_0$
`inv_modp` x_1 t_0; // $t_0 \leftarrow [\frac{1}{x_2 - x_3}]_1, [\frac{1}{x_1 - x_2}]_0$
`mul_modp` t_0 y_1 λ; // $\lambda \leftarrow 0$
`inv_modp` x_1 t_0; // $t_0 \leftarrow 0$
`ctrl_neg_modp` x_1 ctrl; // $x_1 \leftarrow [x_3 - x_2]_1, [x_1 - x_2]_0$
`ctrl_sub_const_modp` y_1 y_2 ctrl; // $y_1 \leftarrow [y_3]_1, [y_1]_0$
`add_const_modp` x_1 x_2; // $x_1 \leftarrow [x_3]_1, [x_1]_0$

Controlled reversible point addition



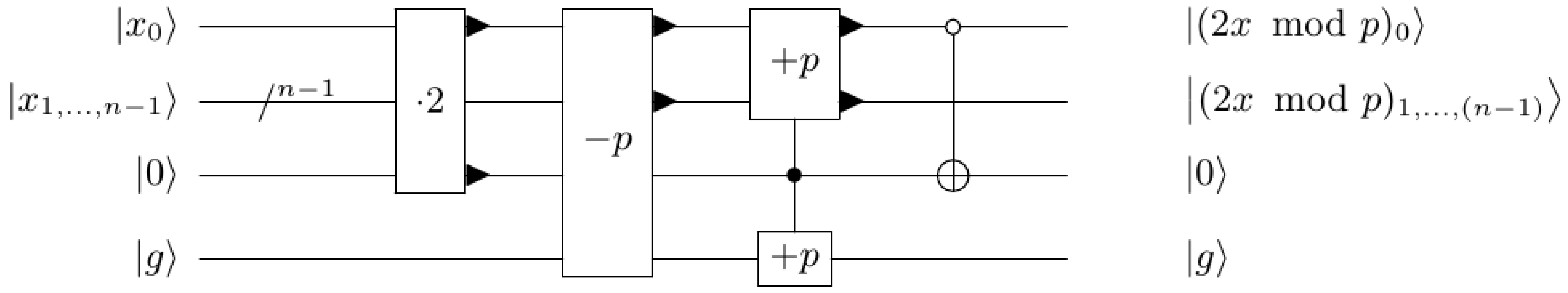
Modular arithmetic

- Integer addition/subtraction [Takahashi-Tani-Kunihiro-10]
- Constant modular addition/subtraction [Haener-Roetteler-Svore-17]
- Modular addition/subtraction



Modular arithmetic

- Integer addition/subtraction [Takahashi-Tani-Kunihiro-10]
- Constant modular addition/subtraction [Haener-Roetteler-Svore-17]
- Modular doubling

