Securing Digital Microfluidic Biochips by Randomizing Checkpoints

Jack Tang, Ramesh Karri
New York University

Mohamed Ibrahim, Krishnendu Chakrabarty
Duke University

***Abstract*—Much progress has been made in digital microfluidic biochips (DMFB), with a great body of literature addressing low-cost, high-performance, and reliable operation. Despite this progress, security of DMFBs has not been adequately addressed. We present an analysis of a DMFB system prone to malicious modification of routes and propose a DMFB defense based on spatio-temporal randomized checkpoints using CCD cameras. Absent the knowledge of the time- and space-randomized checkpoints, an attacker cannot navigate the DMFB without alerting the system. We present an algorithm to guide the placement and timing of the checkpoints such that the probability that an attack can evade detection is minimized. The efficacy of the defense mechanism is illustrated with a case study under stealthy malicious modifications.**

I. Introduction

Digital microfluidic biochips (DMFB) have emerged as a powerful technology to realize the lab-on-a-chip paradigm [1]. In contrast to microfluidic biochips based on continous flow valves [2], DMFBs process droplets in miniscule discrete quantities using a grid of electrodes which are driven by a sequence of actuation voltages from a computer [3]. Inspired by the advances in VLSI design automation, DMFB researchers have generated a large body of work addressing high-level synthesis [4], fault-tolerance [5], [6], error recovery [7], [8], chip testability [9], [10], and pin-count reduction [11].

The progress of DMFB research continues to mirror the evolution of VLSI CAD research at an accelerated pace, except for one key difference—DMFB platforms are still in their relative infancy while there is a growing awareness of the pressing need for hardware security [12]. This presents an opportunity to address security now, instead of waiting for the maturation of the technology to begin applying stopgap fixes. This paper presents the first-ever analysis and design of a security mechanism for a DMFB system under malicious modification. The contributions of this paper are:

1) A detection system for malicious DMFB modification, based on the concept of randomized *checkpoints*—the optical sampling of the DMFB at random locations and times.
2) Analysis of the probability of evasion for a malicious droplet under the proposed detection system.
3) An approximation algorithm to place static checkpoints at critical locations in order to lower the probability of evasion.
4) A case study to illustrate both how a typical assay can be modified, and simulation results showing how an attack can be detected using randomized checkpoints.

The structure of this paper is as follows: Section II gives an overview of DMFB technology. Section III discusses background information on hardware security and describes how a DMFB system may be attacked. Section IV presents a defense system based on randomized checkpoints. Section V presents simulation results and Section VI details the implications of this work while providing directions for future research.

II. Digital Microfluidic Biochips

DMFBs manipulate discrete picoliter volume droplets using the electrowetting-on-dielectric principle [3]. The electrowetting principle allows one to control the contact angle between a droplet and an electrode by adjusting the applied voltage. Applying a low-voltage to an electrode with a droplet and a high-voltage to an adjacent electrode causes the droplet to be moved to the high-voltage electrode. A 2D grid of electrodes provides a substrate upon which droplets can be transported. Fig. 1(a) illustrates the structure of a representative DMFB. The droplet is sandwiched between two plates, each of which contains a control electrode which is coated by a hydrophobic insulator. By applying a suitable sequence of control voltages on the electrodes, droplets of reagents or chemical samples can be dispensed from reservoirs, shuttled around the chip, mixed, separated, and placed in output ports. The functionality of the biochip is entirely determined by control software, which in turn drives the control voltages, termed *actuation sequences*. These operations are sufficient to implement a vast array of biological assays, such as immunoassays, protein crystallization, and DNA sequencing [14].

Recent research on DMFBs has investigated cyberphysical system integration [13]. These control systems utilize sensor feedback to correct for erroneous operation in real-time. The feedback allows for a robust system that is resilient to hardware faults as well as fluidic errors such as incomplete mixing or droplet volume variations. Fig. 1(b) illustrates the concept of a cyberphysical DMFB. A computer executes the control software which sends actuation sequences to the DMFB platform. A camera monitors the quality of the assay in real-time, and sends readings back to the computer for processing. The computer can use this information to re-synthesize actuation sequences in the event of a fault [7].

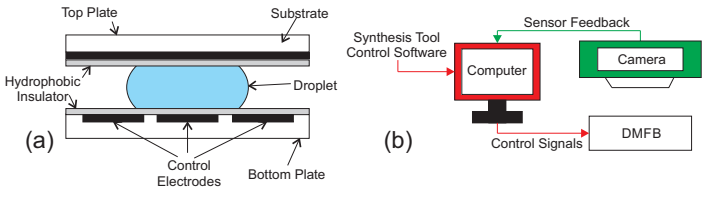

Fig. 1. (a) Structure of a DMFB array as viewed from the side. (b) Schematic diagram of a typical cyberphysical DMFB system [13].

978-1-4673-8773-6/16/$31.00

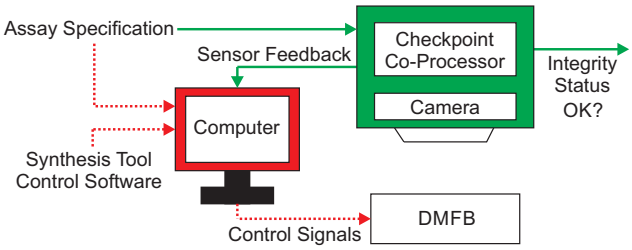

Fig. 2. DMFB secure co-processor implementation. Solid lines indicate signals assumed to be trustworthy while dotted lines are susceptible to attack.

A. CCD-based Error Recovery

Recent work on error recovery has proposed the use of charge-coupled device (CCD) cameras to provide sensor feedback. In contrast to alternative sensors such as ring oscillators and integrated optical detectors, CCD cameras provide the greatest amount of flexibility and localization since any electrode can be monitored by software [15], [16]. The reconfigurable nature of CCD imaging makes it an attractive sensor candidate for implementing a randomized defense system. However, it should be noted that CCD imaging requires more hardware overhead than alternative sensors and may not be usable for assays that utilize light-sensitive reagents [17].

