2017 IEEE 35th International Conference on Computer Design

Security Trade-offs in Microfluidic Routing Fabrics

Jack Tang[1], Mohamed Ibrahim[2], Krishnendu Chakrabarty[3], Ramesh Karri[4]
Department of Electrical & Computer Engineering, New York University, Brooklyn, NY, USA[1, 4]
Department of Electrical & Computer Engineering, Duke University, Durham, NC, USA[2, 3]
jtang@nyu.edu[1], mohamed.s.ibrahim@duke.edu[2], krish@duke.edu[3], rkarri@nyu.edu[4]

***Abstract*—Microfluidic routing fabrics, or crossbars, based on transposer primitives provide benefits in manufacturability, performance, and on-the-fly reconfigurability. Many applications in microfluidics, such as DNA barcoding for single-cell analysis, are expected to benefit from these new devices. However, the control of these critical devices poses new security questions that may impact the functional integrity of a microbiology application. This paper explores the many security implications of microfluidic crossbars that directly result from their structure, programmability and use in critical applications. We analyze security performance using new metrics describing how fluids can be "scattered" to incorrect locations under fault-injection attacks, and from these derive a probability model describing the likelihood of a successful attack. We present a case study of a recently described routing fabric proposed for use in a hybrid DNA barcoding platform, and discuss how fabric designers can improve security through architectural choices.**

I. Introduction

Microfluidics—the study of the manipulation of minute quantities of fluids—reduces sample and reagent consumption, increases throughput, and streamlines usage in a number of applications such as DNA processing [1], [2]. After the initial development in basic physics, devices, and protocols starting in the 1980s, research soon turned to computer-aided design. This lead to breakthroughs in microfluidic protocol and architectural synthesis [3], [4], [5], abstracting away many of the low-level details so that end users could focus on the science instead of the fine details of manually generating control sequences. Microfluidics now finds itself rapidly maturing in an age where security is a top design priority for any emerging technology. To that end, this paper analyzes the security of a recently introduced microfluidic hardware primitive: the transposer [6]. This primitive was developed with the intent of introducing reconfigurability to microfluidics much in the same way that field-programmable gate arrays (FPGAs) did for integrated circuits. Already, transposer-based routing fabrics are being used in sample barcoding platforms for single-cell analysis [7].

The microfluidic routing fabric introduces reconfigurability into a system. When used as part of a cyberphysical system deployed in distributed research settings [8], the system can be susceptible to attack [9]. Rogue researchers may attempt to tamper with laboratory equipment to fabricate results. Unfortunately, this motivation is not speculative; studies have shown that the majority of retractions in the scientific literature are due to misconduct [10]. While the research community is often able to identify fraudulent data, a recent publication in *The FASEB Journal* [11] noted that *"The more the research community responds after the fact to incidents that diminish trust, the more it leaves to chance the public's support for its work."* Additionally, corporations or nation-states may be interested in market manipulation attacks, where the reliability and trustworthiness of the microfluidic platform is called into question. The motivations for attacking microfluidic platforms are real, and microfluidic system designers would do well to adopt a preventative mindset.

The broader security implications of microfluidics are only starting to be understood. Digital microfluidics, a type of microfluidic device based on the manipulation of discrete droplet quantities [2], has been analyzed for novel security threats [12]; malicious parties can attack the control systems responsible for driving microfluidic platforms, leading to subtle manipulation of fluid properties or outright premature failure. In devices designed for diagnostic care, such attacks can mislead physicians into incorrect diagnoses. Result manipulation in other mission-critical applications such as DNA forensics [13] or environmental monitoring [14] would also have disastrous consequences. And in many settings, attacks could destroy samples that are expensive or difficult to obtain.

Our premise is that for simple reconfigurable devices, such as the microfluidic routing fabric, the relationship between security and architectural choices can be satisfactorily quantified so that a system designer can decide what trade-offs to make. The key contribution of this paper is the definition of these security concepts, and elucidating the idea using microfluidic architectures culled from the research literature. To the best of our knowledge, this is the first attempt to identify security issues in a microfluidic technology outside of digital microfluidic biochips (DMFBs).

The rest of this paper is organized as follows. In Section II, we present an overview of microfluidic routing fabrics, their notation, and their use for critical microbiology applications. Section III discusses the security threat model. We introduce notation, security metrics, and perform a security analysis in Section IV. In Section V we study a routing fabric used in DNA barcoding applications and see how alternate architectures can impact security. Section VII concludes the paper.

II. Background

We motivate this work by exploring how security vulnerabilities can arise in important microfluidic applications when novel hardware is introduced.

A. Microfluidics-enabled Single-Cell Analysis

Cellular analysis is a widely used procedure in clinical diagnostics, pharmaceutical research, and forensic science [15]. With evidence that cells, even within the same clonal population, are heterogeneous in their genomic responses, a large number of single-cell analysis methods have been established using microfluidic devices [16]. Single-cell analysis relies on encapsulation of individual cells inside droplets and tagging these droplets with unique DNA barcodes; this procedure is referred to as DNA barcoding [17]. Barcoded samples can

This research is supported in part by ARO grant number W911NF-17-1-0320.

1063-6404/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCD.2017.13

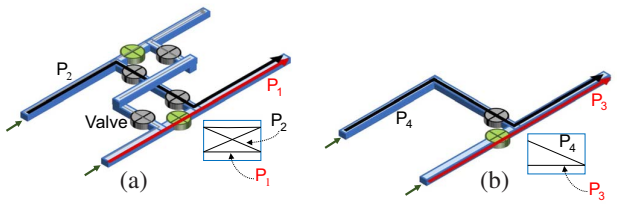

Fig. 1. A schematic view of transposers [7]: (a) a 2-to-2 transposer primitive. Route P_1 enables the valves in green and disables the valves in grey, causing both fluids to be passed straight through, while route P_2 enables the opposite set of valves to make fluids cross over. (b) a 2-to-1 transposer primitive. Enabling either path P_3 or P_4 performs fluid multiplexing.

