An Implementation of SPELT(31, 4, 96, 96, (32, 16, 8))

Tung Chou

January 5, 2012
Stream cipher. Security relies on $MQ$ (Multivariate Quadratics).
Stream cipher. Security relies on $MQ$ (Multivariate Quadratics).

With multivariate quadratic systems $P, Q$, generate key stream $y_0, y_1, y_2, \ldots$

$\begin{align*}
  x_0 &\rightarrow x_1 = Q(x_0) \rightarrow x_2 = Q(x_1) \rightarrow x_3 = Q(x_2) \rightarrow \cdots \\
  y_0 = P(x_0) &\quad y_1 = P(x_1) &\quad y_2 = P(x_2) &\quad y_3 = P(x_3) &\quad \cdots
\end{align*}$
Stream cipher. Security relies on MQ (Multivariate Quadratics).

With multivariate quadratic systems $P$, $Q$, generate key stream $y_0, y_1, y_2, \ldots$

\[
\begin{align*}
x_0 &\rightarrow x_1 = Q(x_0) \rightarrow x_2 = Q(x_1) \rightarrow x_3 = Q(x_2) \rightarrow \cdots \\
y_0 = P(x_0) &\rightarrow y_1 = P(x_1) \rightarrow y_2 = P(x_2) \rightarrow y_3 = P(x_3) \rightarrow \cdots
\end{align*}
\]

Simply speaking, QUAD is polynomial evaluations.
Security relies on $\mathcal{MP}$; i.e., $P, Q$ are sparse.
Security relies on \( SMP \); i.e., \( P, Q \) are \textit{sparse}.

Usually of higher degree than QUAD.
- Security relies on $\mathcal{MP}$; i.e., $P, Q$ are sparse.
- Usually of higher degree than QUAD.
- Example: SPELT(31, 4, 96, 96, (32, 16, 8)):
  - Field: $\mathbb{F}_{31}$, Degree: 4, #Variables: 96, #Equations: 96 (for each of $P$, $Q$)
  - Each equation has only 32 degree-2 terms, 16 degree-3 terms, 8 degree-4 terms

Tung Chou
An Implementation of SPELT(31, 4, 96, 96, (32, 16, 8))
Security relies on $SMP$; i.e., $P, Q$ are sparse.

Usually of higher degree than QUAD.

Example: SPELT(31, 4, 96, 96, (32, 16, 8)):
  - Field: $\mathbb{F}_{31}$, Degree: 4, #Variables: 96, #Equations: 96 (for each of $P, Q$)
  - Each equation has only 32 degree-2 terms, 16 degree-3 terms, 8 degree-4 terms

More efficient than QUAD in practice.
15 \times 32 = 480 \text{ SPs (cores) running at 1.4 GHz (32 SPs in each MP).}
Implementation Platform: GTX480

- $15 \times 32 = 480$ SPs (cores) running at 1.4 GHz (32 SPs in each MP).
- Each MP has 16 KB L1 cache and 48 KB shared memory (the sizes can be switched).

Tung Chou
An Implementation of SPELT\((31, 4, 96, 96, (32, 16, 8))\)
Implementation Platform: GTX480

- $15 \times 32 = 480$ SPs (cores) running at 1.4 GHz (32 SPs in each MP).
- Each MP has 16 KB L1 cache and 48 KB shared memory (the sizes can be switched).
- Each MP has 32K registers.
Implementation Platform: GTX480

- $15 \times 32 = 480$ SPs (cores) running at 1.4 GHz (32 SPs in each MP).
- Each MP has 16 KB L1 cache and 48 KB shared memory (the sizes can be switched).
- Each MP has 32K registers.
- 32 memory banks in shared memory.
15 \times 32 = 480 \text{ SPs (cores) running at 1.4 GHz (32 SPs in each MP).}

Each MP has 16 KB L1 cache and 48 KB shared memory (the sizes can be switched).

Each MP has 32K registers.

32 memory banks in shared memory.

The maximal number of registers assigned to each threads is 64.
Threads in a warp deal with the same equation(s) but different sets of $x_i$’s. In other words, each block generates 32 key streams at the same time.
Threads in a warp deal with the same equation(s) but different sets of $x_i$’s. In other words, each block generates 32 key streams at the same time.

Information of each term is written in instructions.

We need 96×32 bytes. This is augmented into 100×32 to avoid bank conflicts.

There are two buffers in shared memory, serving as source and destination.

The results of the last 96 equations ($Q$) are written to global memory.

DIMGRID=30, DIMBLOCK=512. This means each warp has to deal with 192/16=12 equations.
Threads in a warp deal with the same equation(s) but different sets of $x_i$’s. In other words, each block generates 32 key streams at the same time.

Information of each term is written in instructions.

Values of $x_i$ are stored in shared memory.

- We need $96 \times 32$ bytes. This is augmented into $100 \times 32$ to avoid bank conflicts.
- There are two buffers in shared memory, serving as source and destination.
- The results of the last 96 equations ($Q$) are written to global memory.
Threads in a warp deal with the same equation(s) but different sets of $x_i$’s. In other words, each block generates 32 key streams at the same time.

Information of each term is written in instructions.

Values of $x_i$ are stored in shared memory.

- We need $96 \times 32$ bytes. This is augmented into $100 \times 32$ to avoid bank conflicts.
- There are two buffers in shared memory, serving as source and destination.
- The results of the last 96 equations ($Q$) are written to global memory.

DIMGRID=$30$, DIMBLOCK=$512$. This means each warp has to deal with $192/16=12$ equations.
Each block uses $\leq 64$ KB shared. Each thread uses $\leq 32$ regs. Therefore each MP should be able to run two blocks (32 warps) simultaneously.
Experiment Results

- Each block uses \( \leq 64 \text{ KB} \) shared. Each thread uses \( \leq 32 \text{ regs.} \). Therefore each MP should be able to run two blocks (32 warps) simultaneously.
- Performance: 1.38 Gbps.
  - Good news: Better than the previous result: 0.91 Gbps.
  - Bad news: Peak performance should be 6.99 Gbps if we consider multiplications only.
Experiment Results

- Each block uses $\leq 64$ KB shared. Each thread uses $\leq 32$ regs. Therefore each MP should be able to run two blocks (32 warps) simultaneously.
- Performance: **1.38** Gbps.
  - Good news: Better than the previous result: 0.91 Gbps.
  - Bad news: Peak performance should be **6.99** Gbps if we consider *multiplications* only.
- Mysterious behaviors of *nvcc* make it hard to find the bottleneck.
Tweaks to Accelerate the Evaluations

- Total number of mults: $96 + 32 \cdot 2 + 16 \cdot 3 + 8 \cdot 4 = 240$.

- Classifying terms by $x_i$'s.
  
  $7x_0x_1x_4 + 29x_1 \longrightarrow x_1 \cdot (7x_0x_4 + 29)$

  Saving at least $32 + 16 + 8 = 56$ mults.

- Classifying terms by coefficients.
  
  $14x_0x_1 + 14x_3x_9 \longrightarrow 14 \cdot (x_0x_1 + x_3x_9)$

  Saving at least $(32 + 16 + 8) + (96 - 30) = 122$ mults.

- A mixed approach
Future Works

- AMD GPUs