Towards Practical Tools for Side Channel Aware Software Engineering: ‘Grey Box’ Modelling for Instruction Leakages

David McCann, Elisabeth Oswald, Carolyn Whitnall*

Department of Computer Science, University of Bristol

carolyn.whitnall@bris.ac.uk

August 2017
**Top line:** Simulating instruction-level power consumption to detect side-channel leakage in the development stage.

```
str r4, [r1, #0x4]
ldr r4, [r1, #0x8]
ror r4, r6
str r4, [r1, #0x8]
ldr r4, [r1, #0xC]
```

**Outline:**
- Motivation: what are we hoping to achieve?
- Previous work: what has been achieved already?
- Model building strategy.
- ELMO: a leakage emulator for the M0.
- Performance and applications.
Motivation

Equip software designers to detect and address side-channel vulnerabilities during development.

- Avoids nasty surprises later down the line.
- Problems are cheaper to fix early on.
- Reduces the reliance on (external) expertise.

<table>
<thead>
<tr>
<th>Smart Cards</th>
<th>Internet of Things</th>
</tr>
</thead>
<tbody>
<tr>
<td>A relatively mature industry.</td>
<td>A new, rapidly growing industry.</td>
</tr>
<tr>
<td>Large, established companies.</td>
<td>Lots of small start-ups.</td>
</tr>
<tr>
<td>Security already a priority.</td>
<td>Security not well-incentivised.</td>
</tr>
</tbody>
</table>
Previous Work

Detailed circuit-level knowledge – transistor/cell-level netlists. Transition counts as a proportional approximation of the power consumption. More precise mappings possible if capacitive loads are known. Always a simplification to some degree, e.g. netlists don’t describe crosstalk.
PREVIOUS WORK

- Detailed circuit-level knowledge – transistor/cell-level netlists.
- Transition counts as a proportional approximation of the power consumption.
- More precise mappings possible if capacitive loads are known.
- Always a simplification to some degree, e.g. netlists don’t describe crosstalk.
PREVIOUS WORK

- Only requires general knowledge of the algorithm, rather than implementation specifics.
- Emphasis is on known security-sensitive values and operations.
- Models are fitted empirically to trace measurements as inputs vary.
- Fitted distributions used (e.g.) as ‘templates’ for classification.
Previous Work

White box

- Instruction level simulation.
- Exploits assembly code but doesn’t require circuit-level info.
- Power consumption typically modelled via simplified assumptions such as Hamming weight/distance.
- Power cost analysis indicates greater complexity in practice (e.g. sequence effects) [TMWL96].

Black box

Previous Work

- Instruction level simulation.
- Exploits assembly code but doesn’t require circuit-level info.
- Power consumption typically modelled via simplified assumptions such as Hamming weight/distance.
- Power cost analysis indicates greater complexity in practice (e.g. sequence effects) [TMWL96].

Our contribution: appropriately complex models for instruction-level (i.e. grey box) leakages; ELMO – a tool which uses these models to map code sequences to predicted traces.
LINEAR REGRESSION

dependent variable = linear function (explanatory variables) + noise

power consumption = linear function (data and operations) + other processes + measurement error + etc...

\[ T = \alpha + X\beta + \varepsilon \]

Ordinary Least Squares estimation: Find \( \hat{\alpha} \) and \( \hat{\beta} \) so as to minimise the sum of the squares of the residuals (that is, \( \sum_{i=1}^{N} T - (\hat{\alpha} + \hat{\beta}x_i))^2 \)).

Some assumptions:

- Noise is independent and constant.
- The explanatory variables are not collinear.
- (For some statistical inferences) noise is normally distributed.
How do we know when we’ve arrived at something suitable for purpose?
Model ‘quality’ = closeness to the underlying reality it approximates.
Underlying reality = unknown $\Rightarrow$ Model ‘quality’ = unknown.

$$R^2 = \frac{\text{variation explained by the model}}{\text{total variation}}$$

High $R^2$ can imply overfit, especially when # explanatory variables is large relative to # measurements.
$R^2$ always increases with additional explanatory variables, even if the contribution is not significant.
**Measuring Model Quality: The F-Test**

**F-test (informally):** a set of variables is jointly significant if

\[
\frac{\text{reduction in unexplained variation}}{\text{remaining unexplained variation}} > \text{critical value of F-distribution}
\]

Can also test for overall significance of model.

