On Repeated Squarings in Binary Fields

Kimmo Järvinen

Department of Information and Computer Science
Helsinki University of Technology

August 14, 2009

Introduction

Repeated squaring

- Repeated squaring: $a^{2^e}(x)$ where $a(x) \in \mathbb{F}_{2^m}$ with polynomial basis
- Applications in elliptic curve cryptography (e.g., inversions in the field and scalar multiplications on Koblitz curves)

Field-programmable gate arrays (FPGAs)

- Popular implementation platforms for cryptography
- Existing repeated squarers iterate squaring for e times
- How to implement efficient repeated squarers with FPGAs?

Introduction

Repeated squaring

- Repeated squaring: $a^{2^e}(x)$ where $a(x) \in \mathbb{F}_{2^m}$ with polynomial basis
- Applications in elliptic curve cryptography (e.g., inversions in the field and scalar multiplications on Koblitz curves)

Field-programmable gate arrays (FPGAs)

- Popular implementation platforms for cryptography
- Existing repeated squarers iterate squaring for e times
- How to implement efficient repeated squarers with FPGAs?

Repeated squaring in binary fields

Squaring is $a^2(x) = \sum_{i=0}^{m-1} a_i x^{2i} \mod p(x)$ where $a_i \in \{0,1\}$ and $p(x)$ is an irreducible polynomial

A linear transformation described by $\mathbf{Qa}$ where $\mathbf{a} = [a_0 a_1 \ldots a_{m-1}]^T$ and

$$\mathbf{Q} = \begin{bmatrix} 1 & q_{0,1} & q_{0,2} & \cdots & q_{0,m-1} \\ 0 & q_{1,1} & q_{1,2} & \cdots & q_{1,m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & q_{m-1,1} & q_{m-1,2} & \cdots & q_{m-1,m-1} \end{bmatrix}$$

A repeated squaring is given by $\mathbf{Q}^e\mathbf{a}$

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Repeated squaring in binary fields

Squaring is $a^2(x) = \sum_{i=0}^{m-1} a_i x^{2i} \mod p(x)$ where $a_i \in \{0, 1\}$ and $p(x)$ is an irreducible polynomial

A linear transformation described by $\mathbf{Qa}$ where $\mathbf{a} = [a_0 a_1 \ldots a_{m-1}]^T$ and

$$\mathbf{Q} = \begin{bmatrix} 1 & q_{0,1} & q_{0,2} & \cdots & q_{0,m-1} \\ 0 & q_{1,1} & q_{1,2} & \cdots & q_{1,m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & q_{m-1,1} & q_{m-1,2} & \cdots & q_{m-1,m-1} \end{bmatrix}$$

A repeated squaring is given by $\mathbf{Q}^e\mathbf{a}$

Look-up tables (LUTs)

- Basic building block of an FPGA is an n-to-1 bit look-up table (n-LUT)
- Typically, $n = 4$ but contemporary FPGAs have larger n, e.g., $n = 6$ (Xilinx Virtex-5) or $n = 7$ (Altera Stratix-II, and beyond)
- Notice that using only two inputs of an n-LUT costs as much as using all of its inputs

Definitions

Definition (Weight and row-weight)

Weight, $\mathcal{W}(\mathbf{Q}^e)$, *is the number of ones in* $\mathbf{Q}^e$*; and*
Row-weight, $\mathcal{W}_i(\mathbf{Q}^e)$, *is the number of ones on the* i^{th} *row of* $\mathbf{Q}^e$

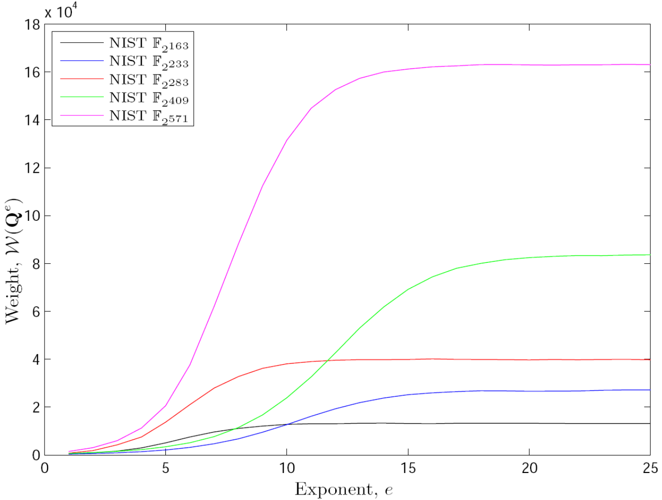
Definition (Area)