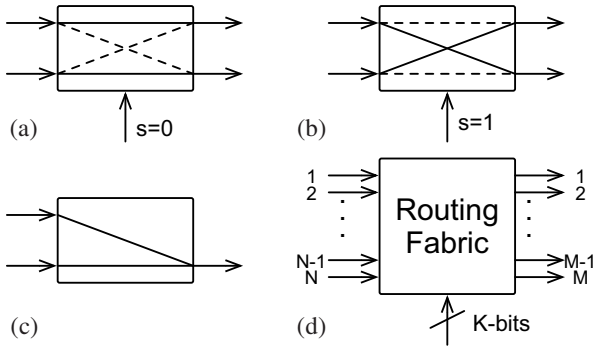

Fig. 2. (a) A microfluidic transposer configured with deasserted control line, passing fluids through. (b) Asserted control lines cause the fluids to cross over. (c) An alternate transposer primitive chooses between two input fluids. (d) In general, a microfluidic routing fabric is designed to pass N number of input ports to M number of output ports as a function of the control port.

then be processed through a variable sequence of biochemical operations while their genomic identity is preserved. To control the DNA barcoding of thousands of heterogeneous cells, a microfluidic routing fabric has been efficiently used [7].

The implications of these security threats are catastrophic. From a system biology perspective, if cellular samples are not properly barcoded, the integrity of a single-cell application may be compromised even before regular genomic analysis begins. Single-cell applications such as DNA forensics assume that the cells under study were collected and barcoded in a trustworthy manner and therefore allow making clinical or judicial decisions based on the genomic study. Hence, insecure single-cell microfluidics can have a significant adverse impact on human lives. Our goal in this paper is to advance the security of single-cell analysis that is carried out using a microfluidic routing fabric.

B. Transposer-Based Microfluidic Routing Fabrics

Traditional continuous flow-based microfluidic devices consist of valves and fluid flow channels that are designed for a specific protocol. The use of a routing fabric introduces reconfigurability by allowing a set of input fluid channels to be dynamically redirected to a set of output channels through the application of a suitable control signal. A microfluidic routing fabric based on a transposer hardware primitive was recently proposed [6].

The basic device is illustrated in Fig. 1. Two fluid flow channels with bridging channels are controlled with a set of valves. In [6], this primitive was constructed using a polydimethylsiloxane (PDMS) substrate with ablated polycarbonate stacked above. The valves are formed as discontinuities in the channels, which means the channels are normally closed. An elastomeric membrane covers the valve. This membrane distends into the gap upon vacuum actuation, allowing fluid to flow through.

Valves with the same label share control pins. It is assumed in this work that the state of the two control pins are always complementary; either the controller ensures the correct anti-polarity of control signals, or some fixed hardware provides the inversion. When the control signal for a transposer is off, the fluids flow straight through to the output ports. When the control signal is asserted, the fluids cross over to the opposite port without any contamination. An alternative transposer is shown in Fig. 1(b). One output port can select between two input ports, forming a microfluidic 2-to-1 multiplexer.

The transposer primitive is then used to build more complex routing fabrics that can select between an arbitrary number of inputs and outputs. The architectural specification of these routing fabrics has not been thoroughly researched, with only some initial results on bounds for the number of required transposers under special cases [7]. The state-of-the-art uses routing fabrics that are designed by hand. The problem of how to derive the control signals used to route a fluid from a desired input to output port is non-trivial and can be solved through graph-theoretic algorithms [6], [7]. Additional complexity is added to the routing problem by considering pipelining [7].

C. Representation and Notation

A microfluidic routing fabric permutes input fluid channels to output fluid channels. We define each fluid as being perfectly distinguishable. That is, we assume that the sensors that ultimately read each of the fluids are not influenced by noise or distortion of the fluid signals. We will discuss extensions of our analysis for non-ideal systems in Sec. VI. Fig. 2(d) illustrates a generic routing block with notation as follows: We define a set of N input fluids arranged in order and indexed with the letter $n \in \{1, 2, ..., N\}$ going from top-to-bottom. The M output ports are indexed with the letter $m \in \{1, 2, ..., .M\}$. The control port is defined as $\boldsymbol{s} \in \{1, 0\}^K$ where K is the number of binary reconfigurable primitives in the fabric. Alternate primitives with more states or more complicated architectures may generalize $\boldsymbol{s}$. If we assume that M, N, K are fixed parameters, then the reconfigurable routing fabric can be interpreted as a function $f : \{1, 0\}^K \rightarrow \{0, 1, 2, ..., M\}^N$ where the domain is the control signal and the range is an N-dimensional vector indicating where each input fluid is directed to. If a fluid is not routed to the output, its value is set to 0.

Any routing fabric can be described in terms of an equivalent state-dependent directed acyclic graph [18], [6], [7]. The state-dependent graph g_S is a map defined as $g_S : S \rightarrow G_V$, where S is the set of system states and G_V is the set of graphs with vertex cardinality V. Fig. 3 illustrates the equivalent graphs for the two transposer primitives. Vertices represent fluid branching points while edges represent possible fluid routing paths.

The vertices are arranged on an integer coordinate grid,

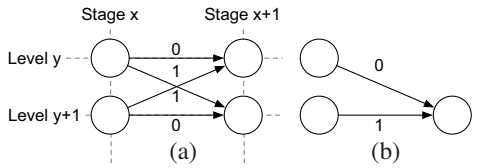

Fig. 3. (a) State-dependent graph equivalent for a 2-to-2 transposer primitive. By convention, x-coordinates are denoted as the *stage* while the y-coordinates are denoted as the *level*. Coordinates increase left-to-right, and top-to-bottom. (b) Graph equivalent for a 2-to-1 transposer primitive. The output vertex can be placed on the same level coordinate as one of the input vertices. The choice of which vertex to use is arbitrary, unless there is a physical meaning.

and adopting a similar nomenclature presented in [6], we call the x-coordinate the *stage* and the y-coordinate the *level*. By convention, the top-leftmost vertex is assigned level and stage 0. Primary inputs to the device can be identified by all vertices with stage 0, and can be indexed directly by the level number. Similarly primary outputs are identified by the highest stage number. Each edge is associated with a transposer, and corresponds to one of two transposer states. We identify these two states with the numbers $s \in \{0, 1\}$ to indicate whether the specific edge is active when the corresponding transposer control signal is a 0 or 1. Each vertex can be uniquely identified by the level and stage number. Decision vertices are shared between transposers, and only the edges are identifiable as belonging to a particular transposer. The placement of vertices on the coordinate grid is in some sense arbitrary; coordinates can be assigned in a way that reflects the physical layout. In this study, coordinates will be assigned such that the distance between neighboring vertices on the same level or stage is 1.