- Low $R^2 \implies$ variation is dominated by unknown / unmeasureable / un-included factors.
- Low $R^2$ + overall significance $\implies$ model might still be useful.
- Model not significant $\implies$ traces may not leak information of interest.
- Model not significant $\not\implies$ device does not leak information! **Review acquisition set-up…**
Launched in 2004 for use within small microcontrollers.

Six variants: M0, M0+, M1, M3, M4, M7 (lowest to highest in cost / size / power).

Exact CPU architecture not publicly available but known approximately:
- Arithmetic logic unit (ALU).
- Hardware multiplier.
- Barrel shifter.
- Two buses to the ALU from the register banks.
- Third bus from ALU output back to the register banks.
ARM CORTEX-M PROCESSOR FAMILY

- Launched in 2004 for use within small microcontrollers.
- Six variants: M0, M0+, M1, M3, M4, M7 (lowest to highest in cost / size / power).
- Exact CPU architecture not publicly available but known approximately:
  - Arithmetic logic unit (ALU).
  - Hardware multiplier.
  - Barrel shifter.
  - Two buses to the ALU from the register banks.
  - Third bus from ALU output back to the register banks.

**Fig:** Simplified ARM CPU architecture. (Redrawn from S. Furber, *ARM System-on-Chip Architecture*, Addison Wesley, 2000).
Launched in 2004 for use within small microcontrollers.

Six variants: M0, M0+, M1, M3, M4, M7 (lowest to highest in cost / size / power).

Exact CPU architecture not publicly available but known approximately:

- Arithmetic logic unit (ALU).
- Hardware multiplier.
- Barrel shifter.
- Two buses to the ALU from the register banks.
- Third bus from ALU output back to the register banks.

**Fig:** Simplified ARM CPU architecture. (Re-drawn from S. Furber, *ARM System-on-Chip Architecture*, Addison Wesley, 2000).
Launched in 2004 for use within small microcontrollers.

Six variants: M0, M0+, M1, M3, M4, M7 (lowest to highest in cost / size / power).

Exact CPU architecture not publicly available but known approximately:

- Arithmetic logic unit (ALU).
- Hardware multiplier.
- Barrel shifter.
- Two buses to the ALU from the register banks.
- Third bus from ALU output back to the register banks.

**Fig:** Simplified ARM CPU architecture. (Redrawn from S. Furber, *ARM System-on-Chip Architecture*, Addison Wesley, 2000).
Launched in 2004 for use within small microcontrollers.

Six variants: M0, M0+, M1, M3, M4, M7 (lowest to highest in cost / size / power).

Exact CPU architecture not publicly available but known approximately:

- Arithmetic logic unit (ALU).
- Hardware multiplier.
- Barrel shifter.
- Two buses to the ALU from the register banks.
- Third bus from ALU output back to the register banks.

**Fig:** Simplified ARM CPU architecture. (Redrawn from S. Furber, *ARM System-on-Chip Architecture*, Addison Wesley, 2000).
Launched in 2004 for use within small microcontrollers.

Six variants: M0, M0+, M1, M3, M4, M7 (lowest to highest in cost / size / power).

Exact CPU architecture not publicly available but known approximately:

- Arithmetic logic unit (ALU).
- Hardware multiplier.
- Barrel shifter.
- Two buses to the ALU from the register banks.
- Third bus from ALU output back to the register banks.

**FIG:** Simplified ARM CPU architecture. (Redrawn from S. Furber, *ARM System-on-Chip Architecture*, Addison Wesley, 2000).
INSTRUCTIONS

THUMB INSTRUCTION SET

A subset of the most commonly used ARM instructions, each 16 bits long. We select 21 of these according to typical implementations of symmetric crypto:

**ALU:** eors, ands, adds, adds $\textit{imm}$, subs, subs $\textit{imm}$, orrs, cmp, cmp $\textit{imm}$, movs, movs $\textit{imm}$.

**Shifts:** lsls, lsrs, rors.

**Stores:** str, strb, strh.

**Loads:** ldr, ldrb, ldrh.

**Multiply:** muls.

THUMBULATOR

An open source C program that emulates the data flow of arbitrary Thumb instruction sequences.

https://github.com/dwelch67/thumbulator.git/
**Measurement Set-up**

**ARM Cortex-M0** implemented on an STM32F0 (30R8T6) Discovery Board:

- ST-Link to flash programs to the processor.
- On-chip debugging capabilities.
- On-board 8MHz RC oscillator clock signals.

**Modifications** (to minimise board and set-up effects):

- Extract power pins.
- Pass power supply through a 360Ω resistor over which a differential probe is connected.