CCD camera-based monitoring of DMFB systems is performed by capturing images of the entire array, and then running a pattern-matching algorithm to locate the position of the droplets. One such algorithm generates a correlation map for the entire array, selecting the electrodes with the highest correlations to locate the droplets [15]. Another technique focuses on specific areas by cropping the image on electrodes where droplets are expected to appear, and then performing the correlation algorithm against a template image [7].

For a DMFB system under attack, malicious droplets, which are not specified by the assay protocol, might be introduced by an attacker. We define a *checkpoint* as the act of comparing the status of an electrode against the state specified by a fully synthesized assay, in order to detect such droplets. To practically capture the DMFB state with an imaging operation, we utilize the cropping pattern-matching technique. The imaging system focuses its attention at specific electrodes at specific times to examine the presence, volume, and concentration of droplets. This technique is chosen over the highest-correlation method since without prior knowledge of how many droplets are present on the array, the system may produce false positives or negatives.

III. Attacks Against DMFBs

A. Related Work

It has been recently shown that DMFBs are susceptible to a variety of attacks. Alteration of high-level assay specifications and low-level actuation sequences can lead to denial-of-service (DoS), where assays are disabled entirely, or subtler attacks where the resulting errors are undetectable [18]. The threat of attacks on DMFBs is real, but to-date, no works have considered any specific security mechanisms to prevent, detect, or counter attacks. We thus outline some concepts and advances in VLSI hardware security in order to guide the discussion on DMFB security.

A *hardware trojan* is a malicious modification of a circuit which may cause unwanted behavior. Hardware trojans can be inserted at any level during the integrated circuit design flow, from high-level system design specification all the way down to the transistor level. The effects include, but are not limited to, leakage of sensitive information, denial-of-service, and alteration of the functionality of a device [19], [20]. The threat of hardware trojans is real, as there have been reported instances of trojan insertion [21], and the modern horizontal manufacturing design flow presents numerous avenues for a malicious adversary to exploit. Techniques to prevent, detect, and thwart hardware trojans include functional testing [22], delay-path fingerprinting [23], ring-oscillator characterization [24], input scrambling [25], and static verification [26]. Recent review papers provide more in-depth coverage [12], [27], [28].

It should be noted that the typical DMFB cyberphysical system implementation utilizes a computer, which is a presumably fabricated using conventional means. Therefore, the computer hardware is susceptible to trojan insertion. Defending against attacks at this level is outside the scope of this paper. We focus our attention on the DMFB itself, and attacks that can be considered a form of hardware trojan: malicious modification of the DMFB actuation sequences that results in unintended operation.

While design automation for VLSI and DMFBs share many similarities, the differences underlying the two technologies suggest that DMFB security techniques may take radically new forms. DMFBs utilize electrodes as a reconfigurable resource that can be used as either a processing element or a routing element, whereas integrated circuits utilize transistors for processing and wires for routing. Furthermore, the resources in a DMFB may be reconfigured during the execution of an assay. Even a reconfigurable technology like an FPGA remains static during execution. This extra degree of freedom contributes to the complexity of DMFB design. Additionally, the execution of a DMFB is easily observable with a camera whereas integrated circuits are opaque to all but the most determined parties. It is precisely these differences that will drive the design of the checkpoint system in Section IV.

B. Threat Model

We assume that a malicious adversary is able to modify the low-level actuation sequences of the DMFB hardware to an extent that she may purposefully mis-route droplets. This can be achieved through several ways. Computer hardware trojans may be present, as described previously. Other avenues include the control software, which may be compromised through a network connection originally integrated for software updates or convenient downloading of assay specifications [29]. Additionally, embedded systems meant to be deployed at the point-of-care are physically vulnerable to modification. The operator of the DMFB is presumed to be trustworthy, and that there is no tampering with the physical aspects of the system such as the loading of samples/reagents and the imaging system (i.e. we ignore camera spoofing through the *analog hole* [30]).

While this study assumes that the execution of control sequences is prone to attack, we do presume the integrity of the defense mechanism. This can be achieved by implementing a separate checkpoint co-processor (Fig. 2). The co-processor

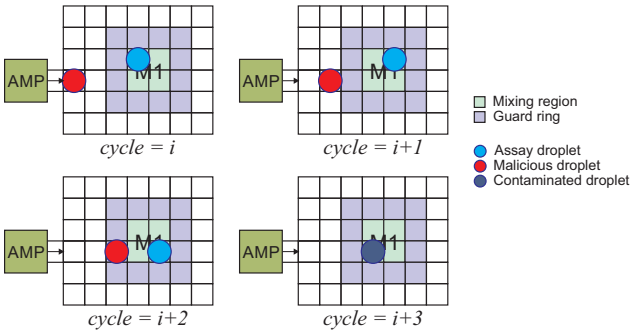

Fig. 3. Potential malicious route. Droplets can be dispensed to foul/contaminate a good droplet in a mix module. The target droplet concentrations can be altered to either cause failure of the assay (denial-of-service) or to report incorrect measurements.

should be isolated from the main controller, and tightly integrated with the camera sensor. The co-processor alerts the operator to any deviations from the assay specification. Note that the assay specification does need to be input separately to the two processors. This may cause some inconvenience, but is necessary to increase the attack surface.

Furthermore, we assume a general-purpose programmable DMFB system that is fully addressable. A general-purpose programmable DMFB system gives the most flexibility in terms of what types of assays may be executed, but at the same time it gives an attacker the most flexibility to modify the actuation sequences. We also assume that either error recovery is absent, or checkpoint-based error recovery [7] has been implemented wherein the attacker can predict the location of the checkpoints and hence can evade them.