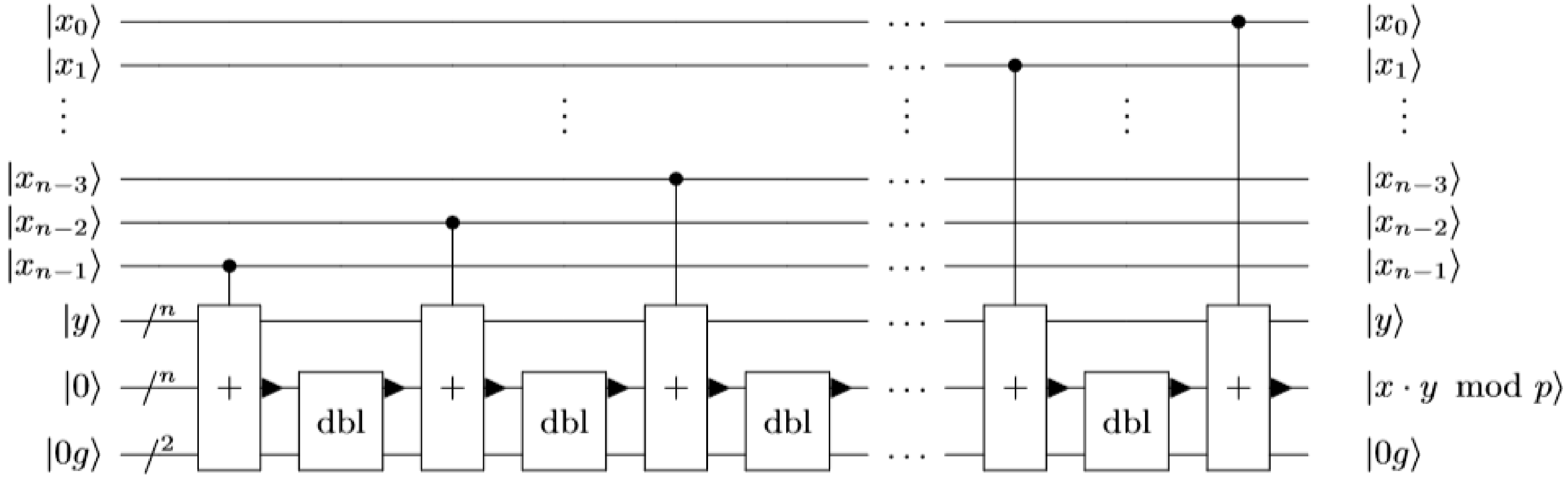


Modular multiplication

$$x = \sum_{i=0}^{n-1} x_i 2^i$$

Modular DBL/ADD approach [Proos-Zalka-04]

$$x \cdot y = x_0 y + 2(x_1 y + 2(x_2 y + \cdots + 2(x_{n-2} y + 2(x_{n-1} y)) \ldots))$$