D. Related Work in DMFB Security

Parties in the digital microfluidic biochip design flow can compromise assays and execute denial-of-service or manipulation attacks [12]. The supply chain security of digital microfluidics has also been investigated [19]. A secure randomized checkpoint system for DMFBs was developed and analyzed in [20]. Symbolic reasoning using a golden actuation sequence and a compromised sequence was proposed for attack localization in [21]. In the realm of intellectual property protection, microfluidic encryption of biochemical assays was developed with the use of microfluidic multiplexers [22].

III. Threat Model

The attacker intentionally, or inadvertently, induces faulty operation of the routing fabric such that the fluids to be routed are misdirected to the wrong output ports. This can be achieved either by attacking the electronic controller unit or the fluid control valves. Microfluidic platforms are often integrated with a microcontroller or a computer; sensor data is fed back to the controller to form a closed-loop cyberphysical system, which can implement advanced functions such as error recovery [23]. Software executing on the controller sends signals that drive relays or level-translators, which are then used to actuate the microfluidic platform. Pneumatic fluid control valves are susceptible to tampering due to their large physical scale; prototypes reported in [6] were fabricated by hand with features as large as a millimeter.

We assume that the attacker induces faults into the hardware by tampering with the electronic control hardware or the fluid control valves rather than exploiting software vulnerabilities. These attacks leverage the physical vulnerability of the microfluidic platform, and require less expertise than software attacks. Such attacks are classified as *fault injection attacks*. Fault injection attacks have been studied extensively in the cryptography literature [24], [25] since they can often facilitate cryptanalysis techniques such as differential fault analysis [26]. We further assume the attacker utilizes low-cost fault injection techniques with poor localization, such that a small number of bit flips occur at random locations in the controller's memory.

A. Attack Implications

Under the previously described threat model, the practical implication of an attack are as follows:

1) *Fluid Redirection*. The purpose of a microfluidic routing fabric is to direct a set of fluids from the input ports to the output ports. Under an attack, some fluids may be redirected to the incorrect port, causing droplets to be mislabeled. In Fig. 4, we see that attacking a single transposer in the routing fabric in dashed lines causes fluids at output ports 1 and 2 to be swapped. In an application where each of the fluids is used for a chemical reaction, the fluids at port A and B may be so different as to cause complete failure of the reaction. In a droplet barcoding application, this attack can cause cells to be mislabeled which has consequences for the integrity of scientific inquiry.
2) *Fluid Mixing*. If the control signals of a transposer are fully accessible, then it is possible to place the valves into a state where the input fluids mix. Such an attack has consequences that have yet to be fully understood. Since the control valves in a single transposer cannot be actuated simultaneously, fluid mixing can only occur at the architectural level.
3) *Inducement of Failure Modes*. Reconfiguring the routing fabric into a prohibited state may cause premature failure. Certain hardware primitives may allow multiple inlet valves to flow into the same port, causing excess pressure to build up. Additionally, repeated actuation of the control valves may lead to premature wear. Since the transposer primitive is a recent development, its reliability and failure modes have yet to be fully described.

It is clear that microfluidic routing fabrics are vulnerable to many security threats with serious real-world implications. The problem is thus how to analyze and design routing fabrics with security in mind. The following section introduces new security notions for the analysis of routing fabric designs. After analysis of some simple designs, we can proceed to gain some insights into how to design better routing fabrics.

IV. Security Analysis

Taken at a high level, a reconfigurable fabric is a system that maps input ports to output ports as a function of the control port. The mapping can be one-to-one, many-to-one, or one-to-many depending on the underlying architecture and routing primitives used. The following security analysis seeks to understand how these mappings behave while under attack.

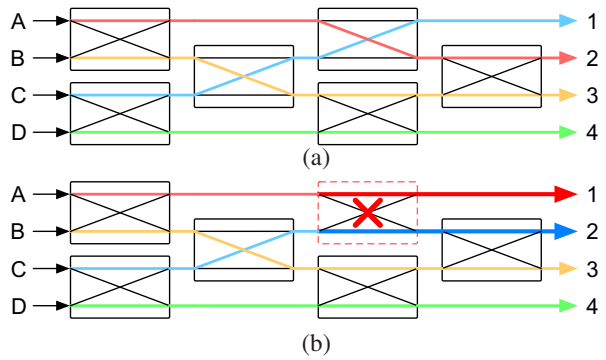

Fig. 4. (a) Each color represents a different fluid flow under normal operation. (b) The transposer in the dashed outline is under attack, causing fluids intended for output ports 1 and 2 to swap places.

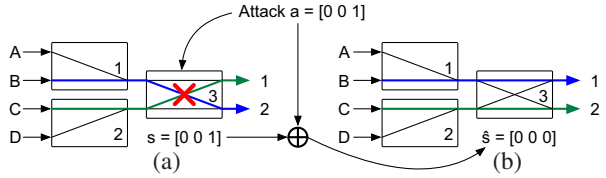

Fig. 5. (a) Simple routing fabric with three transposers. Under state [0 0 1], fluid B (blue) is routed to port 2 and fluid C (green) is routed to port 1. (b) An attack causes transposer 3 to flip states, and the resultant state vector is [0 0 0].