One potential threat is illustrated in Fig. 3. This example shows the final execution cycles of a polymerase chain reaction (PCR) assay. The PCR assay is used in DNA amplification and has been studied in the DMFB literature. At clock cycle i, a malicious droplet has been dispensed from the AmpliTaq DNA polymerase port to be routed to mix module $M1$. Higher concentrations of AmpliTaq increase production of nonspecific products, lowering the quality of the assay output [31], [32], [33]. Either the DMFB error detection scheme will detect the wrong concentration at the output of this mix stage, or this altered droplet will be allowed to propagate through the assay if no error recovery is implemented. In either case, the result of the attack is either a denial-of-service, or output alteration/contamination.

The consequence of output contamination/alteration is that an assay result may be interpreted as accurately reflecting reality. This can be dangerous in cases where the assay is used to perform some measurement or test [34], for example, in-vitro glucose measurement. If a user's glucose measurement is inaccurate, the wrong dosage of insulin may be administered which could lead to overdose. The result of a DoS attack is that the DMFB is not able to perform its intended function, causing inconvenience while wasting samples, reagents and money. But more insidiously, a DoS attack, if not detected as a DoS attack, may trick error correction to believe that a hardware fault has occurred. Electrodes may be marked as faulty when they are still functional, causing the DMFB hardware to have reduced fault-tolerance and shorter operating lifespan.

C. DMFB Attack Modeling

All practical malicious modifications require the movement of droplets from a *source* to a *target* on the DMFB. Examples of sources include dispense ports, waste reservoirs, and backup reservoirs. Examples of targets include output ports, backup reservoirs, mix modules and droplets in transit. It is conceivable that an adversary could mount an attack that does not alter the result, such as dispensing extra reagents into unused electrodes, but the focus in this paper remains on attacks that change the assay result.

Hence, we model the class of attacks that can be formulated as a mis-routing problem between a source and a target. It should be noted that not all (source, target) combinations result in a meaningful attack. For example, routing a wash droplet into an output port would be easily detected as a fault since it bears no resemblance to the desired output droplet. This observation will save some time in analyzing and evaluating the proposed defense system in Section IV. Table I enumerates the typical resources in a DMFB platform and classifies them as potential sources or targets for a malicious droplet.

TABLE I. CLASSIFICATION OF RESOURCES.

Resource	Source	Target
Dispense Port	✓	
Output Port		✓
Waste Reservoir	✓	
Backup Reservoir	✓	✓
Mix Module	✓	✓
Droplets in Transit	✓	✓

IV. RANDOMIZED OPTICAL CHECKPOINTS

We propose a checkpoint-based detection system where CCD cameras are utilized to monitor the DMFB for malicious modification. If the system is able to monitor the progress of the assay as it is being executed, it would be able to determine if there is any deviation from the desired behavior. In the ideal case, the system would be able to monitor the entire DMFB array for every execution cycle. However, this is resource-intensive since the CCD camera must be focused at specific points and a pattern-matching algorithm must be executed for each electrode. While it is conceivable that a video recording of the assay execution could be used for offline forensic analysis, it is desirable to implement a real-time system so that the user can be immediately alerted to a security breach and to prevent waste of precious samples or reagents.

To make the defense system lightweight, we will make use of randomized checkpoints. The CCD camera system will monitor a random subset of the DMFB electrodes at random sampling times, and then compare the result to the desired actuation sequence. The system only needs to determine the state of an electrode as being occupied or not. Thus the design problem is to determine how many checkpoints to monitor, which electrodes to monitor, and how often to refresh the checkpoints. Fig. 4 illustrates this concept with an imaginary assay using two mix modules and a single malicious droplet. At the first cycle, five checkpoints are selected, but none of them intersect with the malicious droplet. The following three cycles do not check any electrodes. The fifth cycle randomly selects five checkpoints, and at the final cycle no electrodes are monitored, resulting in malicious modification of a droplet in the process of being mixed.

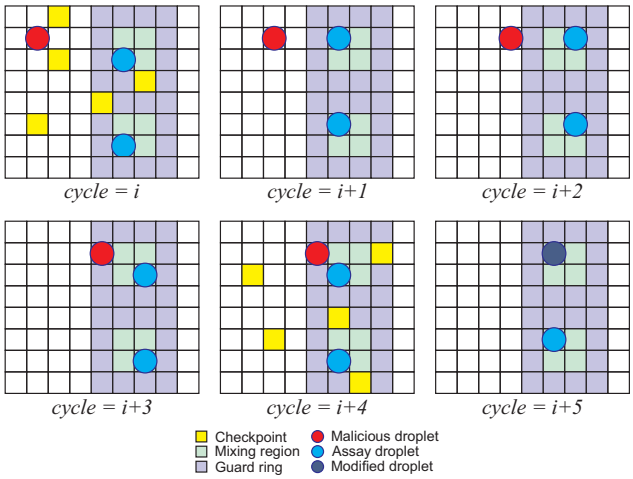

Fig. 4. Example assay execution with random checkpoints applied at cycle i and $i+4$. In this case, $s = 72$, $k = 5$, $L = 5$. This example is a realization of the random sampling times. Assuming $c = 0.33$, then $P(E) = 0.89$.

A. Probability of Evasion

We define the main metric of interest as the probability that a malicious droplet evades detection by the defense system. If the defense system can monitor the entire chip at every cycle, then clearly the probability of evasion is 0. If there are no checkpoints, the probability of evasion is 1.

