UNIVERSITY OF CAMBRIDGE
Department of Engineering

Compiling Physical Invariants to Hardware for a Secure & Private Sensor Interface

Author Name: Gregory Brooks

Supervisor: Phillip Stanley-Marbell

Date:

I hereby declare that, except where specifically indicated, the work submitted herin is my own original work.

Signed _____ *date* _____

IIB Project Report:
Compiling Physical Invariants to Hardware for a Secure & Private Sensor Interface

Gregory Brooks, gb510, Christ's College

Contents

1 Technical Abstract

Compiling Physical Invariants to Hardware for a Secure & Private Sensor Interface

Gregory Brooks, gb510, Christ's College

1.1 System Diagram

⟨ *TODO: insert block diagram of system and describe which parts have been the focus of this project*⟩

⟨ *TODO: write this last*⟩

2 Introduction

3 Theoretical Development

3.1 Brainstorming Example Applications

3.1.1 *Intelligent* Noising

⟨ *TODO: allowing the compiler to choose how to add noise, rather than explicity stating where to add noise*⟩

3.1.2 Digital Camera/Viola-Jones Facial Detection

3.1.3 Microphone

3.2 System Specification

⟨ *TODO: move to technical abstract?*⟩

3.3 Multi-Sensor Differential Privacy Loss

3.4 Privacy Budget Management Algorithm

In the following section, I propose a privacy management system architecture that could be used to manage multi-sensor privacy on an FPGA. Due to project time constraints, the system has not been completely implemented, since this would divert time and attention away from investigating the more important research questions posed by the project.
⟨ *TODO: describe what will have been implemented by the submission date e.g. use privacy yaml file and Newton AST to generate some of the logic implementation by filling in a Verilog template*⟩
The proposed system architecture is as follows:

- The FPGA acts as a sensor interface, allowing external circuitry to request noised sensor values. Noised measurement signals will be referred to as *protected*, as their true value has been masked to preseve privacy; the system also allows *unprotected* signals to be forwarded to the outside world with no random noise added.
- A *privacy file* (e.g. a YAML file) is used to specify protected measurement signals for a particular embedded system.
- In order to apply random noise to a particular signal, the FPGA requires a range of parameters to be defined for each protected signal:

  - Range of possible sensor outputs.
  - Privacy *budget* value allocated to the sensor.
  - Privacy *budget* replenishment rate.
  - ϵ, a measure of privacy where a smaller value results in greater privacy by increasing the variance of applied noise.

  These values can be assigned to protected signals in the privacy file.

- The FPGA maintains a differential privacy *budget* for each protected signal. Querying a value for a protected signal (i.e. requesting a noised measurement from the FPGA) causes that signal to incur a privacy loss. The FPGA quantifies this loss (a function of the random noise sample added to the raw measurement [**Choi2018GuaranteeingLD**]) and subtracts it from the signal's privacy budget. Once the budget is depleted, further queries of the signal in question are ignored — a nonzero budget replenishment rate can be specified in the privacy file so that the signal can eventually be queried again once the budget has replenished sufficiently.

- Invariants in the Newton description define relationships between signals. This is important as privacy is lost if a measurand's value is calculated by taking measurements from related sensors, without directly measuring the measurand. The system must account for this *indirect* privacy loss:

  - Each Newton invariant containing protected measurement signals has an associated bitfield register on the FPGA. Each bit in the register acts as a read flag for each measurement signal contained in the associated invariant — the flag is set when the sensor is queried (i.e. when a measurement is taken and the noised measurement value becomes *known* to the outside world) and is not cleared until the sensor's privacy budget is completely replenished some time after a query (at which point the measurement value can be considered *unknown* to the outside world).
  - As long as any two flags remain unset, there are two variables in the equation that are *unknown* to an attacker. The equation cannot be solved and so no information[1] can be gained about the *unknown* values.

[1]This assumes that the probability distributions of sensor outputs are independent, see Appendix B for an investigation into correlated measurements.

- If one of the final two *unknown* values is then queried, then the equation becomes solvable for the final *unknown* value — this measurement incurs a privacy loss. The mathematics behind the calculation of this privacy loss is investigated in Appendix B; whilst calculation of an exact value for privacy loss (Equation 4 in Appendix B) requires the use of a mathematical optimisation technique such as Lagrange multipliers, a simpler calculation can be used to obtain an upper bound on the value. This simpler calculation can be performed more easily on a small FPGA such as the iCE40.
- Each subsequent measurement of a 'known' value
- If responding to a query would exceed the privacy budget for any protected signal (not just the one being queried) then the response should not be granted.