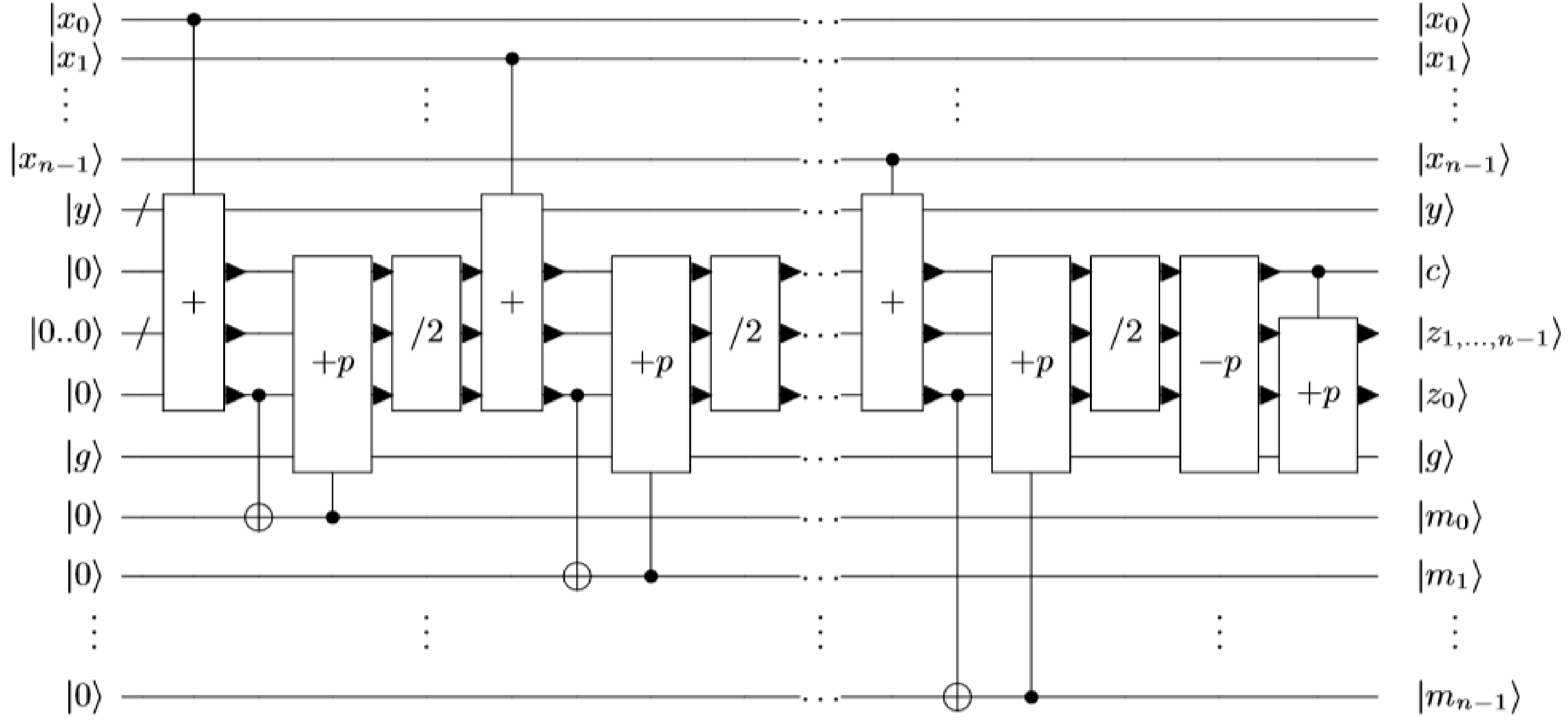


Modular multiplication

Montgomery multiplication [Montgomery-85]