A. Preliminaries

Attacks on the routing fabric alter the control signals. The most general attack consists of arbitrary modifications of the control signals. The fault-injection attack model assumes that an attacker can only alter a single bit. A bit is selected from random and is forced to flip to the opposite state. If we define the routing fabric state as a column vector

$$\boldsymbol{s} = \begin{bmatrix} s_1 & s_2 & \dots & s_K \end{bmatrix}^T \tag{1}$$

where each entry contains 0 if the k-th routing primitive should pass signals straight through or 1 if it should cross them. For example, the simple fabric in Fig. 5(a) has three routing primitives, each of which is uniquely identifiable by the numeric index inside the primitive. The corresponding state vector has dimension 3. When the state is $\boldsymbol{s} = [0\ 0\ 1]$, transposers 1 and 2 are set to pass-through while transposer 3 is set to cross over.

We then define an attack vector $\boldsymbol{a}$ of dimension K where the k-th entry means a bit flip on the k-th transposer:

$$\boldsymbol{a} = \begin{bmatrix} a_1 & a_2 & \dots & a_K \end{bmatrix}^T \tag{2}$$

For instance, if an attack results in transposer 3 flipping its state, we would model the attack with the vector $\boldsymbol{a} = [0\ 0\ 1]$. The state of an attacked routing fabric can be represented as a modified state vector $\hat{\boldsymbol{s}}$ which is the element-by-element XOR of the attack vector and the unmodified state.

$$\hat{\boldsymbol{s}} = \boldsymbol{a} \oplus \boldsymbol{s} \tag{3}$$

Therefore, the attacked state in the example is $\hat{\boldsymbol{s}} = [0\ 0\ 1] \oplus [0\ 0\ 1] = [0\ 0\ 0]$ (Fig. 5(b)).

We will find it useful to classify attack vectors according to their degree. We denote the set of all possible system states as $\boldsymbol{S}$ and attack vectors as $\boldsymbol{A}$, and define classes of attacks using the notation $\boldsymbol{A}_w = \{\boldsymbol{a} \subset \boldsymbol{A} \mid \mathrm{HW}(\boldsymbol{a}) = w\}$, where $\mathrm{HW}(\boldsymbol{x})$ is the Hamming weight of the vector $\boldsymbol{x}$. For example, the set of attacks inducing a single bit flip is $\boldsymbol{A}_1$. Attacks producing a byte flip is $\boldsymbol{A}_8$. Alternately, we may use the subscript w as a percentage of the total number of control signals. Thus, the attack class that flips all bits is $\boldsymbol{A}_{100\%}$, which contains one element: the ones vector of length K.

B. Security Metrics

Abstractly, the problem from a security analysis perspective is to understand how the system architecture in a reconfigurable fabric affects the function f and its response to different attack vectors $\boldsymbol{a}$. For small routing fabrics, it is possible to fully enumerate all the states of the routing fabric and investigate how each attack interacts with the states. Undesirable states and transitions can then be trimmed out. The exponential dependence on the number of primitives means that it soon becomes impractical to do such an analysis, and alternate techniques are required.

Scattering Vector. We define a *scattering vector* $\boldsymbol{sv}$ as a measurement of how far each input fluid deviates from its intended destination. This measurement is made with a fixed configuration input pattern and an attack vector as parameters. If there is no attack, or if the attack is not successful, the scattering vector is the zero vector. The entries of the scattering vector consist of the coordinate differences $(\Delta x_n, \Delta y_n) = (\hat{x}_n - x_n, \hat{y}_n - y_n)$, that is the final coordinates under attack for the n-th input port minus the final coordinates under normal operation.

$$\boldsymbol{sv}(\boldsymbol{s}, \boldsymbol{a}) = \begin{bmatrix} (\Delta x_1, \Delta y_1) \\ (\Delta x_2, \Delta y_2) \\ \dots \\ (\Delta x_N, \Delta y_N) \end{bmatrix} \tag{4}$$

Using the example from Fig. 5, we see that there are four input ports so $N = 4$. For port 1, which is connected to fluid A, and port 4 connected to fluid D, there is no difference between the normal and attacked states. For port 2, the attack redirects fluid B from port 2 to port 1, so the difference in y-coordinates is 1 with no change in x-coordinates.. Similarly for port 3, the difference is 1. Thus, the scattering vector is $\boldsymbol{sv}([0\ 0\ 1]^T, [0\ 0\ 1]^T) = [(0,0)\ (0,1)\ (0,1)\ (0,0)]^T$.

The scattering vector is the output of some function $h : \{1,0\}^K \times \{1,0\}^K \to (\mathbb{Z} \times \mathbb{Z})^N$. It is desirable for this function to be minimized over the entire domain—a perfect attack-resilient routing fabric would have scattering vectors equal to the zero vector. Summing all the entries of the scattering vector leads to a scalar useful for direct comparison, under the assumption that all fluids are equally important. If this does not hold, then a weighting can be introduced. While this metric measures the effect of a particular attack on a particular routing architecture, it does not describe the security of an architecture in general.

Computation of the scattering vector is a straightforward graph traversal problem given a representation of the routing fabric as a directed graph. For each primary input, identifiable by vertices occupying level 0, we enable the set of edges

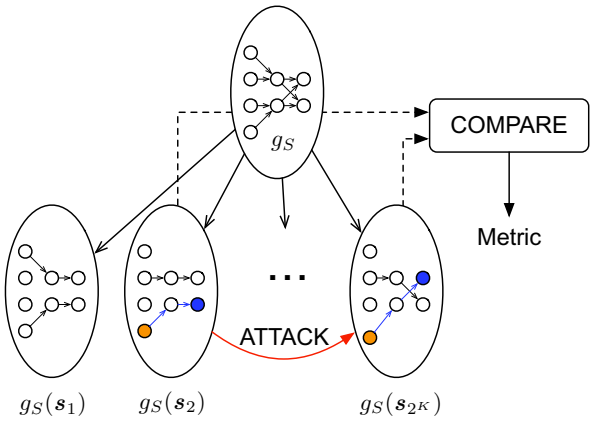

Fig. 6. Visualization of the problem. A state-dependent graph g_S leads to a family of related graphs depending on the current state $\boldsymbol{s}$. Attacks alter the state, and the differences must be quantified.