4 Apparatus, Equipment and Techniques

4.1 GitHub

⟨ *TODO: list GitHub repositories*⟩ ⟨ *TODO: describe how GitHub was used*⟩ ⟨ *TODO: include an example of commit history for a repository*⟩

4.2 iCE40 FPGA

4.3 Software Prototyping

4.4 FPGA Synthesis

4.5 Repository Structure and Build System

4.6 Verilog Development Techniques

⟨ *TODO: describe good practices picked up over the course of this project*⟩

5 Deliverables

5.1 Hardware Entropy Source

One of the key components of a security/privacy application is a true random number generator (TRNG). Unlike a pseudorandom number generator (PRNG), which produces a predictable deterministic outcome, a true random number generator's output cannot be anticipated by an attacker, preserving system security and users' privacy. The output rate of TRNGs is often limited so can be used to seed a cryptographically secure PRNG (CSPRNG) algorithm to increase output data rate without significantly compromising security. This extra step was not required for this project, since the rate at which random numbers are consumed is relatively low.

I investigated using the iCE40 FPGA's differential I/O hardware to generate random bits which could be fed into a TRNG to generate random samples from an arbitrary distribution. Initially, it was hoped that simply leaving two comparator inputs floating would cause the output to fluctuate randomly — in practice this did not occur as the comparator hardware requires one of the inputs to be biased within a narrow common mode voltage range (roughly half the I/O supply voltage, allowing the other input to swing about this reference voltage) [**ice40˙diff˙io**]. I decided to design a PCB to set this bias voltage, as well as provide an alternative entropy source in the form of a reverse biased zener diode producing avalanche noise. In theory, this circuit would provide a 'better' i.e. less predictable source of entropy since, unlike the floating input pin, the avalanche noise (and therefore random number output) produced is independent of the device's external environment[2].

The circuit used in this noise generator is based on a design by Professor Paul Horowitz ⟨ *TODO: cite book*⟩, modified to operate at 3.3V and 0V power rails (rather than $\pm 5V$). Avalanche noise produced by a reverse biased zener diode is amplified through a dual op-amp amplifier stage before being modulated about a 1.65V DC bias $\left(\frac{V_{cc}}{2}\right)$; this output can then be fed into one of the iCE40's comparator inputs. A second output on the PCB comes from a simple potentiometer circuit to set the common mode reference voltage on the second comparator input (with series resistors to allow for fine control about $\frac{V_{cc}}{2}$). Since the comparator output depends on whether the instantaneous voltage of the noise waveform lies above or below this reference voltage, this reference point can be adjusted to set the ratio of ones to zeros in the random comparator output. ⟨ *TODO: finish e.g. talk about zener circuit specs*

[2] A floating input pin can easily be influenced by its external environment e.g. picking up 50Hz noise from nearby electrical mains, resulting in a non-uniform frequency spectrum for 'random' output.

(50MHz noise bandwidth?)⟩

⟨ *TODO: finish*⟩

⟨ *TODO: cite Horowitz and Hill*⟩

The full circuit schematic and PCB layout can be found in Appendix E, as well as the GitHub repository listed in Section 4.1.

5.2 Uniform Random Number Generator

⟨ *TODO: insert block diagram*⟩

5.3 Inversion Method Random Number Generator

⟨ *TODO: insert block diagram*⟩

6 Results

6.1 Scaling of Logic Implementations

⟨ *TODO: investigate how URNG and RNG logic implementations scale with various parameters (and how this compares to a naive implementation)*⟩

⟨ *TODO: investigate scaling of adder hardware (inferred from Verilog vs iCE40 hardware accumulators)*⟩

7 Conclusions

A Risk Assesment Retrospective

The risk assessment submitted at the start of the project does not mention any specific hazards besides office (computer) work, since the project is predominately software/firmware based (all project hardware operated at low voltages i.e. 12V or less). No other hazards were encountered during the course of the project, since the random noise generator PCBs were manufactured by the Dyson Centre's Electronics Development Group. In retrospect, although not part of the initial project specification, the risk assessment could have anticipated the possibility of manufacturing PCBs for the project. The hazards associated with this activity include high temperatures (from a soldering iron/oven/hot air gun) as well as chemical hazards associated with solder, fume extraction etc.