**Oscilloscope**: Lecroy Waverunner 700 Zi.

- Sampling rate of 500 MS/S chosen empirically according to quality of DPA outcomes.
- Clock speed set to 8MHz.
- Five measurements acquired and averaged per input, to reduce noise.
Identifying Basic Leakage Characteristics

**Aim:** Examine and compare basic data-dependent power consumption characteristics of the 21 instructions.

**Data collection:** 21 $\times$ 5,000 traces of the form `mov-instr-mov`, with random inputs.

**Point selection:** Maximum peak of the clock cycle in which `instr` was performed.

**Model:**

$$\text{power consumption} = \alpha + \left[ \begin{array}{c} \text{operand bits} \\ \text{bit-flips} \end{array} \right] \beta + \varepsilon$$

**Findings:** All models overall significant. Form of the leakage varies:

- Loads only depend (jointly) on operand bits, not bit-flips.
- Stores only depend on 2nd operand bits and bit-flips.
- Multiplication doesn’t depend on 2nd operand bit-flips.
- Instructions on immediates essentially have no 2nd operand.
**Aim:** Remove redundant complexity $\implies$ increase scope for explanatory complexity.

**Experiment:** Hierarchical clustering analysis of the data-dependent coefficients for each modelled instruction.

**Findings:** Best quality clustering according to the average silhouette index confirms intuition.
Cluster Analysis

**Aim:** Remove redundant complexity $\implies$ increase scope for explanatory complexity.

**Experiment:** Hierarchical clustering analysis of the data-dependent coefficients for each modelled instruction.

**Findings:** Best quality clustering according to the average silhouette index confirms intuition.
Data-dependent coefficients for representative instructions...
Data-dependent coefficients for representative instructions versus average coefficients within instruction groups.
**Aim:** Control for the effect of previous and subsequent instructions on power consumption.

**Data collection:** $125 \times 1,000$ traces for each triplet (interleaved within one acquisition) with random inputs.

**Point selection:** Maximum peak of the clock cycle most strongly associated with the inputs to the target instruction.

**Model:** Same as before, plus:
- Previous and subsequent instructions (8 dummies, 2 baseline categories).
- Previous instruction $\times$ Hamming weight of each operand, and the same for subsequent instruction (16 continuous interactions).
- Previous instruction $\times$ Hamming distance of each operand, and the same for subsequent instruction (16 continuous interactions).

**Findings:** Significant differential data effects in almost all cases (except operand 1 and transition 2 terms for $\text{str}$).
Allowing for Nonlinearity in Data Dependency

Limitation of the model so far: Restricts the relationship between the leakage and the data bits/bit-flips to be linear.

- Bits carried on adjacent wires (e.g.) might produce an interaction effect.

Solution: Test for inclusion of adjacent and non-adjacent interactions.

Findings:

- Significant bit interactions for `lsls` and `muls` only – (i.e. instructions which involve explicit joint manipulation of bits).
- No significant (adjacent) bit-flip interactions found.

<table>
<thead>
<tr>
<th>Model Term</th>
<th>eor</th>
<th>lsl</th>
<th>str</th>
<th>ldr</th>
<th>mul</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data (2 × 32 bits, 2 × 32 bit-flips)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Instructions (2 × 4 dummies)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Data × Instructions (2 × 4 × 2 × 2 interactions)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Data × Data (2 × (\binom{32}{2}) interactions)</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>
**Integration into Leakage Emulator**

**Have:** Five model equations for predicting the power consumption for each of our representative Thumb instructions.

**Want:** Facility to predict power consumption for arbitrary code sequences.

---

**Thumbulator**

- Takes binary assembly code as input.
- Decodes and executes sequentially.
- Traces instruction and memory flow for purposes of debugging.

---

**ELMO**

- Adapt Thumbulator to also store the values of the operands in a linked list data structure.
- Map data flow information to power consumption predictions using the model equations.
How well do the model predictions correlate with measured traces?

- Correlation traces for ELMO-predicted intermediate values (top) and Hamming weight model predictions (bottom) in 500 real traces.
Model Correlation with Real Leakages

How well do the model predictions correlate with measured traces?

![Correlation traces for ELMO-predicted intermediate values (top) and Hamming weight model predictions (bottom) in 500 real M0 traces.](image)
Using ELMO for Leakage Detection

Fixed-versus-random leakage detection:
- Proposed (among other tests) by Goodwill et al. at the 2008 NIST attack testing workshop.
- Two sets of traces are collected from the target device, one with some fixed data input, one with random inputs.
- Welch’s 2-sample t-test used to decide whether (at each time point) the traces are significantly different.