corresponding to the applied control signal $\boldsymbol{s}$ and traverse the graph until a vertex with no outgoing edges is reached. The same procedure is followed with the control signal under attack $\hat{\boldsymbol{s}}$, and the final coordinates of the two traversals are computed as $(|x_{\boldsymbol{s}} - x_{\hat{\boldsymbol{s}}}|, |y_{\boldsymbol{s}} - y_{\hat{\boldsymbol{s}}}|)$ to build an entry in the scattering vector. Fig. 6 illustrates the concept: We assume that the desired routing results in graph $g_S(\boldsymbol{s}_2)$, and that an attack alters the state such that graph $g_S(\boldsymbol{s}_{2^K})$ is achieved. The fluid starting at the vertex in orange gets routed to the vertex in blue. The scattering is calculated as the coordinate difference of the two blue vertices.

Probability of Scattering. We are primarily interested in knowing whether a particular fluid from some input port is routed to the correct output port under some attack class $\boldsymbol{A}_w$, and if not, how far it has deviated. This means that the system state will be fixed in such a way that the fluid is properly routed, while attack vectors are drawn randomly from the attack class under consideration. We define $E_{n,d}$ as the event that the fluid at the input indexed by n, when routed to any output port, deviates under attack by more than d coordinate spaces. We define the probability of scattering as a vector:

$$\boldsymbol{p}(d) = [P(E_{1,d}) \quad P(E_{2,d}) \quad \ldots \quad P(E_{N,d})]^T \qquad (5)$$

We calculate an individual $P(E_{n,d})$ by defining a helper function *getRoutedStates(n)* that returns all possible ways to route input vertex n to every output vertex. We then attack each of these routed system states and compare coordinate differences using the function *calcScattering*$(\boldsymbol{s}, \boldsymbol{a}, n)$ $= (\Delta x_n, \Delta y_n)$. If an attack results in a combined x-y difference $(\Delta x_n + \Delta y_n)$ greater than d, a counter is incremented. The final probability is the number of successful attacks divided by the number of routes times number of attacks. We repeat this for every possible vertex in the set of input vertices, V_{IN} and collect them to form the probability of scattering vector. Computation of the probabilities has worst-case complexity $\mathcal{O}(N \cdot X \cdot 2^X \cdot {}^K C_w)$, where N is the number of inputs, X is the largest stage, K is the number of transposers, and w is the number of attack bits. The procedure is summarized in Fig. 7.

Computation of the probability vector can be further simplified by identifying structural similarities and symmetries.

Input: Routing fabric graph g_S, attack class $\boldsymbol{A}_w$, distance d
Output: Probability of scattering $\boldsymbol{p}$

```
p ← 0
for each input vertex v ∈ V_IN do
    success ← 0
    R ← getRoutedStates(v)
    for each system state r ∈ R do
        for each possible attack a ∈ A_w do
            if sum(calcScattering(r, a, v)) > d then
                success ← success + 1
            end if
        end for
    end for
    p[v] = success / (|R| · |A_w|)
end for
return p
```

Fig. 7. Pseudocode for calculating the probability of scattering.

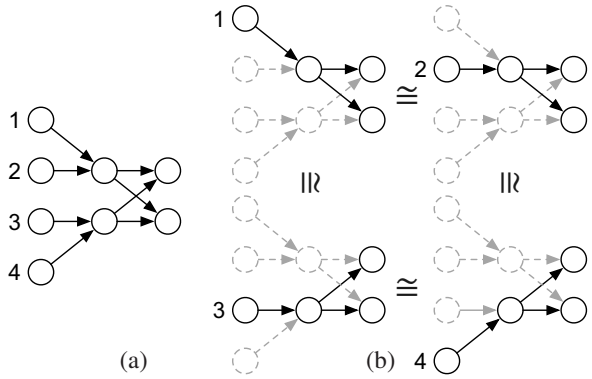

Fig. 8. (a) Graph representation of the 4-to-2 transposer. (b) Each fluid path in the 4-to-2 transposer is structurally equivalent when we ignore non-reachable vertices. Identifying isomorphisms can save computation time if such symmetry is designed into the system intentionally.

Only a subset of transposers are used when routing a fluid from input to output. Attacks that exclusively target transposers not on potential routing paths have no effect and can be ignored. Thus, calculation of the probability of scattering only depends on graph vertices that are reachable from the current input vertex under consideration. When we consider each fluid input path and compare the equivalent graphs when eliminating non-reachable vertices, we can group fluids that have identical structure by calculating whether they are isomorphic. In Fig. 8, the 4-to-2 transposer has four fluid paths. Each of these paths can be simplified, and we see every path is isomorphic to each other. Thus the probability of successful attack on a single fluid path can be calculated, and this result can be duplicated for each isomorphic entry of the probability vector. While efficient determination of graph isomorphism may be limited to special cases, the main benefit of considering isomorphism is that often during the design phase, the routing architecture is known a priori to have symmetry.

A state-dependent graph representing the routing fabric admits several realizations based on the transposer states. Attacks alter the system states, and the differences between the resultant graphs are compared for analysis (Fig. 6). If attacks

are probabilistic, then the set of these comparisons form a sample space. Given our security metrics, the problem becomes more clear: the issue is that a routing fabric has 2^K number of system states where K is the number of binary primitives, while there are ${}^nP_m$ number of useful permutations. If there is a gap between these two numbers, that means there are redundant states. Under certain attack classes, these redundant states may skew the scattering probability distribution. This skew can be either harmful or beneficial for an attacker. For instance, adding redundancy through parallel processing may ensure correct operation under attack by increasing system states.

Average Probability of Scattering. Taking the average of the probability of scattering vector gives a convenient way to compare the security across different architectures. We denote this metric as

$$P_{AVG} = \frac{1}{N} \sum_{n \in V_{IN}} P(E_n, d) \tag{6}$$

and use this as our main security metric. The average probability of scattering is a function of both the attack class and the distance d.

