# Analysis and Design of Tamper-Mitigating Microfluidic Routing Fabrics

Jack Tang, *Member, IEEE,* Mohamed Ibrahim, *Member, IEEE,* Krishnendu Chakrabarty, *Fellow, IEEE,* and Ramesh Karri, *Senior Member, IEEE*

***Abstract*—Microfluidic routing fabrics are reconfigurable primitives that permit the dynamic redirection of fluids on a flow-based microfluidic biochip. Such primitives are bringing the benefits of rapid prototyping and on-the-fly reconfigurability from integrated circuits to the microfluidic domain. An unfortunate side effect of this increased flexibility is susceptibility to tampering. A malicious adversary can alter either the electronic control signals or the pneumatic control lines used to drive the routing fabric. In this work, we provide a high-level security assessment of microfluidic systems utilizing routing fabrics, and analyze their security under actuation tampering attacks. We show that under reasonable assumptions, the permissible states of a routing fabric forms a probability distribution. We provide methods for efficiently determining this distribution through a binary tree representation. We then show how to synthesize routings fabrics that exhibit well-defined behaviors. We call a routing fabric designed in such a way *tamper-mitigating*, as it makes the effects of tampering probabilistically less severe. We then show how the proposed methodology can be used to protect a forensic DNA barcoding application from attack.**

***Index Terms*—Flow-based microfluidic biochip, routing fabric, transposer, security, tampering, mitigation.**

## I. Introduction

Flow-based microfluidic biochips have emerged as an effective means of realizing the laboratory-on-a-chip [2], [3]. By integrating fluid handling channels and chambers, microvalve-based flow control, and sensors on elastomer substrates, many biochemical protocols once relegated to complex, manual laboratory procedures can now be performed efficiently and automatically in a miniaturized platform [4], [5]. Among the myriad of microfluidic technologies available today, flow-based microfluidics are one of the most developed and have been successfully commercialized in benchtop platforms such as the Fluidigm BioMark HD [6]. The latest research in flow-based biochips has drawn from well-established concepts in the semiconductor industry in an effort to manage increasing complexity while streamlining design, fabrication, and operation. This has led to several breakthroughs in microfluidic large scale integration (mLSI) [7], design automation [8], [9], and test [10]. Even reconfigurable flow-based biochips (RFBs) analogous to field-programmable gate arrays (FPGAs) have been reported, which enables rapid prototyping and real-time adjustments to biochemical protocols [11], [12].

A side effect of the adoption of semiconductor design and fabrication techniques is the unintentional incorporation of their associated security vulnerabilities. System complexity and programmability can be leveraged by an attacker to cause unintended behavior. A state-of-the-art flow-based microfluidic biochip is a complex cyberphysical system and as such presents several attack surfaces for exploitation [13], [14]. Furthermore, as biochips become more advanced and move to the commercial sector, a horizontally integrated supply chain will become attractive [15]. This has been the predominant design paradigm in the semiconductor industry for decades, and has resulted in an untrusted supply chain where hardware Trojans and intellectual property theft are prevalent [16]. This is especially worrisome considering the potential catastrophic consequences: result manipulation, and destruction of precious, difficult-to-replace samples [17].

Microfluidic security threats must be taken seriously given that several cases of fraud and misconduct have been reported in applications where microfluidics are designed to be employed. For instance, in 2013 it was revealed that a third-party research laboratory tasked with FDA drug screening was essentially fabricating test results [18]. Paradoxically, the adoption of microfluidics may make successful attacks more plausibly deniable; built-in monitoring and logging capabilities may be difficult to bypass [17]. Microfluidic system designers would do well to fully analyze the performance of their systems under attack.

This paper investigates the analysis and design of one RFB in detail: the microfluidic routing fabric based on transposer primitives [11], which was employed as an efficient droplet barcoding mechanism for single-cell analysis [19]. The transposer primitive is illustrated in Fig. 1(a). Two fluid flow channels with bridging channels are controlled with a set of valves. In [11], this primitive was constructed using a polydimethylsiloxane (PDMS) substrate with ablated polycarbonate stacked above. The valves are formed as discontinuities in the channels, and can be normally open or normally closed. An elastomeric membrane covers the valve. This membrane distends into the gap upon vacuum actuation (or pressurized

Manuscript received June 1, 2018; revised December 3, 2018; revised January 28, 2019; accepted March 6, 2019. A preliminary version of this paper was published in Proc. IEEE Int. Conf. Comput. Des., Newton, MA, USA, November 2017, pp. 25-32 [1]. This work was supported in part by the Army Research Office under grant number W911NF-17-1-0320, the National Science Foundation under grant CNS-1833624, the NYU Center for Cyber Security (cyber.nyu.edu), and the NYU-AD Center for Cyber Security (sites.nyuad.nyu.edu/ccs-ad/).

J. Tang and R. Karri are with the Department of Electrical and Computer Engineering, New York University, Brooklyn, NY, 11201 USA (e-mail: jtang@nyu.edu; rkarri@nyu.edu).