Example: Thumb assembly implementation of masked ShiftRows...

<table>
<thead>
<tr>
<th>Cycle No.</th>
<th>Address</th>
<th>Machine Code</th>
<th>Assembly Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>0x08000206</td>
<td>0x684C</td>
<td>ldr r4, [r1, #0x4]</td>
</tr>
<tr>
<td>3</td>
<td>0x08000208</td>
<td>0x41EC</td>
<td>ror r4, r5</td>
</tr>
<tr>
<td>4-5</td>
<td>0x0800020A</td>
<td>0x604C</td>
<td>str r4, [r1, #0x4]</td>
</tr>
<tr>
<td>6-7</td>
<td>0x0800020C</td>
<td>0x688C</td>
<td>ldr r4, [r1, #0x8]</td>
</tr>
<tr>
<td>8</td>
<td>0x0800020E</td>
<td>0x41F4</td>
<td>ror r4, r6</td>
</tr>
<tr>
<td>9-10</td>
<td>0x08000210</td>
<td>0x608C</td>
<td>str r4, [r1, #0x8]</td>
</tr>
<tr>
<td>11-12</td>
<td>0x08000212</td>
<td>0x68CC</td>
<td>ldr r4, [r1, #0xC]</td>
</tr>
<tr>
<td>13</td>
<td>0x08000214</td>
<td>0x41FC</td>
<td>ror r4, r7</td>
</tr>
<tr>
<td>14-15</td>
<td>0x08000216</td>
<td>0x60CC</td>
<td>str r4, [r1, #0xC]</td>
</tr>
</tbody>
</table>
Fixed-versus-random leakage detection:

- Proposed (among other tests) by Goodwill et al. at the 2008 NIST attack testing workshop.
- Two sets of traces are collected from the target device, one with some fixed data input, one with random inputs.
- Welch’s 2-sample t-test used to decide whether (at each time point) the traces are significantly different.

Example: Thumb assembly implementation of masked ShiftRows...

<table>
<thead>
<tr>
<th>Cycle No.</th>
<th>Address</th>
<th>Machine Code</th>
<th>Assembly Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>0x08000206</td>
<td>0x684C</td>
<td>ldr r4, [r1, #0x4]</td>
</tr>
<tr>
<td>3</td>
<td>0x08000208</td>
<td>0x41EC</td>
<td>ror r4, r5</td>
</tr>
<tr>
<td>4-5</td>
<td>0x0800020A</td>
<td>0x604C</td>
<td>str r4, [r1, #0x4]</td>
</tr>
<tr>
<td>6-7</td>
<td>0x0800020C</td>
<td>0x688C</td>
<td>ldr r4, [r1, #0x8]</td>
</tr>
<tr>
<td>8</td>
<td>0x0800020E</td>
<td>0x41F4</td>
<td>ror r4, r6</td>
</tr>
<tr>
<td>9-10</td>
<td>0x08000210</td>
<td>0x608C</td>
<td>str r4, [r1, #0x8]</td>
</tr>
<tr>
<td>11-12</td>
<td>0x08000212</td>
<td>0x68CC</td>
<td>ldr r4, [r1, #0xC]</td>
</tr>
<tr>
<td>13</td>
<td>0x08000214</td>
<td>0x41FC</td>
<td>ror r4, r7</td>
</tr>
<tr>
<td>14-15</td>
<td>0x08000216</td>
<td>0x60CC</td>
<td>str r4, [r1, #0xC]</td>
</tr>
</tbody>
</table>
**Fig:** Fixed vs random t-tests against the simulated power consumption of masked ShiftRows. (Dotted lines indicate the ±4.5 threshold for t-test significance).

- Leakage found in the ELMO traces where expected.
- Same test against real measurements closely matches.
- Tying leakage to instructions (not clock cycles) degrades visual similarity but aids diagnosis.
Summary and Further Work

Achievement:
- We have presented a tool which...
  - Is able to capture vulnerabilities in arbitrary code sequences.
  - Replicates leakages that go undiscovered in simulations relying on standard assumptions such as the Hamming weight.
  - Can be used in place of real measurements to pre-empt problems in the software development stage.
- Methodology generalises for use with different devices and side-channels.

Possible applications:
- Automated insertion and testing of countermeasures.
- Optimisation of protected code with respect to energy efficiency.

Available at https://github.com/bristol-sca/ELMO.