$$x = \sum_{i=0}^{n-1} x_i 2^i$$

$$z = x \cdot y \cdot 2^{-n} \bmod p$$



for $i = 0, \ldots, n-1$

$$z \leftarrow z + x_i y$$

$$z \leftarrow z + z_0 p$$

$$z \leftarrow z/2$$

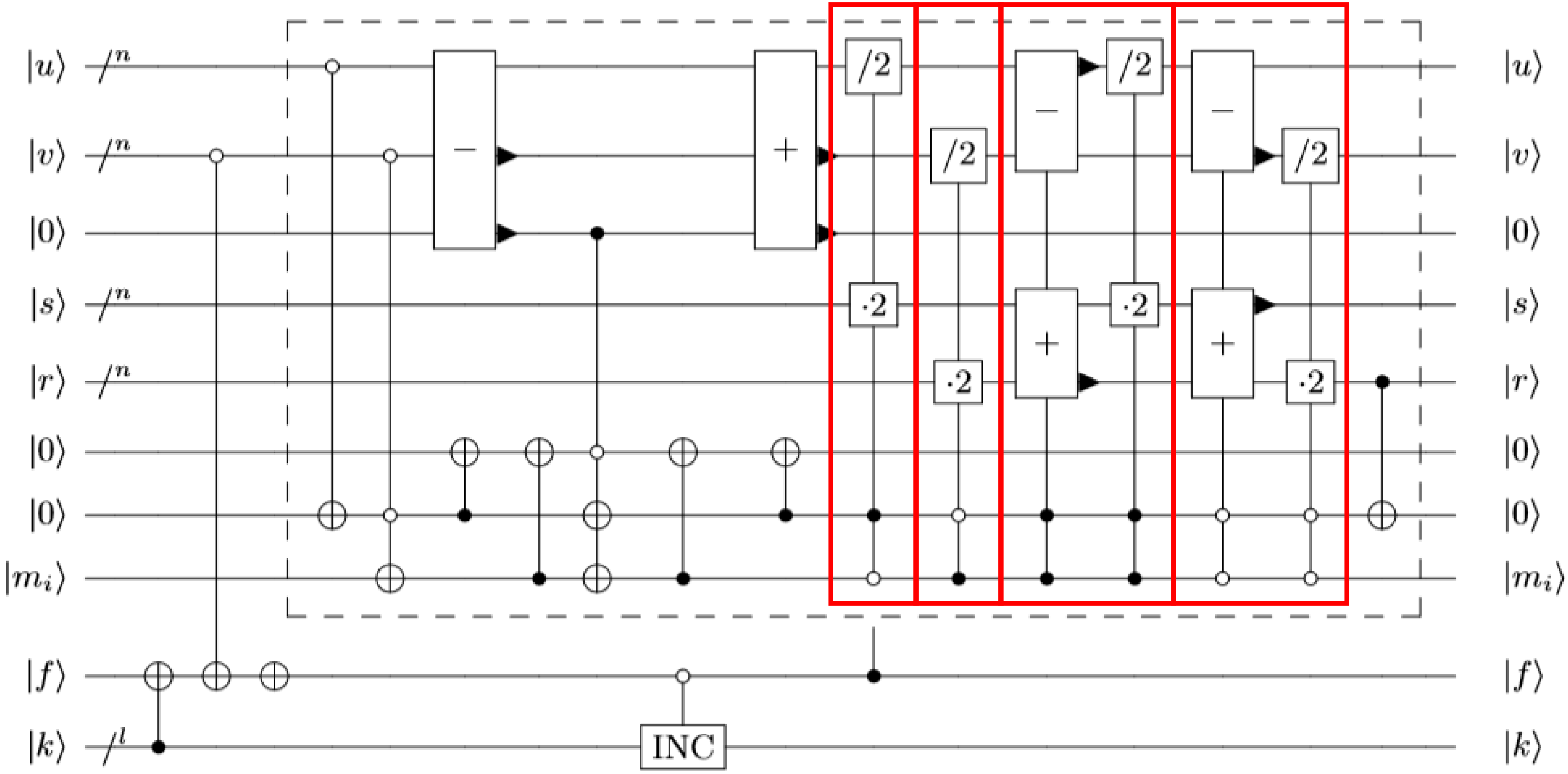
Modular inversion

Kaliski's binary GCD

- Invariant $p = rv + su$
- Computes $x^{-1}2^k \bmod p$
- Upper bound on # of while iterations: $2n$

$u \leftarrow p, v \leftarrow x, r \leftarrow 0, s \leftarrow 1$
$k \leftarrow 0$
while $v > 0$ **do**
  if u even **then**
    $u \leftarrow u/2,\ s \leftarrow 2s$
  else if v even **then**
    $v \leftarrow v/2,\ r \leftarrow 2r$
  else if $u > v$ **then**
    $u \leftarrow (u-v)/2,\ r \leftarrow r+s,\ s \leftarrow 2s$
  else
    $v \leftarrow (v-u)/2,\ s \leftarrow r+s,\ r \leftarrow 2r$
  $k \leftarrow k+1$
return $r \bmod p$

Modular inversion



Resource estimates: modular arithmetic

- Circuit implementations in LIQ*Ui*|> framework

Circuit	**# qubits**	**# Toffoli gates**
mul_modp (dbl/add)	$3n+2$	$\approx 32n^2\log_2(n)$
mul_modp (Montgomery)	$5n+4$	$\approx 16n^2\log_2(n)$
inv_modp	$7n+2\lceil\log_2(n)\rceil+9$	$\approx 32n^2\log_2(n)$

Resource estimates: Shor's algorithm

Concrete LIQ$Ui|\rangle$ simulation results

ECDLP

n	# qubits	# Toffoli gates	Toffoli depth
224	2042	$8.43 \cdot 10^{10}$	$7.73 \cdot 10^{10}$
256	2330	$1.26 \cdot 10^{11}$	$1.16 \cdot 10^{11}$
384	3484	$4.52 \cdot 10^{11}$	$4.15 \cdot 10^{11}$
521	4719	$1.14 \cdot 10^{12}$	$1.05 \cdot 10^{12}$

$9n + 2\lceil\log_2(n)\rceil + 10$ qubits

$\approx 448n^3\log_2(n)$ Toffoli gates

[Proos-Zalka-04] estimate $\approx 6n$ qubits

Factoring [Haener-Roetteler-Svore-17]

$\lceil\log_2(N)\rceil$	# qubits	# Toffoli gates
2048	4098	$5.20 \cdot 10^{12}$
3072	6146	$1.86 \cdot 10^{13}$
7680	15362	$3.30 \cdot 10^{14}$
15360	30722	$2.87 \cdot 10^{15}$

$2n' + 2$ qubits, $n' = \lceil\log_2(N)\rceil$

$\approx 64n'^3\log_2(n')$ Toffoli gates