Let E be the event that a malicious droplet evades detection for the lifetime of the droplet that executes over L total cycles. Note that L can be much less than the lifetime of the assay. E_i is the event that a malicious droplet evades detection for the i-th execution cycle, and F_i is the event that the i-th cycle is sampled and G_i is the event that the i-th cycle's checkpoints intersect with the malicious droplet. If each cycle's checkpoints are chosen independently, and the events F_i and G_i are independent, we have the evasion event as the event that the cycle is not sampled or the cycle is sampled and the set of checkpoints do not monitor the droplet.

$$P(E_i) = P(\overline{F_i} \cup (F_i \cap \overline{G_i})) = P(\overline{F_i}) + P(F_i) \cdot P(\overline{G_i}) \quad (1)$$

$$P(E) = P(E_1 \cap E_2 \cap \cdots \cap E_T) = \prod_{i=1}^{L} P(E_i) \quad (2)$$

The probability that a malicious droplet does not intersect with any checkpoints is simply the complement of the ratio of active checkpoints k at that time over the number of total electrodes s. Thus

$$P(\overline{G_i}) = 1 - \frac{k}{s} \quad (3)$$

The ratio k/s is called the *electrode coverage ratio*. Then we define the probability of sampling any execution cycle using some constant c as

$$P(F_i) = c \quad (4)$$

This constant is a design parameter that can be adjusted in software to achieve some performance criteria. Therefore, the probability of evasion can be expressed as

$$P(E) = \prod_{i=1}^{L} \left((1-c) + c\left(1 - \frac{k}{s}\right)\right) = \left(1 - \frac{ck}{s}\right)^{L} \quad (5)$$

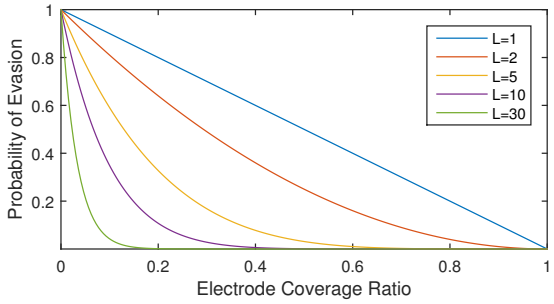

Fig. 5. The probability of evasion decreases as the checkpoint coverage ratio, k/s, increases. Here $c = 1$, meaning checkpoints are monitored on every execution cycle. Increasing L as a parameter shows exponential advantage, illustrating the intuitive notion that the longer a malicious droplet exists on an assay the more likely it is to be detected.

The parameter s is determined by the size of the DMFB array and can be considered to be constant. The parameters c and k should be maximized in order to minimize the likelihood of evasion, subject to the computational and imaging capabilities of the DMFB platform. The lifetime of the malicious droplet is clearly not under the user's control, but probability of evasion becomes exponentially smaller with L, as shown in Fig. 5. This key observation leads to the possibility of decreasing $P(E)$ through influencing the routability of malicious droplets. One possibility for influencing malicious routes is through the judicious placement of static checkpoints.

B. Static Checkpoint Placement

We next propose the introduction of a set of static checkpoints whose purpose is to force malicious droplets to take a more circuitous route. Either the adversary will be unaware of these checkpoints and will route through them, with high probability of being detected, or they will be aware of the static checkpoints and route around them. By causing the malicious droplet to take a longer, more circuitous route, we improve the odds that the randomized checkpoints will triggered. Assuming that a particular DMFB platform can handle a fixed number of checkpoints whether random or static, the problem is now to determine the ratio of static to random checkpoints, as well as the location of the static checkpoints which minimize the probability of evasion.

The placement of static checkpoints must be tailored specifically for any given assay. The effectiveness of the static checkpoint placement will be a function of how the assay was synthesized, and whether any resources were dedicated to error recovery. Note that if the malicious route is known *a priori*, it is trivial to place a static checkpoint to detect it. For an $m \times n$ array with k random checkpoints and q static checkpoints, there are

$$\sum_{q=0}^{m \times n - k} \binom{m \times n}{q} \quad (6)$$

different arrangements of static checkpoints. Even for a small 15×13 array limiting the number of static checkpoints to at most 10, this number is greater than 2^{54}. It is clearly impractical to evaluate all possible arrangements of static checkpoints to determine the optimal placement.

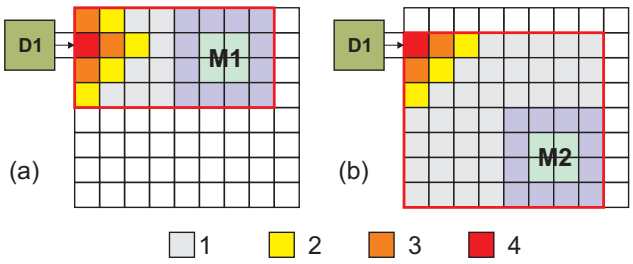

Fig. 6. (a) Rectangle used to approximate usefulness of electrodes in harboring a malicious route for dispense port 1 and mix module 1. (b) Rectangle for approximation between dispense port 1 and mix module 2.

Instead, we propose an algorithm to approximately choose an arrangement of static checkpoints. The idea is to rank each electrode on the DMFB array in terms of how useful it is to an attacker for routing a malicious droplet. Given this ranking, we can target the most useful electrodes, forcing malicious droplets to take a circuitous route. The number of static checkpoints can be increased easily, and evaluated for probability of evasion until a balance is made between static and random checkpoints.

The ranking of electrodes can be approximated by enumerating all useful combinations of sources and sinks, forming a rectangle with the farthest edges of these resources, and filling a matrix representation of the DMFB array with ones matching the placement of this rectangle. We call this matrix the $electrodeWeight$ matrix. Electrodes close to the source are given higher weight to satisfy the intuitive notion that it's easier to monitor a problem at the source. Fig. 6 illustrates the concept, considering only a dispense port as a source and two mix modules as the destinations. The corresponding matrices are

$$electrodeWeight_{1,1} = \begin{bmatrix} 3 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 4 & 3 & 2 & 1 & 1 & 1 & 1 & 1 & 0 \\ 3 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

$$electrodeWeight_{1,2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 3 & 2 & 1 & 1 & 1 & 1 & 1 & 0 \\ 3 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{8}$$