Area, $\mathcal{A}(\mathbf{Q}^e)$, *is the number of* n*-LUTs required to implement* $\mathbf{Q}^e$

Definition (Critical path)

Critical path, $\mathcal{D}(\mathbf{Q}^e)$, *is the length of the longest path of consecutive* n*-LUTs in the circuit computing* $\mathbf{Q}^e$

Weights of the NIST fields



Area and delay

Area

It is possible to implement $\mathbf{Q}^e$ with a circuit whose area $\mathcal{A}_n(\mathbf{Q}^e)$ satisfies

$$\mathcal{A}_n(\mathbf{Q}^e) \leq \sum_{i=1}^{m} \left\lceil \frac{\mathcal{W}_i(\mathbf{Q}^e) - 1}{n - 1} \right\rceil$$

Delay

Critical path, $\mathcal{D}_n(\mathbf{Q}^e)$, is bounded by

$$\mathcal{D}_n(\mathbf{Q}^e) \leq \max_i \lceil \log_n \mathcal{W}_i(\mathbf{Q}^e) \rceil$$

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Area and delay

Area

It is possible to implement $\mathbf{Q}^e$ with a circuit whose area $\mathcal{A}_n(\mathbf{Q}^e)$ satisfies

$$\mathcal{A}_n(\mathbf{Q}^e) \leq \sum_{i=1}^{m} \left\lceil \frac{\mathcal{W}_i(\mathbf{Q}^e) - 1}{n - 1} \right\rceil$$

Delay

Critical path, $\mathcal{D}_n(\mathbf{Q}^e)$, is bounded by

$$\mathcal{D}_n(\mathbf{Q}^e) \leq \max_i \lceil \log_n \mathcal{W}_i(\mathbf{Q}^e) \rceil$$

Example

Consider computing $a^{2^2}(x)$ *in* $\mathbb{F}_2[x]/x^4 + x + 1$. *We have*

$$\mathbf{Q}^2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

- *Weights:* $\mathcal{W}(\mathbf{Q}^2) = 9$ *and* $\mathcal{W}_1(\mathbf{Q}^2) = 4$, $\mathcal{W}_2(\mathbf{Q}^2) = 2$, $\mathcal{W}_3(\mathbf{Q}^2) = 2$, *and* $\mathcal{W}_4(\mathbf{Q}^2) = 1$.
- *Area: if* $n = 2$, *we get* $\mathcal{A}_2(\mathbf{Q}^2) \leq 5$ *(minimum* $\mathcal{A}_2(\mathbf{Q}^2) = 4$*). If* $n = 4$, *we get* $\mathcal{A}_4(\mathbf{Q}^2) = 3$
- *Delay: if* $n = 2$, *we get* $\mathcal{D}_2(\mathbf{Q}^2) = 2$ *and* $\mathcal{D}_4(\mathbf{Q}^2) = 1$.

Example

Consider computing $a^{2^2}(x)$ *in* $\mathbb{F}_2[x]/x^4 + x + 1$. *We have*

$$\mathbf{Q}^2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

- *Weights:* $\mathcal{W}(\mathbf{Q}^2) = 9$ *and* $\mathcal{W}_1(\mathbf{Q}^2) = 4$, $\mathcal{W}_2(\mathbf{Q}^2) = 2$, $\mathcal{W}_3(\mathbf{Q}^2) = 2$, *and* $\mathcal{W}_4(\mathbf{Q}^2) = 1$.
- *Area: if* $n = 2$, *we get* $\mathcal{A}_2(\mathbf{Q}^2) \leq 5$ *(minimum* $\mathcal{A}_2(\mathbf{Q}^2) = 4$*). If* $n = 4$, *we get* $\mathcal{A}_4(\mathbf{Q}^2) = 3$
- *Delay: if* $n = 2$, *we get* $\mathcal{D}_2(\mathbf{Q}^2) = 2$ *and* $\mathcal{D}_4(\mathbf{Q}^2) = 1$.