M. Ibrahim and K. Chakrabarty are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, 27708 USA (e-mail: mohamed.s.ibrahim@duke.edu; krish@duke.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier xx.xxxx/TCAD.2018.xxxxxxx

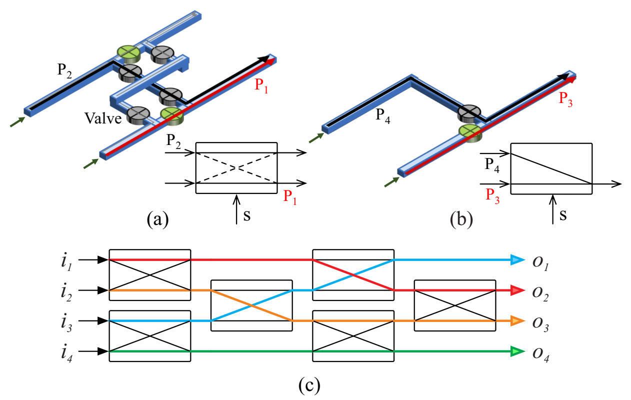


Fig. 1. (a) A 2-to-2 transposer primitive and its schematic representation. Route $P_1$ enables the valves in green and disables the valves in grey, causing both fluids to be passed straight through, while route $P_2$ enables the opposite set of valves to make fluids cross over. (b) A 2-to-1 transposer primitive and its schematic representation. Enabling either path $P_3$ or $P_4$ performs fluid multiplexing. (c) Complex routing fabrics can be constructed using transposer primitives, such as this 4-to-4 fabric that permits all permutations of inputs to outputs [11].

actuation), causing fluid to flow (or be blocked).

Control lines for the six valves in Fig. 1(a) are grouped into two pins. 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 [20]. The two control pins can thus be considered as a single control port accepting a single control bit. In one state, the fluids flow straight through to the output ports. In the other, the fluids cross over to the opposite port without any contamination. The control signals used to program a set of transposers is called a *control vector* and is denoted by $s$.

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, or alternately, a 1-to-2 demultiplexer. The transposer primitive is then used to build more complex routing fabrics that can select between an arbitrary number of inputs and outputs (Fig. 1(c)).

We study microfluidic routing fabrics under *actuation tampering attacks*. We define an actuation tampering attack as a malicious modification of the transposer control state. In this work, we focus on actuation tampering achieved through the injection of physical force directly into the control lines [21]. Such attacks are important to consider since one of the main usage scenarios for microfluidic systems is at the point-of-care (PoC), where devices are physically exposed. Furthermore, physical tampering requires almost no technical sophistication given that pneumatic control valves are large enough to be manipulated by hand. While PoC diagnostics can greatly increase the quality of care in medical applications by lowering errors and result turnaround time, it has been noted that the benefits of PoC diagnostics can only be realized with the adoption of security mechanisms [22]. This work is a novel hardware-based design technique contributing to goal of low-error, trustworthy microfluidic systems deployable at the PoC.

Intuitively, a biochip designed for a single function is physically unable to realize an undesired operation. On the other hand, a reconfigurable biochip could be configured in a way that is not only undesirable, but potentially destructive. Thus, microfluidic routing fabrics have security implications. This work is the first to address both the analysis and design of secure RFBs. We call a routing fabric designed using our methodology *tamper-mitigating*, as it probabilistically reduces the consequences of an attack after it has occurred. The contributions of this paper beyond those in the initial conference publication are as follows:

1) We provide a high-level security assessment of flow-based microfluidic platforms with an emphasis on actuation tampering attack surfaces.
2) We describe an improved threat model that is more realistic than that considered in the conference version. From this, we show that the security of a routing fabric is determined by the distribution of its routing states.
3) We present a probabilistic graphical model for evaluating the state distribution, and define distributions and metrics to classify the security of routing fabrics.
4) We show how to synthesize a routing fabric with arbitrary security attributes based on the merging of binary decision diagrams.
5) We demonstrate how the proposed analysis and design methodology can be used in practice for uncontrollable and fail-safe operation.

We note that there are two design problems associated with microfluidic routing fabrics, each with implications for performance under actuation tampering attacks. The first is architectural synthesis, which is the problem of constructing the routing fabric with routability guarantees. For mason-brick pattern fabrics, sufficient conditions for routing $n$ inputs to $m$ outputs have been derived [19]. The initial version of this paper studied the effect of reconfigurability and architecture on security, which remains the focus of this paper. The second is the routing problem, which is the determination of transposer states such that the desired input fluids reach the desired output ports. Algorithms have been proposed which leverage graph-theoretic algorithms [11], [19]. We do not consider routing in this paper, although it may be a promising area for research.

This paper is organized as follows. We perform a security assessment of microfluidic routing fabrics in Section. II, describing attacks, attack surfaces and their implications. The problem addressed in this work is summarized in Section III. We present a framework for analyzing routing fabrics in Section IV. We then present our synthesis methodology in Section V. Applications in Section VI demonstrate how tamper-mitigating routing fabrics can be designed and used in practice. We review related works in Section VII. We conclude and remark on future research in Section VIII.