The weighting for assay as a whole, denoted as *arrayWeight*, is calculated by adding each combination of source and sink electrode weights as follows

$$\begin{aligned} arrayWeight &= \sum_i \sum_j electrodeWeight(source(i), target(j)) \\ &= electrodeWeight_{1,1} + electrodeWeight_{1,2} \end{aligned} \tag{9}$$

1: $arrayWeight \leftarrow 0$
2: **for** each connection between source and destination, where the destination exists at a time later than the source **do**
3: $electrodeWeight \leftarrow$ weighting rectangle enclosed by the farthest coordinates of both source and destination
4: $arrayWeight \leftarrow arrayWeight + electrodeWeight$
5: **end for**
return $arrayWeight$

Fig. 7. Electrode ranking pseudocode.

$$= \begin{bmatrix} 3 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 8 & 6 & 4 & 2 & 2 & 2 & 2 & 2 & 0 \\ 6 & 4 & 2 & 2 & 2 & 2 & 2 & 2 & 0 \\ 4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{10}$$

This example leads to some important insights. For simple assays and architectures, the best place to insert a static checkpoint is directly in front of the source of the malicious droplet. In larger arrays with more complicated assays that feature storage, backup and waste reservoirs, the answer may not be so clear. In cases where the resultant $arrayWeight$ matrix contains several entries of the same highest rank, the static checkpoint placer should select one of the top electrodes randomly. And in general, it may be difficult to force a malicious route to be redirected without having to apply a large number of static checkpoints. The electrode weighting approach is summarized in Fig. 7.

C. Temporal Randomization of Static Checkpoints

A static checkpoint is only useful if a malicious droplet is guaranteed to be caught while crossing it. If the DMFB platform, either through hardware limitations or choice, does not monitor the static checkpoint at every cycle, this increases the probability of evasion. Let v be the probability that any static checkpoint is monitored, q be the number of static checkpoints, and Q be the number of static checkpoints on the malicious route. Then the probability of evasion can be modeled as

$$P(E) = (1-v)^Q \left(1 - \frac{ck}{s-q}\right)^{L-Q} \tag{11}$$

That is, if the static checkpoints are monitored 100% of the time, the probability of evasion is exactly zero, unless the malicious route does not cross any static checkpoints ($Q = 0$ leads to Eq. 5). Adding temporal randomness to a spatially static checkpoint presents a trade-off for probability of evasion. In order for temporal randomness to provide any benefit, the following inequality must hold:

$$(1-v) \leq \left(1 - \frac{ck}{s-q}\right) \tag{12}$$

Rearranging, we find

$$v \geq c\left(\frac{k}{s-q}\right) \tag{13}$$

which can be interpreted as a tuning requirement. Recall that v and c are constants to be tuned by the system designer. Since

the $k/(s-q)$ term is less than or equal to 1, v can be less than c while still lowering $P(E)$. Therefore, static checkpoints provide a net gain: they potentially induce a malicious route to become longer, and for the same level of security, they take less resources to implement than a randomized checkpoint.

D. Security of Checkpoint-Based Error Recovery

The formulation presented previously for static and random checkpoints can be generalized to interpret a checkpoint-based error-recovery scheme. Thus, the security provided by the error-recovery system can be evaluated. First we introduce a generalized notion of a checkpoint. A general checkpoint is represented as an ordered 7-tuple

$$\begin{aligned} C_i = <& x(i), y(i), t(i), \\ & vol_{low}(i), vol_{high}(i), \\ & conc_{low}(i), conc_{high}(i) > \end{aligned} \tag{14}$$

where x and y are the coordinates that the detection takes place, t is the actuation cycle that detection occurs, $vol_{low}(i)$ and $vol_{high}(i)$ define a valid interval of droplet volumes, and $conc_{low}(i), conc_{high}(i)$ define a valid interval of concentrations. These interval specifications can be set to *don't-care* values simply by setting the low value to 0, and the high value to largest value perceptible by the imaging system.

Then we define a checkpoint arrangement M as a set of k randomized checkpoints

$$M(t) = \{C_1, C_2, ...C_k\} \tag{15}$$

where each x, y coordinate is chosen uniformly from the possible electrodes of the DMFB array, without replacement. These checkpoints are all active at the same cycle t, testing only for presence of a droplet by determining if the volume is below a threshold (absence) or above a threshold (presence). Concentration is set to *don't-care*. Finally, a randomized checkpoint system C_{rand} can be defined as a set of arrangements M, where each cycle t is selected by performing a *Bernoulli(c)* trial for each cycle of the assay execution. A checkpoint arrangement is added to the set for each success.

Using this formulation, it can be seen that the proposed security mechanism provides some level of error detection while error detection provides some measure of security. They both are able to determine when the behavior of a DMFB system deviates from the specification. The difference lies in the error detection mechanism, and subsequently, the attribution to an underlying fault. For error recovery, in general, it is difficult to infer that a malicious adversary was the cause since faulty hardware can generate the same result. On the other hand, in checkpoints used for detecting a malicious adversary, faulty hardware can lead to false positives. For instance, when a droplet gets stuck and remains on an electrode during cycles where there should be no electrode present.

In short, the only way to reconcile these differences is to interpret any kind of failure by any type of checkpoint as being a generalized fault. Answering the question of whether the fault was caused by hardware failure or malicious modification must be evaluated using studies on platform reliability in conjunction with a security audit of the operating environment (e.g. network connectivity, access control protocols, location).

V. Experimental Results

To evaluate the proposed system, we study the PCR assay under a DoS attack. The goal of the attacker is to route a KCl droplet from the dispense port to mix module M4 (Fig. 8), as excess KCl concentration inhibits PCR [35]. The target module M4 is bound by the synthesis tool to mix KCl with Tris-HCl. We compare the efficacy of the defense system under different design parameters, with and without static checkpoints, and demonstrate the validity of our analysis while showing that a sufficient level of security is achieved.