B Derivation of an Upper Bound on *Indirect* Differential Privacy Loss

Let $\hat{X}$ be a random variable denoting a sensor measurement, including any measurement error.

Let X be a random variable representing a *noised* sensor measurement i.e. the masked value that the differential privacy system provides to the outside world after applying Laplace distributed noise to a measurement:

$$X = \hat{X} + N_{Laplace} \tag{1}$$

A measurement event comprises the variable $\hat{X}$ taking on value $\hat{x}_i$. Similarly, a noising event can be defined by X taking value $x_i = \hat{x}_i + n_i$.

Let Y denote a variable derived from one or more X variables, such as measurements from different sensors in an embedded system:

$$Y = f(X_1, X_2, ..., X_n) = f(\mathbf{X})$$
$$y_i = f(x_{i1}, x_{i2}, ..., x_{in}) = f(\mathbf{x_i})$$

This same mapping function $f(\mathbf{x})$ can take unnoised measurements as arguments:

$$\hat{Y} = f(\hat{X}_1, \hat{X}_2, ..., \hat{X}_n) = f(\hat{\mathbf{X}})$$
$$\hat{y}_i = f(\hat{x}_{i1}, \hat{x}_{i2}, ..., \hat{x}_{in}) = f(\hat{\mathbf{x}}_\mathbf{i})$$

If measurements x_{i1} to x_{in} are taken, a value for y_i can be computed from them — variable Y has experienced some privacy loss, l_Y that can be calculated as a log-likelihood ratio [**Choi2018GuaranteeingLD**] (Equation 3). This privacy loss function requires two parameters, $\hat{y}_a$ and $\hat{y}_b$, which are the possible *true* i.e. unnoised values for $\hat{Y}$ that maximise the privacy loss (Equation 2).

$$y_{obs} = f(\mathbf{x_{obs}}) = \text{value for } y \text{ calculated from noised measurements } \mathbf{x_i}$$

$$\hat{y}_a, \hat{y}_b = \underset{\hat{y}_a, \hat{y}_b}{\operatorname{argmax}}\, l_Y(\hat{y}_a, \hat{y}_b) \tag{2}$$

$$l_Y(\hat{y}_a, \hat{y}_b) = log\left(\frac{Pr\{Y = y_{obs}|\hat{Y} = \hat{y}_a\}}{Pr\{Y = y_{obs}|\hat{Y} = \hat{y}_b\}}\right) \tag{3}$$

Equation 3 can be rewritten as follows, where $\hat{\mathbf{x}}_\mathbf{a}$ and $\hat{\mathbf{x}}_\mathbf{b}$ are chosen to maximise l_Y as before, but subject to constraints $\hat{\mathbf{x}}_\mathbf{a} = f^{-1}(\hat{y}_a)$ and $\hat{\mathbf{x}}_\mathbf{b} = f^{-1}(\hat{y}_b)$:

$$l_Y(\hat{\mathbf{x}}_\mathbf{a}, \hat{\mathbf{x}}_\mathbf{b}) = log\left(\frac{Pr\{\mathbf{X} = \mathbf{x_{obs}}|\hat{\mathbf{X}} = \hat{\mathbf{x}}_\mathbf{a}\}}{Pr\{\mathbf{X} = \mathbf{x_{obs}}|\hat{\mathbf{X}} = \hat{\mathbf{x}}_\mathbf{b}\}}\right) \tag{4}$$