C. Reconfigurability Ratio

Since we are interested in exploring how security trades off with architectural choices, we express architectural information in terms of the *reconfigurability ratio*. This metric describes the gap between the number of functional reconfigurations of the fabric and the number of system states. The goal of a microfluidic fabric is to route a set of n input fluids to m output ports. A perfectly reconfigurable device would allow all m-permutations of the input to output, i.e. ${}^nP_m = n!/(n-m)!$. A semi-reconfigurable device allows some subset of these permutations, with cardinality denoted as $r \leq {}^nP_m$. This number represents the functional reconfigurability. Meanwhile, the number of system states reflects the structural reconfigurability of the fabric. If the primitives used in the routing fabric admit two possible states, and there are K primitives, then we have 2^K number of system states. We express the reconfigurability ratio R_R as

$$R_R = \frac{r}{2^K} \tag{7}$$

With the example in Fig. 5, a fully reconfigurable 4-to-2 transposer would have 12 possible permutations. However, we see that it is impossible for fluids A and B, or fluids C and D to be routed simultaneously. This eliminates 4 permutations, giving $r = 8$. The fabric has $K = 3$ transposers, so the number of system states is 8. Therefore, we evaluate the reconfigurability ratio as $R_R = 1$.

V. Case Study

Microfluidic technology has made high-throughput DNA barcoding a reality, allowing scientists to study gene expression as a function of cell type. The transposer-based microfluidic routing fabrics described in this paper were recently proposed as part of a hybrid single-cell analysis platform [7] for its ability to barcode cells in a space-efficient manner. In this case study, we analyze the security implications of this platform and describe the performance of alternate architectures.

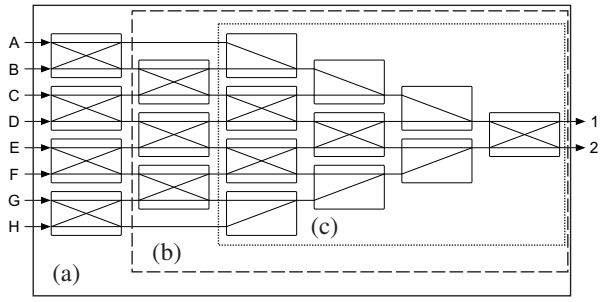

Fig. 9. (a) A 6-level, 8-to-2 routing fabric. Extra transposers permit additional reconfigurability, useful for pipelining operations. (b) Removal of the four input transposers provides some security benefit at the expense of reconfigurability. (c) Removing another three input transposers places 2-to-1 transposer primitives at the input. Fluids A/B and G/H can never be simultaneously routed, lowering of the fabric's functional reconfigurability.

A. Security Implications

The routing fabric utilized in [7] has 6 levels, and permutes 8 inputs to 2 outputs. (Fig. 9(a)). Eight types of barcoding droplets, identified by the letters A through H, are connected to the input ports, to be dispensed on-demand to any of the two output ports. Once dispensed, these droplets are routed to the rest of the platform for further processing and sensing using digital microfluidic technology integrated with actuators and optical detectors.

To illustrate what can potentially go wrong, we show an example fluid routing in Fig. 10(a). Barcode type A is to be dispensed to port 2 while barcode type F is to be dispensed to port 1. These barcodes are intended for simultaneous application to two cell droplets. If an attacker causes a fault within the system such that all transposer states are swapped (i.e. an attack within class A_{17}), we see that barcode A is halted at an intermediate transposer. Barcode F is still able to make it to the correct port. Barcode H, which was never intended for use at this point in the protocol, is now dispensed to port 2.

The practical implication of this attack for the cell study is that the biochemical procedure will provide misleading outcomes. As a consequence of this attack, a cell that has been identified as type A (based on the *in vivo* activity of a certain biomarker) will be wrongly tagged with a barcode that belongs to a different sub-population of type H. During the process of biomolecular analysis, cells are lysed and type-driven DNA analysis is applied by using the barcode. Hence, the alteration of barcoding causes the gene reads of type-A cells to be interpreted as a part of type-H genomic landscape, thus leading to a false conclusion on the gene expression of type-A cells. If this routine analysis is carried out as a part of a DNA forensic investigation, a suspect (with type-A cells collected from a crime scene) may eventually escape prosecution.

B. Security Analysis

The 8-to-2 transposer has full reconfigurability. It permits all 56 possible permutations of inputs to be routed to the output, while the structural reconfigurability is high, with 2^{17} states. P_{AVG} values are summarized in Table I. We observe

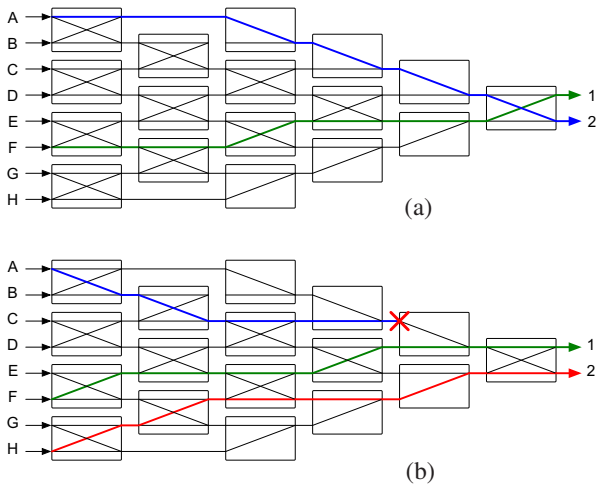

Fig. 10. (a) Routed fabric under normal operation. Fluid A (blue) is routed to output port 2 while fluid F (green) is routed to output port 1. (b) After all transposers are attacked, fluid H (red) ends up at output port 2 while fluid F (green) moves to port 1. Fluid A (blue) is blocked at an intermediate transposer.