We assume that our DMFB defense system is capable of monitoring the progression of the assay execution every cycle. However, we also assume the checkpoint co-processor is only able to image 20 checkpoints at a time (i.e. $k+q=20$) to impose a performance constraint. The array under consideration has dimensions of 15×13, thus the electrode coverage ratio cannot be higher than 10.3%. We investigate the system performance with $s=0.5$ and $v=0.5$ to decrease how often the checkpoints are examined—in practice there may be ancillary reasons for not checking the DMFB state every cycle, such as reducing dynamic power consumption.

The following attack is simulated using an open-source DMFB synthesis tool [36], [37], extended to incorporate randomized checkpoints. The attacker is simulated using the Lee routing algorithm [38], which guarantees minimum-distance. Since minimizing the malicious route decreases the probability of evasion, a minimum-distance router provides an optimal strategy for the attacker. The success of the randomized checkpoint algorithm is validated using Monte Carlo analysis, with the end result being success or failure depending on whether the system detects the attack or not, respectively. The ratio of success to total number of attempted trials gives an estimate of the probability of detection, which is the complement of the probability of evasion.

A. Random Checkpoints Only

The route taken by our malicious software router is shown in Fig. 8(a). It takes 8 execution cycles to reach its destination, which is minimal since it is equal to the Manhattan distance. Note that it is possible to route the droplet through mix module 1 if the timing is chosen carefully; there is a small window of execution where the droplet being mixed in M1 is too far to be affected by the malicious droplet. Fig. 9 illustrates how $P(E)$ varies as a function of the electrode coverage ratio for the given route. With the given constraint of $k=20$, the electrode coverage ratio is 10.3%. This route yields $P(E)=0.66$.

B. Random and Static Checkpoints

We assume the adversary is able to learn about the static checkpoints, and thus is able to route around them. We fix $q=16$ and use the ranking algorithm in Section IV-B to place the static checkpoints. The result is that all the dispense ports are covered. With q set and total checkpoints limited to 20, $k=4$. With $v=0.5$, the inequality in Eq. 13 is satisfied, so the optimal strategy for the adversary is to route around the static checkpoints if possible. Fig. 8(b) illustrates the placement of the top 16 static checkpoints and the path chosen by the router. Note that the malicious route is forced to cross one static mine ($Q=1$), and then takes 9 cycles to reach the target ($L=9$).

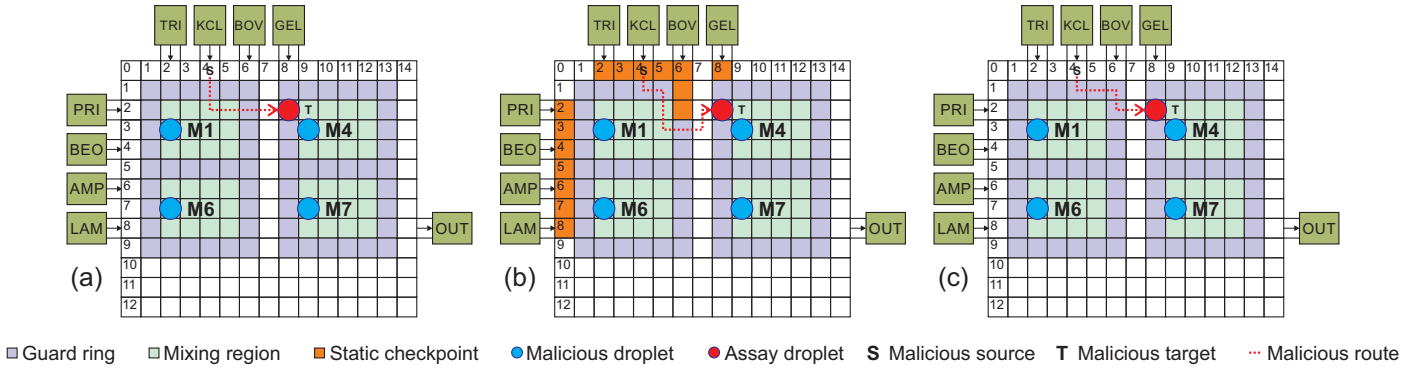

Fig. 8. (a) Random checkpoints only. Malicious route from KCl port targets M4 mix module. No obstacles mean that the adversary is free to route with minimal distance, minimizing probability of evasion. (b) Random and static checkpoints. The minimum Manhattan distance is no longer achievable, decreasing the probability of evasion. (c) Random checkpoints with error recovery. Malicious route is redirected to avoid mixing region M1 due to error recovery checkpoints. While the original route is no longer achievable, there still exists a minimal path from the source to target.

Fig. 9 shows $P(E)$ for this longer route as being lower than the original route for all electrode coverage ratios. Blocking off the dispense port provides a tremendous advantage. In this case, the electrode coverage ratio is 2.2% and $P(E) = 0.45$. Note that if v had been set to 1, this attack would have been detected immediately.

C. Random Checkpoints with Error Recovery Checkpoints

Now we investigate how error-recovery checkpoints interact with the system. A malicious route attempting to cross through an error-recovery checkpoint would immediately trigger the error recovery process. Assuming the attacker's goal has not changed, the route must now move around the region defined by M1, which is being actively monitored (Fig. 8(c)). The malicious droplet joins the droplet being mixed in M4. This is detectable depending on the how the error recovery system is setup. If the error thresholds are not set properly, changing the concentration of KCl this way is conceivable. The route length is the same as the case in Fig. 3. Thus the error recovery mechanism provides no advantage in terms of detecting malicious droplets in transit, and $P(E)$ is exactly the same as in the case without error recovery.