This constrained optimisation problem could be solved to calculate an exact value for privacy loss. Alternatively, an upper bound on privacy loss can be determined using a far simpler calculation (the sum of privacy losses for each individual sensor):

$$\begin{aligned} l_Y(\hat{\mathbf{x}}_\mathbf{a}, \hat{\mathbf{x}}_\mathbf{b}) &= log\left(\prod_{u=1}^{n} \frac{Pr\{X_u = x_{u,obs}|\hat{X}_u = \hat{x}_{u,a}\}}{Pr\{X_u = x_{u,obs}|\hat{X}_u = \hat{x}_{u,b}\}}\right) \\ &\leq log\left(\prod_{u=1}^{n} \frac{Pr\{X_u = x_{u,obs}|\hat{X}_u = \hat{x}_{u,c}\}}{Pr\{X_u = x_{u,obs}|\hat{X}_u = \hat{x}_{u,d}\}}\right) \end{aligned} \tag{5}$$

where:

$$\hat{x}_{u,c} = \underset{\hat{x}_{u,c}}{\operatorname{argmax}}\, Pr\{X_u = x_{u,obs}|\hat{X}_u = \hat{x}_{u,c}\}$$

$$\hat{x}_{u,d} = \underset{\hat{x}_{u,d}}{\operatorname{argmin}}\, Pr\{X_u = x_{u,obs}|\hat{X}_u = \hat{x}_{u,d}\}$$

i.e. the constraint $f^{-1}(\hat{y})$ has been removed. Equation 5 can be interpreted as the sum of the privacy losses incurred by the X variables i.e. $\sum_{u=1}^{n} l_{X_u}$. $Pr\{X_u = x_{u,obs}|\hat{X}_u = \hat{x}_u\}$ is simply the Laplace distribution of the noise applied to the measurement represented by $\hat{X}$, centred on value $\hat{x}$.

Since the Newton language recently gained a mutual information operator, I attempted to factor mutual information into this calculation. Unfortunately, it appears that mutual information alone does not provide enough information to calculate indirect privacy loss — the exact nature of the correlation between two (or more) random variables is required i.e. the conditional distribution for the unknown variable given known ones:

Define $\hat{X}$ and X as before and let $\hat{Y}$ be a random variable correlated with $\hat{X}$. Both $\hat{X}$ and $\hat{Y}$ can take values within some range (e.g. due to finite sensor precision):

$$\hat{X} \in R_{\hat{X}},\ \hat{Y} \in R_{\hat{Y}}$$

The mutual information $I(\hat{X};\hat{Y})$ is defined as:

$$I(\hat{X};\hat{Y}) = \int_{R_{\hat{Y}}} \int_{R_{\hat{X}}} p(\hat{X},\hat{Y}) log\left(\frac{p(\hat{X},\hat{Y})}{p(\hat{X})p(\hat{Y})}\right) d\hat{X}d\hat{Y} \tag{6}$$

where $p()$ denotes the probability density function for a random variable. I have been unable to insert this value into the privacy loss equation, however a value for privacy loss can be obtained if the conditional distribution of $\hat{X}$ given $\hat{Y}$ is known, as this allows the conditional distribution of X given $\hat{Y}$ to be calculated:

$$Pr\{X = x|\hat{Y} = \hat{y}\} = \int_{R_{\hat{X}}} Pr\{X = x|\hat{X} = \hat{x}\}Pr\{\hat{X} = \hat{x}|\hat{Y} = \hat{y}\}d\hat{x} \tag{7}$$

Equation 7 can be substituted into Equation 8, the formula for privacy loss incurred by $\hat{Y}$ as a result of observing a value x_{obs} for X. This results in Equation 9:

$$l_y(\hat{y}_a,\hat{y}_b) = log\left(\frac{Pr\{X = x_{obs}|\hat{Y} = \hat{y}_a\}}{Pr\{X = x_{obs}|\hat{Y} = \hat{y}_b\}}\right) \tag{8}$$

$$= log\left(\frac{\int_{R_{\hat{X}}} Pr\{X = x_{obs}|\hat{X} = \hat{x}\}Pr\{\hat{X} = \hat{x}|\hat{Y} = \hat{y}_a\}d\hat{x}}{\int_{R_{\hat{X}}} Pr\{X = x_{obs}|\hat{X} = \hat{x}\}Pr\{\hat{X} = \hat{x}|\hat{Y} = \hat{y}_b\}d\hat{x}}\right) \tag{9}$$

for $\hat{y}_a, \hat{y}_b = \text{argmax}_{\hat{y}_a,\hat{y}_b}(l_y)$ as before.