TABLE I. AVERAGE PROBABILITY OF SCATTERING VS DISTANCE AND ATTACK CLASS FOR 8-TO-2 ROUTING FABRICS

	4-level			5-level			6-level		
d	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3
0	0.40	0.65	0.80	0.28	0.48	0.63	0.35	0.58	0.72
1	0.25	0.45	0.60	0.18	0.32	0.44	0.23	0.40	0.53
2	0.23	0.41	0.55	0.16	0.30	0.41	0.21	0.38	0.49

that single-bit attacks have a fairly low success rate, and that increasing the attacks class by even a single bit can dramatically increase the success rate.

An alternate 4-level 8-to-2 fabric was also considered (Fig. 9(c)), which features only ten transposers and limits the number of permutations to 52 out of the possible 56. The values of P_{AVG} in Table I show that in general, the probability of a successful attack is higher than in the 6-level architecture. This result is counter-intuitive, in that the fabric with a higher degree of flexibility is also more secure. This illustrates the fact that architectural choices can have a greater impact on security than strictly reducing reconfigurability.

We also analyze a new 5-level design (Fig. 9(b)). This fabric does not offer the full pipelining capabilities of the 6-level design, but does offer more functional reconfigurability than the 4-level design. P_{AVG} values in Table I show that it performs better than both the 4 and 6-level architectures under every attack parameter. The three extra transposers beyond the 4-level design do not provide a substantial benefit in functional reconfigurability, but introduces an exponential increase in structural reconfigurability. This redundancy provides better security performance.

VI. DISCUSSION

Table II summarizes the properties of the architectures studied in this paper. Security, reconfigurability, and redundancy trade-off with each other in a complex fashion during the architectural design phase. To illustrate this point, we

TABLE II. PROBABILITY OF SCATTERING AND RECONFIGURABILITY METRICS FOR SELECTED ROUTING FABRIC ARCHITECTURES

Routing Fabric	r	${}^nP_m$	K	R_R	$P_{AVG}(A_1, d=0)$
2-level 4-to-2	8	12	3	1.000	0.67
3-level 3-to-3	6	6	3	0.7500	0.72
4-level 8-to-2	52	56	10	0.0508	0.40
5-level 8-to-2	56	56	13	0.0068	0.28
6-level 8-to-2	56	56	17	0.0040	0.35

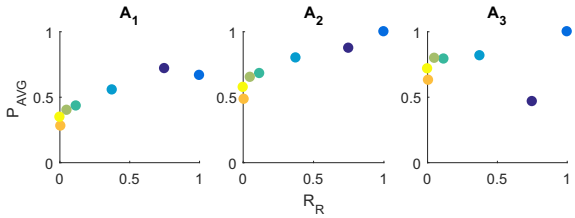

Fig. 11. Scatter plots of average probability of scattering (P_{AVG}) vs. reconfigurability ratio (R_R) for attack classes A_1, A_2, A_3 evaluated over distances greater than 0. Each point represents a unique transposer fabric architecture. The trend is that redundant states (lower R_R) generally results in better security (lower P_{AVG}).

plot the average probability of scattering in Fig. 11 under three attack classes and distances greater than 0 for all the architectures studied in this paper, as well as additional architectures from [6]. For attack classes A_1 and A_2, a clear trend emerges: a lower probability of scattering is more likely when the ratio between functional states to structural states is lower. Although, as demonstrated in the DNA barcoding fabric case study, further lowering of R_R does not always lead to benefits. The relationship is less clear for attack class A_3.

A. Question & Answer

To highlight some key points about this work, we present further analysis in a question and answer format.

Does any evidence exist for the proposed adversarial model?
Routing fabrics are experimental, so no evidence for tampering exists. However, there is ample evidence for tampering in applications that microfluidics are expected to make headway. See: drug screening [27], environmental monitoring [28], research laboratories [29].

Since the routing fabric is only experimental, is there any value in this analysis given that a commercial implementation may diverge significantly?
Yes, as we have mentioned the framework here is general and can accommodate other metrics and attack models. Furthermore, security studies of experimental technologies are important; security vulnerabilities are being discovered in a multitude of unlikely places. Security-minded design may drive further adoption and bridge the gap between engineering and the life sciences [30].

Why is distance used as the key metric for the impact of an attack?
Distance is used because barcodes used in the single-cell analysis platform impose a hierarchy. Related cells are to be tagged with similar barcodes. If barcodes are misapplied, there is a possibility that due to close grouping that at least the droplets can be within the same grouping.

Can the methodology be extended to account for alternate metrics and attacks?
Yes, the distance metric can be substituted for other properties that fluids can pick up as they move along the fabric, e.g. temperature and contaminants. If a probabilistic attack model is available, it can be used to scale the probability of scattering.

How does the analysis differ from fault-tolerance analysis?
Fault-tolerant analysis and design is based on the premise that typical hardware faults can be characterized. In contrast, this work assumes fault injection attacks with poor localization, and models it as a random perturbation on the control signals.

How do architectural choices affect security?
Adjusting the allowable permutations benefits security only in certain use cases. The use of the 2-to-1 transposer primitive prevents the simultaneous routing of adjacent fluids. Yet in the case study, we saw that the 4-level 8-to-2 fabric was the least secure design. Limiting permutations should only be leveraged if there are two fluids that should never be output at the same time.

B. Limitations

In our analysis we have considered that every fluid is equally important. Certain fluids may actually be critically important, while others may be benign (e.g., wash fluids). The perfectly distinguishable assumption on the sensors is optimistic, and in reality, any real physical system is subject to noise. The design of the sensors as well as the coding of the DNA barcodes could potentially affect the distance metric; if a fluid is misdirected but nearly indistinguishable from the correct fluid, then what is the practical implication? These questions are important, but are outside of the scope of this paper and will be addressed as a part of future work.

VII. Conclusion

We presented an analysis of emerging microfluidic hardware primitive in terms of its security and architecture. These new concepts were applied to designs presented in the literature for real-time quantitative analysis. Security trade-offs are fundamental to many systems, but have been difficult to quantify due to complexity. The simple structure and low complexity of microfluidic devices has lent itself to analysis, and we find that in general, a lower reconfigurability ratio (i.e. more redundancy) generally leads to better security properties.