D. Discussion

The probability of evasion achieved in the PCR case study provides a strong disincentive for would-be attackers. A probability of evasion equal to a fair coin flip is easy to obtain and can be achieved even if the electrode coverage ratio is less than 10% (Fig. 9), using both random and static checkpoints being monitored with 50% probability each cycle. Forcing the random checkpoints to be monitored on every cycle causes the probability of evasion curves to drop (Fig. 10). Given that typical DMFB arrays operate at relatively slow speeds, with actuation frequencies in the hundreds of hertz, such performance may be feasible to achieve. The situation only gets worse for the attacker the longer an attack takes place, as the lifetime L gives exponentially lower probability of evasion. Even if we consider other types of attacks where droplets are routed between modules where L is lower, and consequently, $P(E)$ is higher, evaluation of probabilistic models likely underestimates the real-world effect of implementing such a system. The fact that a randomized checkpoint system exists is a deterrent to any would-be attackers.

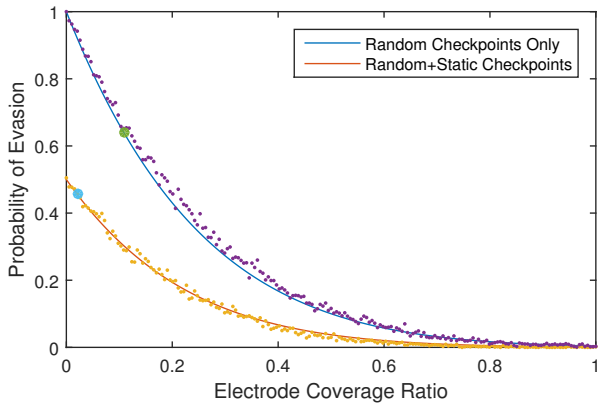

Fig. 9. Probability of evasion vs. electrode coverage ratio for a minimal length route and a route with static checkpoints attempting to dispense KCl into M4. Probability that a given cycle is monitored was set to 50% for both random and static checkpoints. Solid lines are analytic results, dotted lines are simulation data. The two large dots indicate data points from the PCR case study, where the number of checkpoints is limited to 20.

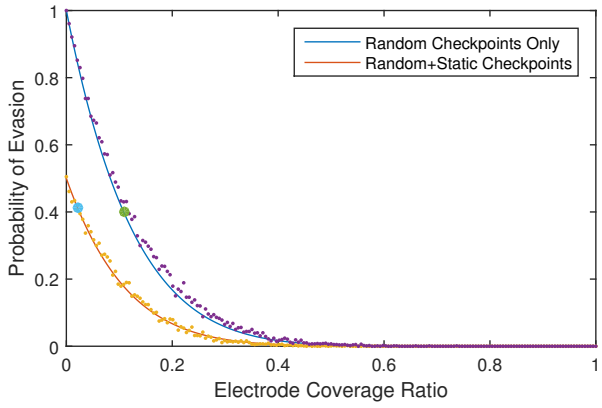

Fig. 10. Performance improves when the probability of monitoring random checkpoints at each cycle is increased to 100%. Probability of monitoring static checkpoints is 50%. The two large dots indicate data points from the PCR case study, where the number of checkpoints is limited to 20. $P(E)$ drops to 0.40 at 10.3% coverage ratio for random checkpoints only.

VI. Conclusion

The implications of compromising the integrity of an assay go beyond productivity and financial losses, as human health is now at stake. False information can drive incorrect medical decisions resulting in poor outcomes or even death. A malicious party thus faces the possibility of having their attack detected, traced back to the source, and facing charges of fraud or even manslaughter. An adversary requires that their attack is undetectable with near absolute certainty. The proposed randomized checkpoint detection system provides a high level of assurance that the adversary cannot achieve its goal of modifying an assay without being detected.

There is much more work to be done to make DMFB systems secure against all forms of attack. This cyberphysical system is but one approach, and it has its limitations. For instance, it would be better to find provably optimal solutions for the placement of static checkpoints instead of using the heuristic placement algorithm suggested in this work. Future work could address other weaknesses in the DMFB platform, such as at the high-level synthesis level or at an even lower hardware level, by developing sensors or pin-mappers that can provide assurances of authenticity and integrity.

The evolution of DMFB design continues to take its cues from VLSI design, with security being the latest continuation of this trend. However, it is important to note what sets DMFBs apart from VLSI in order to secure them properly—the scale, the way they are manufactured, the component integration for constructing the cyberphysical systems, even the fact that movement of droplets is visible with the naked eye. These differences should be investigated to discover more creative security solutions. Securing DMFBs is an important task, and it is hoped that enough work will be done early on to make this technology trustworthy, promoting widespread adoption.

References

[1] R. B. Fair, "Digital microfluidics: is a true lab-on-a-chip possible?" *Microfluidics and Nanofluidics*, vol. 3, no. 3, pp. 245–281, 2007.

[2] T. Thorsen, S. J. Maerkl, and S. R. Quake, "Microfluidic large-scale integration," *Science*, vol. 298, no. 5593, pp. 580–584, 2002.

[3] M. Pollack, A. Shenderov, and R. Fair, "Electrowetting-based actuation of droplets for integrated microfluidics," *Lab on a Chip*, vol. 2, no. 2, pp. 96–101, 2002.

[4] F. Su and K. Chakrabarty, "High-level synthesis of digital microfluidic biochips," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 3, no. 4, p. 1, 2008.

[5] ——, "Module placement for fault-tolerant microfluidics-based biochips," *ACM Transactions on Design Automation of Electronic Systems*, vol. 11, no. 3, pp. 682–710, 2006.

[6] P. Pop, M. Alistar, E. Stuart, and J. Madsen, *Design Methodology for Digital Microfluidic Biochips*. Springer, 2016.

[7] Y. Luo, K. Chakrabarty, and T.-Y. Ho, "Error recovery in cyberphysical digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design*, vol. 32, no. 1, pp. 59–72, 2013.

[8] K. Hu, M. Ibrahim, L. Chen, Z. Li, K. Chakrabarty, and R. Fair, "Experimental demonstration of error recovery in an integrated cyberphysical digital-microfluidic platform," in *Proc. IEEE Biomed. Circuits and Syst. Conf.*, 2015, pp. 1–4.