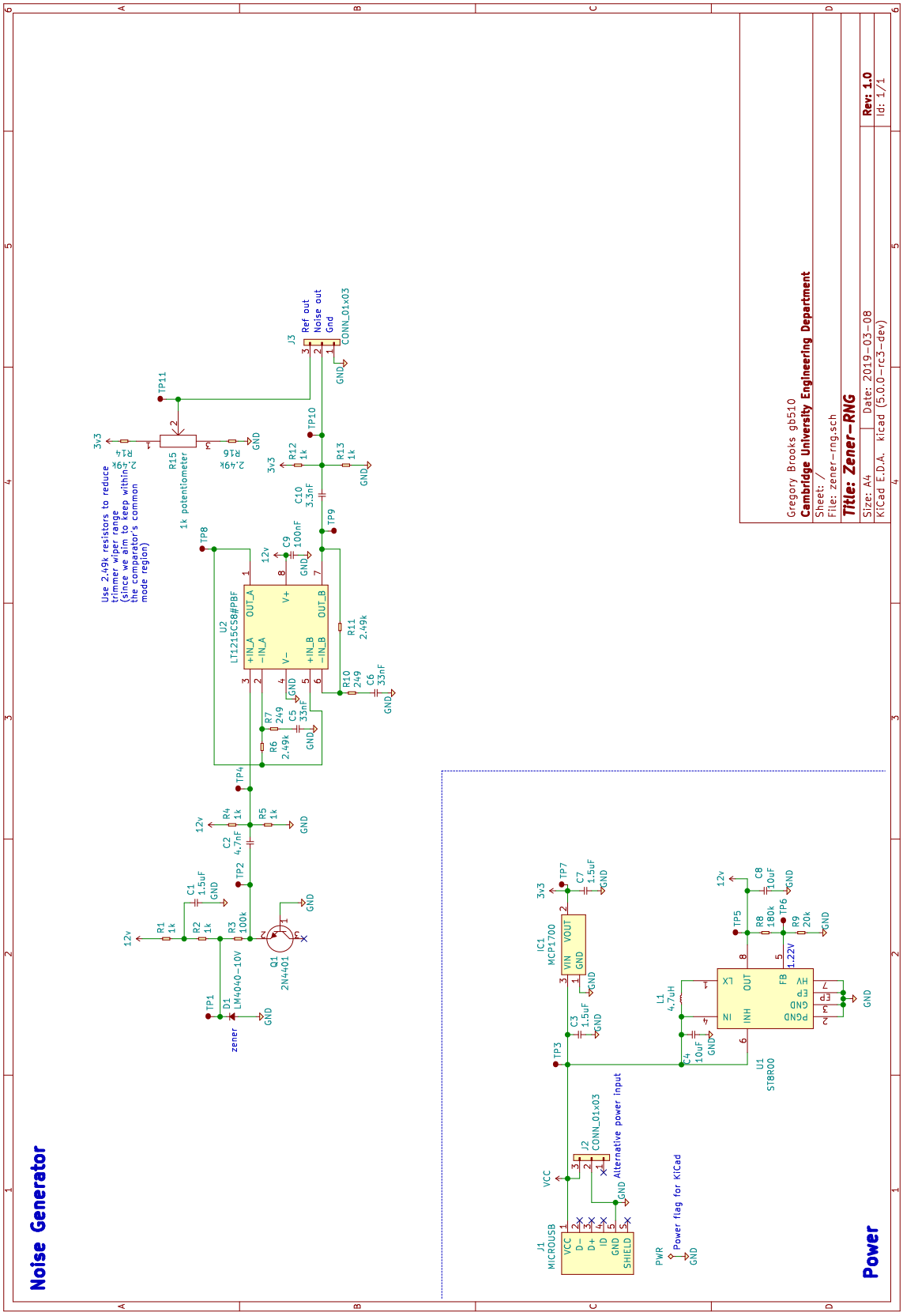
C Uniform Random Number Generator: Block Diagram