Example

Consider computing $a^{2^2}(x)$ *in* $\mathbb{F}_2[x]/x^4+x+1$. *We have*

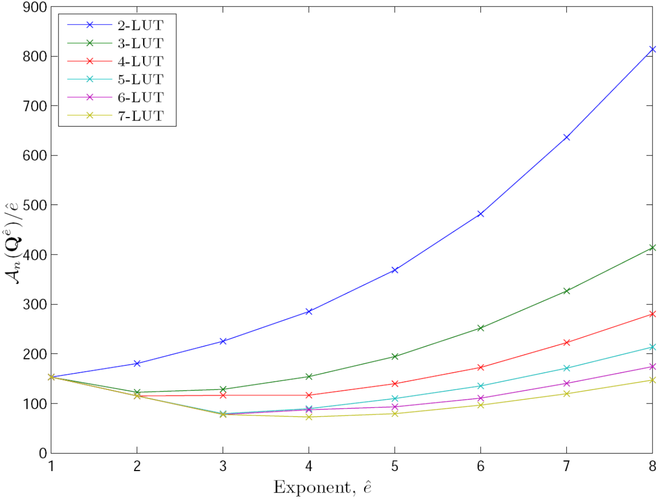
$$\mathbf{Q}^2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

- *Weights:* $\mathcal{W}(\mathbf{Q}^2) = 9$ *and* $\mathcal{W}_1(\mathbf{Q}^2) = 4$, $\mathcal{W}_2(\mathbf{Q}^2) = 2$, $\mathcal{W}_3(\mathbf{Q}^2) = 2$, *and* $\mathcal{W}_4(\mathbf{Q}^2) = 1$.
- *Area: if* $n = 2$, *we get* $\mathcal{A}_2(\mathbf{Q}^2) \leq 5$ *(minimum* $\mathcal{A}_2(\mathbf{Q}^2) = 4$*). If* $n = 4$, *we get* $\mathcal{A}_4(\mathbf{Q}^2) = 3$
- *Delay: if* $n = 2$, *we get* $\mathcal{D}_2(\mathbf{Q}^2) = 2$ *and* $\mathcal{D}_4(\mathbf{Q}^2) = 1$.

Example

Table: Areas and delays for NIST $\mathbb{F}_{2^{233}}$ with different n

	$\mathcal{A}_n(\mathbf{Q}^e)$						$\mathcal{D}_n(\mathbf{Q}^e)$					
n	2	3	4	5	6	7	2	3	4	5	6	7
$e = 1$	153	153	153	153	153	153	1	1	1	1	1	1
$e = 2$	361	245	230	230	230	230	2	2	2	1	1	1
$e = 3$	676	385	349	238	233	233	3	2	2	2	1	1
$e = 4$	1141	616	466	358	349	291	4	3	2	2	2	2
$e = 5$	1844	973	699	550	466	396	4	3	2	2	2	2
$e = 6$	2892	1511	1035	812	663	580	5	3	3	2	2	2


900
800
700
600
500
400
300
200
100
0
1
2
3
4
5
6
7
8
2-LUT
3-LUT
4-LUT
5-LUT
6-LUT
7-LUT
$\mathcal{A}_n(\mathbf{Q}^{\hat{e}})/\hat{e}$
Exponent, $\hat{e}$

Implementation: Idea

Rather than

- iterating a squarer for e clock cycles,
- computing $\mathbf{Q}^e$ directly, or
- using unrolled squarers ($\mathbf{Q}||\mathbf{Q}||\ldots||\mathbf{Q}$, e times)

we search a concatenation $\mathbf{Q}^{e_1}||\mathbf{Q}^{e_2}||\ldots||\mathbf{Q}^{e_N}$ with $e = \sum_{i=1}^{N} e_i$ minimizing the metric under optimization (area, delay, etc.)

Example

If $e = 9$ and $n = 6$, the setup minimizing area is $\mathbf{Q}^3||\mathbf{Q}^3||\mathbf{Q}^3$ which has an area estimate of 699 LUTs and a critical path of 3 LUTs. (Iterative: 153 LUTs + 233 regs / 9 cycles, Direct: 1944 / 3 LUTs, Square chain: 1377 / 9 LUTs)

Implementation: Idea

Rather than

- iterating a squarer for e clock cycles,
- computing $\mathbf{Q}^e$ directly, or
- using unrolled squarers ($\mathbf{Q}||\mathbf{Q}||\ldots||\mathbf{Q}$, e times)

we search a concatenation $\mathbf{Q}^{e_1}||\mathbf{Q}^{e_2}||\ldots||\mathbf{Q}^{e_N}$ with $e = \sum_{i=1}^{N} e_i$ minimizing the metric under optimization (area, delay, etc.)