[9] T. Xu and K. Chakrabarty, "Parallel scan-like test and multiple-defect diagnosis for digital microfluidic biochips," *IEEE Trans. Biomed. Circuits Syst.*, vol. 1, no. 2, pp. 148–158, 2007.

[10] T. A. Dinh, S. Yamashita, T.-Y. Ho, and K. Chakrabarty, "A general testing method for digital microfluidic biochips under physical constraints," in *Proc. IEEE Int. Test Conf.*, 2015, pp. 1–8.

[11] C. C.-Y. Lin and Y.-W. Chang, "ILP-based pin-count aware design methodology for microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 9, pp. 1315–1327, 2010.

[12] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug. 2014.

[13] Y. Luo, K. Chakrabarty, and T.-Y. Ho, *Hardware/Software Co-Design and Optimization for Cyberphysical Integration in Digital Microfluidic Biochips*. Springer, 2014.

[14] H.-H. Shen, S.-K. Fan, C.-J. Kim, and D.-J. Yao, "EWOD microfluidic systems for biomedical applications," *Microfluidics and Nanofluidics*, vol. 16, no. 5, pp. 965–987, 2014.

[15] Y.-J. Shin *et al.*, "Machine vision for digital microfluidics," *Review of Scientific Instruments*, vol. 81, no. 1, p. 014302, 2010.

[16] C.-L. Sotiropoulou, L. Voudouris, C. Gentsos, A. M. Demiris, N. Vassiliadis, and S. Nikolaidis, "Real-time machine vision FPGA implementation for microfluidic monitoring on lab-on-chips," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 2, pp. 268–277, 2014.

[17] D. Witters, K. Knez, F. Ceyssens, R. Puers, and J. Lammertyn, "Digital microfluidics-enabled single-molecule detection by printing and sealing single magnetic beads in femtoliter droplets," *Lab on a Chip*, vol. 13, no. 11, pp. 2047–2054, 2013.

[18] S. Subidh Ali, M. Ibrahim, O. Sinanoglu, K. Chakrabarty, and R. Karri, "Security assessment of cyberphysical digital microfluidic biochips," *IEEE/ACM Transactions Computational Biology and Bioinformatics*, vol. 13, no. 3, pp. 1–14, 2015.

[19] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39–46, Oct. 2010.

[20] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, Jan. 2010.

[21] S. Skorobogatov and C. Woods, *Breakthrough silicon scanning discovers backdoor in military chip*. Springer, 2012.

[22] F. Koushanfar and A. Mirhoseini, "A unified framework for multimodal submodular integrated circuits trojan detection," *IEEE Trans. Inf. Forens. Security*, vol. 6, no. 1, pp. 162–174, Mar. 2011.

[23] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *IEEE Int. Workshop on Hardware-Oriented Security and Trust*, Jun. 2008, pp. 51–57.

[24] J. Rajendran, V. Jyothi, O. Sinanoglu, and R. Karri, "Design and analysis of ring oscillator based design-for-trust technique," in *Proc. 29th VLSI Test Symp.*, May 2011, pp. 105–110.

[25] A. Waksman and S. Sethumadhavan, "Silencing hardware backdoors," in *Proc. IEEE Symp. Security and Privacy*, May 2011, pp. 49–63.

[26] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith, "Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically," in *Proc. IEEE Symp. Security and Privacy*, May 2010, pp. 159–172.

[27] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: Threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, Aug. 2014.

[28] J. Rajendran, O. Sinanoglu, and R. Karri, "Regaining trust in VLSI design: Design-for-trust techniques," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1266–1282, Aug. 2014.

[29] P. A. Williams and A. J. Woodward, "Cybersecurity vulnerabilities in medical devices: a complex environment and multifaceted problem," *Medical Devices: Evidence and Research*, vol. 8, p. 305, Jul. 2015.

[30] E. Diehl and T. Furon, "© watermark: Closing the analog hole," in *Proc. IEEE Conf. Consum. Electron.*, Jun. 2003, pp. 52–53.

[31] Z. Hua, J. L. Rouse, A. E. Eckhardt, V. Srinivasan, V. K. Pamula, W. A. Schell, J. L. Benton, T. G. Mitchell, and M. G. Pollack, "Multiplexed real-time polymerase chain reaction on a digital microfluidic platform," *Analytical chemistry*, vol. 82, no. 6, pp. 2310–2316, 2010.

[32] V. Srinivasan, "A digital microfluidic lab on a chip for clinical diagnostic applications," Ph.D. dissertation, Duke University, 2005.

[33] S. Kennedy, *PCR troubleshooting and optimization: The essential guide*. Horizon Scientific Press, 2011.

[34] T. W. S. Journal. (2016, Mar.) Theranos Results Could Throw Off Medical Decisions, Study Finds. [Online]. Available: http://www.wsj.com/articles/theranos-results-could-throw-off-medical-decisions-study-finds-1459196177?mod=e2fb

[35] R. Higuchi, C. Fockler, G. Dollinger, and R. Watson, "Kinetic pcr analysis: real-time monitoring of dna amplification reactions," *Biotechnology*, vol. 11, pp. 1026–1030, 1993.

[36] D. Grissom, K. O'Neal, B. Preciado, H. Patel, R. Doherty, N. Liao, and P. Brisk, "A digital microfluidic biochip synthesis framework," in *Proc. IEEE/IFIP Int. Conf. VLSI and System-on-Chip*, Oct. 2012, pp. 177–182.

[37] D. Grissom and P. Brisk, "A field-programmable pin-constrained digital microfluidic biochip," in *Proc. IEEE/ACM Design Automation Conf.*, 2013, p. 46.

[38] C. Y. Lee, "An algorithm for path connections and its applications," *IEEE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 346–365, Sep. 1961.