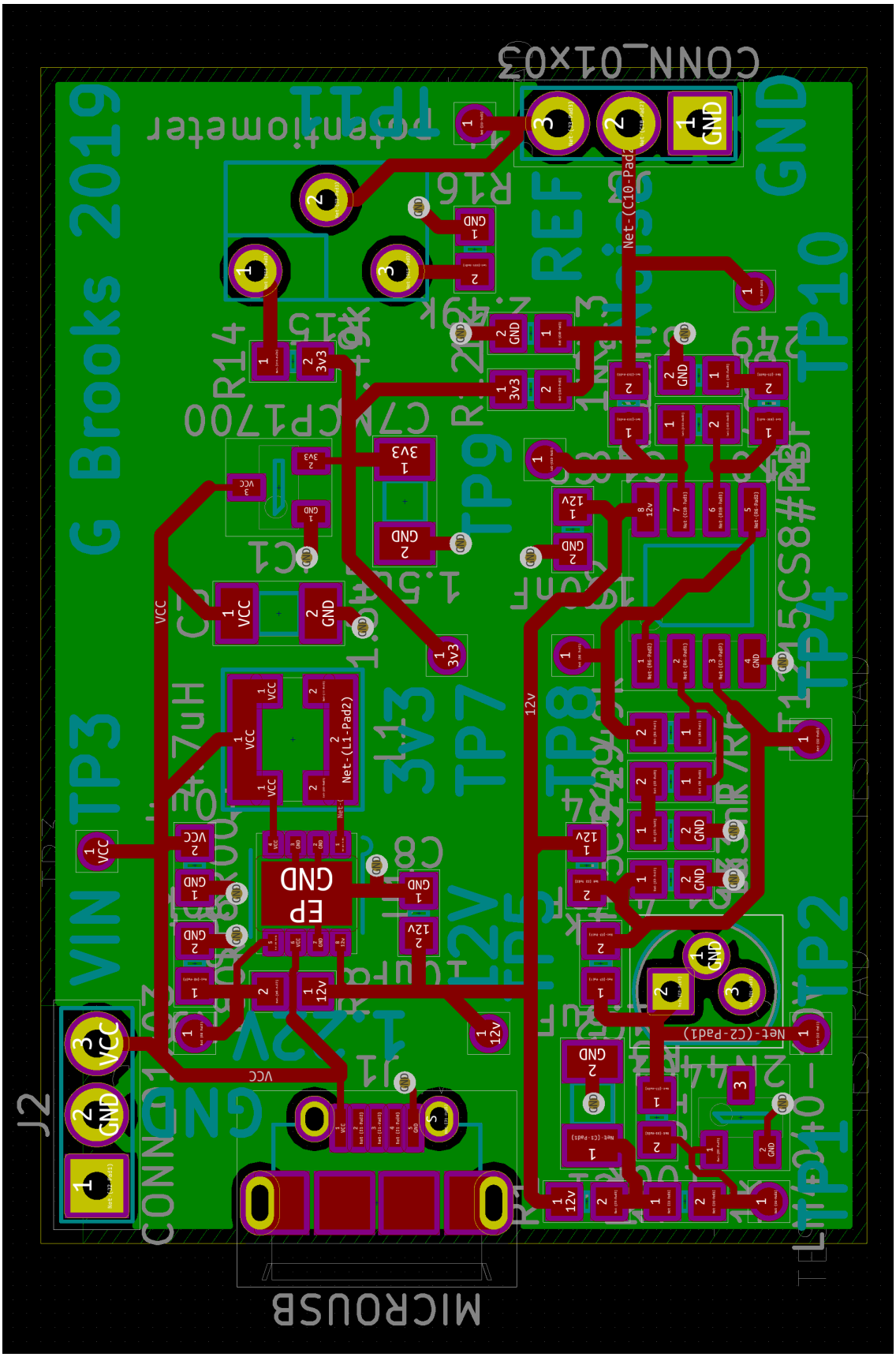
⟨ *TODO: insert block diagram*⟩

D Inversion Method Random Number Generator: Block Diagram

⟨ *TODO: insert block diagram*⟩

E Hardware Entropy Source Schematic and PCB




G Brooks 2019
VIN
TP3
TP9
TP7
TP8
TP4
TP10
TP2
GND
3V3
12V
VCC
EP
GND
J2
CONN_01X03
MICROUSB
Potentiometer
Net-(C10-Pad2)
Net-(C2-Pad1)
Net-(L1-Pad2)
Net-(C1-Pad1)

F EuroSys 2019 Poster: Safeguarding Sensor Device Drivers Using Physical Constraints

F.1 Description

This poster [**eurosys˙poster**] was submitted to EuroSys 2019, to communicate the idea of using information about a physical system to condition electronic sensor measurements. The example discussed in the poster is the detection and safeguarding against transduction attacks [**Fu˙2018**], where an attacker manipulates a sensor's output to gain control over a system. For example, a presentation [**autonomous˙vehicles**] illustrates how the proximity sensors on a Tesla vehicle can be fooled into providing erroneous (or even no) data using ultrasonic interference produced by a device built using off-the-shelf electronics. An attacker would not need to have access to the Tesla software (or even physical access to the hardware) in order to control the system's behaviour.