Example

If $e = 9$ and $n = 6$, the setup minimizing area is $\mathbf{Q}^3||\mathbf{Q}^3||\mathbf{Q}^3$ which has an area estimate of 699 LUTs and a critical path of 3 LUTs. (Iterative: 153 LUTs + 233 regs / 9 cycles, Direct: 1944 / 3 LUTs, Square chain: 1377 / 9 LUTs)

Implementation: Idea

Rather than

- iterating a squarer for e clock cycles,
- computing $\mathbf{Q}^e$ directly, or
- using unrolled squarers ($\mathbf{Q}||\mathbf{Q}||\ldots||\mathbf{Q}$, e times)

we search a concatenation $\mathbf{Q}^{e_1}||\mathbf{Q}^{e_2}||\ldots||\mathbf{Q}^{e_N}$ with $e = \sum_{i=1}^{N} e_i$ minimizing the metric under optimization (area, delay, etc.)

Example

If $e = 9$ and $n = 6$, the setup minimizing area is $\mathbf{Q}^3||\mathbf{Q}^3||\mathbf{Q}^3$ which has an area estimate of 699 LUTs and a critical path of 3 LUTs. (Iterative: 153 LUTs + 233 regs / 9 cycles, Direct: 1944 / 3 LUTs, Square chain: 1377 / 9 LUTs)

Implementation: Idea

Rather than

- iterating a squarer for e clock cycles,
- computing $\mathbf{Q}^e$ directly, or
- using unrolled squarers ($\mathbf{Q}||\mathbf{Q}||\ldots||\mathbf{Q}$, e times)

we search a concatenation $\mathbf{Q}^{e_1}||\mathbf{Q}^{e_2}||\ldots||\mathbf{Q}^{e_N}$ with $e = \sum_{i=1}^{N} e_i$ minimizing the metric under optimization (area, delay, etc.)

Example

If $e = 9$ and $n = 6$, the setup minimizing area is $\mathbf{Q}^3||\mathbf{Q}^3||\mathbf{Q}^3$ which has an area estimate of 699 LUTs and a critical path of 3 LUTs. (Iterative: 153 LUTs + 233 regs / 9 cycles, Direct: 1944 / 3 LUTs, Square chain: 1377 / 9 LUTs)

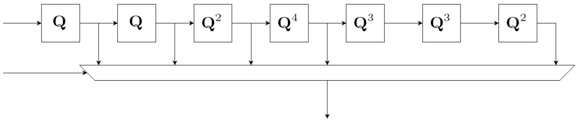
Implementation: Varying exponent

Solution 1 (Distinct exponents, $\{e_1, \ldots, e_\ell\}$)

- Let $\Delta_i = e_i - e_{i-1}$
- Find the optimal circuits for each Δ_i and concatenate them
- Select results using a multiplexer

Example

If $E = \{1, 2, 4, 8, 16\}$ and $n = 6$, we get the repeated squarer shown below with an area estimate of 1600 LUTs.



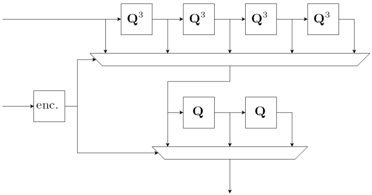
Implementation: Varying exponent

Solution 2 (Range, $0 \le e \le e_{max}$)

- Let e_{opt} be the exponent that minimizes $\mathcal{A}_n(\mathbf{Q}^{\hat{e}})/\hat{e}$
- Concatenate $\lfloor e_{max}/e_{\mathrm{opt}} \rfloor$ $\mathbf{Q}^{e_{\mathrm{opt}}}$ blocks
- Compute the remaining squarings with a square chain

Example

If $0 \le e \le 14$ and $n = 6$, we get the repeated squarer shown below with an area estimate of 1238 LUTs.



Results

- Several repeated squarers were synthesized for Spartan-3A and Virtex-5 FPGAs (see the paper)
- The results show that repeated squarers are small and fast enough to be included in existing finite field processors

Example (NIST $\mathbb{F}_{2^{233}}$, Virtex-5)

Solution 1 with $\{1, 2, 4, 8, 16\}$*: area 1823 LUTs and delay 8.23 ns*
Solution 2 with $0 \leq e \leq 11$*: area 1809 LUTs and delay 8.10 ns*

Results

- Several repeated squarers were synthesized for Spartan-3A and Virtex-5 FPGAs (see the paper)
- The results show that repeated squarers are small and fast enough to be included in existing finite field processors

Example (NIST $\mathbb{F}_{2^{233}}$, Virtex-5)

Solution 1 with $\{1, 2, 4, 8, 16\}$*: area 1823 LUTs and delay 8.23 ns*
Solution 2 with $0 \leq e \leq 11$*: area 1809 LUTs and delay 8.10 ns*

Inversions in binary fields

- Fermat's Little Theorem $\Rightarrow a^{-1}(x) = a^{2^m-2}(x)$
- Computed with a series of multiplications and (repeated) squarings
- Itoh and Tsujii: $\lfloor \log_2(m-1) \rfloor + w(m-1) - 1$ multiplications and $m-1$ squarings