There may be other systems that are amenable to the security analysis presented here, where possible system states are quantified. New microfluidic routing primitives could be investigated; it was noted that in [6] that multiplexer-demultiplexer pairs could have been implemented instead of the transposer primitive. Furthermore, this paper has left open the question of how to synthesize an optimal architecture. An optimal design that considers area, routing time, reconfigurability and security together using optimization techniques or combinatorial analysis would be interesting research to pursue. The security of microfluidics itself is an open question, which could be tackled from a multitude of angles beyond the threat model used here.

References

[1] G. M. Whitesides, "The origins and the future of microfluidics," *Nature*, vol. 442, no. 7101, pp. 368–373, 2006.

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

[3] Y. Luo *et al.*, *Hardware/software Co-Design and Optimization for Cyberphysical Integration in Digital Microfluidic Biochips*. New York, NY: Springer, 2014.

[4] F. Su and K. Chakrabarty, "High-level synthesis of digital microfluidic biochips," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 3, no. 4, p. 1, 2008.

[5] ——, "Module placement for fault-tolerant microfluidics-based biochips," *ACM Trans. Des. Autom. Electron. Sys.*, vol. 11, no. 3, pp. 682–710, 2006.

[6] R. Silva *et al.*, "A reconfigurable continuous-flow fluidic routing fabric using a modular, scalable primitive," *Lab. Chip*, vol. 16, no. 14, pp. 2730–2741, 2016.

[7] M. Ibrahim *et al.*, "CoSyn: efficient single-cell analysis using a hybrid microfluidic platform," in *Proc. Conf. Des. Autom. Test Europe*, Lausanne, Switzerland, Mar. 2017.

[8] ——, "BioCyBig: a cyberphysical system for integrative microfluidics-driven analysis of genomic association studies," *IEEE Trans. Big Data*, in press.

[9] A. Cardenas *et al.*, "Challenges for securing cyber physical systems," in *Proc. Workshop Future Dir. Cyber-physical Syst. Security*, Newark, NJ, Jul. 2009, p. 5.

[10] F. C. Fang *et al.*, "Misconduct accounts for the majority of retracted scientific publications," *Proc. National Academy Sci.*, vol. 109, no. 42, pp. 17 028–17 033, 2012.

[11] M. Yarborough, "Taking steps to increase the trustworthiness of scientific research," *The FASEB Journal*, vol. 28, no. 9, pp. 3841–3846, 2014.

[12] S. Subidh Ali *et al.*, "Security assessment of cyberphysical digital microfluidic biochips," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 13, no. 3, pp. 1–14, 2015.

[13] B. Bruijns *et al.*, "Microfluidic devices for forensic DNA analysis: a review," *Biosensors*, vol. 6, no. 3, p. 41, 2016.

[14] L. Marle and G. M. Greenway, "Microfluidic devices for environmental monitoring," *TrAC, Trends Anal. Chem.*, vol. 24, no. 9, pp. 795–802, 2005.

[15] J. El-Ali *et al.*, "Cells on chips," *Nature*, vol. 442, no. 7101, pp. 403–411, 2006.

[16] S. Hosic *et al.*, "Microfluidic sample preparation for single cell analysis," *Anal. Chem.*, vol. 88, no. 1, pp. 354–380, 2015.

[17] A. M. Klein *et al.*, "Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells," *Cell*, vol. 161, no. 5, pp. 1187–1201, 2015.

[18] M. Mesbahi, "State-dependent graphs," in *Proc. IEEE Conf. Decision and Control*, vol. 3, Lahaina, HI, Dec. 2003, pp. 3058–3063.

[19] S. S. Ali *et al.*, "Supply-chain security of digital microfluidic biochips," *Computer*, vol. 49, no. 8, pp. 36–43, 2016.

[20] J. Tang *et al.*, "Securing digital microfluidic biochips by randomizing checkpoints," in *Proc. IEEE Int. Test Conf.*, Fort Worth, TX, Nov. 2016, pp. 1–8.

[21] P. Roy and A. Banerjee, "A new approach for root-causing attacks on digital microfluidic devices," in *Proc. IEEE Asian Hardware-Oriented Security Trust Symp.*, Yilan, Taiwan, Dec. 2016, pp. 1–6.

[22] S. S. Ali *et al.*, "Microfluidic encryption of on-chip biochemical assays," in *Proc. Biomed. Circuits Syst. Conf.*, Shanghai, China, Oct. 2016, pp. 152–155.

[23] Y. Luo *et al.*, "Error recovery in cyberphysical digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 59–72, 2013.

[24] A. Barenghi *et al.*, "Fault injection attacks on cryptographic devices: theory, practice, and countermeasures," *Proc. IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[25] H. Bar-El *et al.*, "The sorcerer's apprentice guide to fault attacks," *Proc. IEEE*, vol. 94, no. 2, pp. 370–382, 2006.

[26] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Proc. Annual International Cryptology Conference*, Santa Barbara, CA, Aug. 1997, pp. 513–525.

[27] R. G. Johnston *et al.*, "Research note: The security of urine drug testing," *Journal of Drug Issues*, vol. 39, no. 4, pp. 1015–1028, 2009.

[28] Department of Justice, U.S. Attorneys Office, District of Massachusetts, "Former Berkshire power manager sentenced for conspiring to tamper with air pollution monitors," May 2017. [Online]. Available: https://www.justice.gov/usao-ma/pr/former-berkshire-power-manager-sentenced-conspiring-tamper-air-pollution-monitors

[29] The New York Times. (2017, Jan.) A Crime in the Cancer Lab. [Online]. Available: https://nyti.ms/2jJc5rT

[30] H. H. Caicedo and S. T. Brady, "Microfluidics: the challenge is to bridge the gap instead of looking for a 'killer app'," *Trends Biotechnol.*, vol. 34, no. 1, pp. 1–3, Jan. 2016.