The poster illustrates how the Newton language can be used to describe a physical system, in this case an accelerometer mounted to a PCB without vibration isolation. In this scenario, vibrations of the PCB (e.g. due to a nearby loudspeaker or even, in the case of a smartphone, due to loudspeakers mounted to the PCB) are measured by the accelerometer, obscuring a *true* acceleration measurement. This effect is particularly pronounced if the board is driven at its resonant frequency, since the resulting oscillation will have a greater amplitude [**adi**]. This phenomenon could, in theory, be used as part of a transduction attack e.g. where a smartphone's loudspeaker is used to interfere with an application's estimate of the device's physical orientation.

To test whether this phenomenon can be distinguished from regular measurement noise, an experiment [**poster˙experiment**] was performed using an MMA8451Q accelerometer mounted on an FRDM-KL03Z board. With the sensor resting on a desk, five minutes worth of accelerometer samples were recorded at 10Hz in order to obtain a probability distribution for the accelerometer's measurement noise (along the z axis, aligned with the vertical); this data was empirically observed to fit a Laplace distribution. This measurement was then repeated with the board resting on top of a smartphone playing a 440Hz audio tone — in theory, this data (random samples from a sinusoid) would fit a bimodal beta distribution (derivation in section F.2) allowing a log likelihood ratio to be computed:

$$LLR = -2\sum_{i=1}^{N} \frac{\frac{1}{\pi}\left|\frac{1}{\sqrt{A^2\omega^4 - \ddot{x}_i^2}}\right|}{\frac{1}{2b}exp\left(-\frac{|\ddot{x}_i - \mu|}{b}\right)} \tag{10}$$

For the data collected in the experiment, the log-likelihood ratio was found to be around -24600.08; the negative value indicates that the measurements taken whilst the tone was playing were indeed more accurately described by the bimodal beta distribution, compared to the Laplace distribution of sensor noise. Computing this log-likelihood ratio could therefore be used as evidence to suggest whether an audio tone based transduction attack is in progress.

F.2 Derivation of Bimodal Beta Distribution

Let T be a uniformly distributed random variable representing the time at which acceleration is sampled:

$$T \sim U\left(-\frac{\pi}{\omega}, \frac{\pi}{\omega}\right)$$

$$f_T(t) = \begin{cases} \frac{\omega}{2\pi} & -\frac{\pi}{\omega} \leq t \leq \frac{\pi}{\omega} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$F_T(t) = \begin{cases} 0 & t < -\frac{\pi}{\omega} \\ \frac{\omega(t+\pi/\omega)}{2\pi} & -\frac{\pi}{\omega} \leq t \leq \frac{\pi}{\omega} \\ 1 & t > \frac{\pi}{\omega} \end{cases} \tag{12}$$

Then let X = g(T) represent the acceleration value at time T. By considering the dynamics of the system, we know that:

$$g(t) = A\omega^2 sin(\omega t) \tag{13}$$

where A is a frequency-dependent constant (equal to the product of quality factor and board displacement at resonance). We can also define a monotonically increasing function $h(x)$ as the inverse of $g(t)$:

$$h(x) = g^{-1}(x) = \frac{1}{\omega} arcsin\left(\frac{x}{A\omega^2}\right) \tag{14}$$

The cumulative distribution function for X can therefore be written in terms of the cumulative distribution function for T, accounting for the fact that g(t) is not monotonic:

$$F_X(x) = F_T(h(x)) + \Big(1 - F_T(\pi/\omega - h(x))\Big) \tag{15}$$

Taking the derivative results in the probability density function for X:

$$f_X(x) = \Big(f_T(h(x) - f_T(-h(x)))\Big)\frac{d(h(x))}{dx} \tag{16}$$

$$= \begin{cases} \frac{1}{\pi\sqrt{A^2\omega^4 - x^2}} & |x| \leq A\omega^2 \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

F.3 Poster

⟨ *TODO: insert the poster into this report?*⟩