Example (Inversion in $\mathbb{F}_{2^{233}}$)

Computed with 10 multiplications and 232 squarings
A repeated squarer (solution 1) with $e \in \{1, 2, 4, 8, 16\}$ gives the following speedups with different multiplier latencies:
$M = 18 \Rightarrow 52\,\%$, $M = 6 \Rightarrow 73\,\%$, and $M = 1 \Rightarrow 88\,\%$
(19 repeated squarings instead of 232 squarings)

Inversions in binary fields

- Fermat's Little Theorem $\Rightarrow a^{-1}(x) = a^{2^m-2}(x)$
- Computed with a series of multiplications and (repeated) squarings
- Itoh and Tsujii: $\lfloor \log_2(m-1) \rfloor + w(m-1) - 1$ multiplications and $m-1$ squarings

Example (Inversion in $\mathbb{F}_{2^{233}}$)

Computed with 10 multiplications and 232 squarings
A repeated squarer (solution 1) with $e \in \{1, 2, 4, 8, 16\}$ *gives the following speedups with different multiplier latencies:*
$M = 18 \Rightarrow 52\,\%$, $M = 6 \Rightarrow 73\,\%$, *and* $M = 1 \Rightarrow 88\,\%$
(19 repeated squarings instead of 232 squarings)

Scalar multiplication on Koblitz curves

- Scalar multiplication on Koblitz curves, kP where $k = \sum_{i=0}^{\ell-1} k_i \tau^i$, computed with the binary algorithm: $w(k)$ point additions and $\ell - 1$ Frobenius maps
- Frobenius map: $(x, y) \mapsto (x^2, y^2)$
- e successive Frobenius maps can be computed with two repeated squarings: (x^{2^e}, y^{2^e})

Example (Scalar multiplication on NIST K-233)

k given in width-2 τNAF $\Rightarrow w(k) \approx m/3$
Point addition takes $8M + 13$ clock cycles (based on existing work) and we have a repeated squarer (solution 2) with $0 \leq e \leq 11$:
Speedups: $M = 17 \Rightarrow 3.8\,\%$, $M = 8 \Rightarrow 7.0\,\%$, and $M = 5 \Rightarrow 9.7\,\%$

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Scalar multiplication on Koblitz curves

- Scalar multiplication on Koblitz curves, kP where $k = \sum_{i=0}^{\ell-1} k_i \tau^i$, computed with the binary algorithm: $w(k)$ point additions and $\ell - 1$ Frobenius maps
- Frobenius map: $(x, y) \mapsto (x^2, y^2)$
- e successive Frobenius maps can be computed with two repeated squarings: (x^{2^e}, y^{2^e})

Example (Scalar multiplication on NIST K-233)

k given in width-2 τNAF $\Rightarrow w(k) \approx m/3$
Point addition takes $8M + 13$ clock cycles (based on existing work) and we have a repeated squarer (solution 2) with $0 \leq e \leq 11$:
Speedups: $M = 17 \Rightarrow 3.8\,\%$, $M = 8 \Rightarrow 7.0\,\%$, and $M = 5 \Rightarrow 9.7\,\%$

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Information and Computer Science

Side-channel resistivity

Problem

Computing e Frobenius maps takes $2e$ clock cycles which can be distinguished simply by counting clock cycles from the power trace (confer, weaknesses of the normal binary algorithm).
$\Rightarrow$ Leaks the positions of nonzero k_i

Solution

Use repeated squarers
$\Rightarrow$ the attacker sees only a series of point additions and (two) repeated squarings
$\Rightarrow$ the attacker must be able to disinguish e from the power trace of the repeated squarer (one clock cycle)

Side-channel resistivity

Problem

Computing e Frobenius maps takes $2e$ clock cycles which can be distinguished simply by counting clock cycles from the power trace (confer, weaknesses of the normal binary algorithm).
$\Rightarrow$ Leaks the positions of nonzero k_i

Solution

Use repeated squarers
$\Rightarrow$ the attacker sees only a series of point additions and (two) repeated squarings
$\Rightarrow$ the attacker must be able to disinguish e from the power trace of the repeated squarer (one clock cycle)

Summary

A new component called repeated squarer computing $a^{2^e}(x)$ directly in one clock cycle was introduced

- Small and fast enough to be used in existing finite field processors on FPGAs
- Improves the speed of inversion and scalar multiplication on Koblitz curves
- Enhances resistivity against side-channel attacks

Thank you.
Questions?