## II. Security Assessment

The structure of a typical cyberphysical flow-based microfluidic biochip platform is illustrated in Fig. 2. Samples and reagents are loaded manually onto the biochip, which may contain hardware for manipulation of fluids. Inside the platform, the biochip connects to a routing fabric for interface with additional fluid processing blocks such as heaters and detectors. The routing fabric is used to connect different fluid

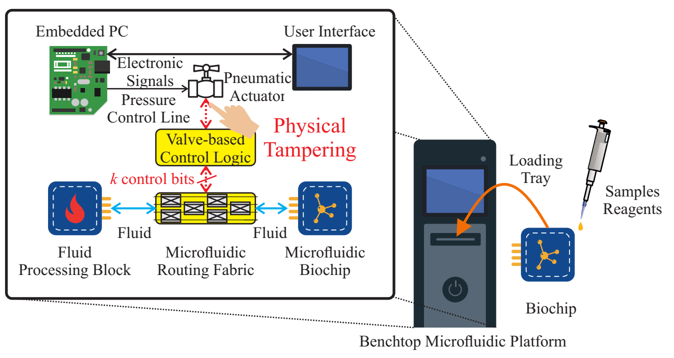


Fig. 2. Typical construction of a cyberphysical flow-based microfluidic biochip platform. When deployed as a point-of-care solution, these platforms are prone to physical tampering by end users and can result in undesired operation.

processing elements and reservoirs, but is not used to process fluids directly. It only serves as an interconnection network. Pneumatic control lines are connected from the biochip to actuators that are computer-controlled.

We assume that the biochips under consideration utilize a state-of-the-art process with the ability to integrate valve-based control logic [23], [24], [25]. Such a biochip would allow the designer to incorporate blocks such as multiplexers, demultiplexers, and even processors such that *a single pneumatic control line* can drive all the control valves in the biochip [26], [27]. Control signals are sent serially over the pneumatic control line and decoded on-chip using valve-based control logic. This is critical as pneumatic control lines are often bulky, with dimensions on the order of millimeters [11]. Recent research [28] has sought to address precisely this issue through pin-constrained design but this is beyond the scope of this work as they are targeted toward more complex biochips.

## A. Threat Model: Physical Tampering

The attacker is a malicious end user or someone with physical access to the microfluidic platform at the point-of-care. They are interested in *stealthily* altering results. Such an attacker poses the greatest threat to undermining the quality of a diagnostic test result as the barrier to tampering is low [22] while motivation is high—spoofed results can be used to falsify compliance when the true result does not match the attacker's desired outcome [18], [29]. Note that this stealthy attack model excludes denial-of-service (DoS) attacks.

Such an attacker, under the tampering taxonomy in [30], is a Class I attacker: a clever outsider who lacks detailed knowledge of the system and has little to no access to sophisticated equipment. Such an attacker will often attempt to leverage existing weaknesses rather than create new ones. We assume that the majority of end users are potential Class I attackers. Expending effort to reduce Class I attacks is warranted given that no evidence exists for Class II (knowledgeable insider) and Class III (funded organization) attacks in the context of microfluidic platforms and medical devices in general.

The attacker induces faulty operation of the routing fabric such that the fluids to be routed are misdirected to the wrong ports. Inducing faults using the fluid control valves leverages the physical vulnerability of the microfluidic platform and requires less expertise than software attacks. Related attacks in the literature are classified as *fault injection attacks*. In cryptographic hardware, such attacks [31], [32] can facilitate cryptanalysis techniques such as differential fault analysis [33].

We assume the attacker is able to open the microfluidic platform to expose the pneumatic control lines. This is a given since these platforms are often constructed using sheet-metal chassis and standard screws. Constructing a platform that is physically tamper-resistant would drive up manufacturing costs while reducing serviceability. Further, we assume the attacker does not possess a device that can synchronize injected faults with the biochip controller. Since a single pneumatic control line drives the entire biochip, the attacker has to perfectly time their fault injection attacks in order to precisely control the state of the biochip. The attacker is motivated to tamper with the control state rather than directly swapping the fluid input ports for several reasons:

- Swapping the inputs will likely result in a DoS attack.
- Sensors in the microfluidic platform may easily detect if the samples/reagents at the input reservoirs are incorrect.
- Switching fluids during the execution of an assay through manipulation of valves is stealthy and minimally invasive.
- Routing fabrics can be used as an intermediary between other functional blocks and as such may not have a direct relationship with the inputs of the system.

When the control signals are tampered with, the result is modeled as a change in the control vector from a known state to a randomly selected control vector. The set of control vectors maps into routing fabric states, and therefore the routing fabric states induce a probability distribution. We note that by considering this threat model, we eliminate “low-hanging fruit” for attackers. Consider the alternative: a biochip with parallel control lines fully exposed. By visual inspection [34], an attacker can precisely change the biochip state.

## B. 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. In Fig. 3, 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

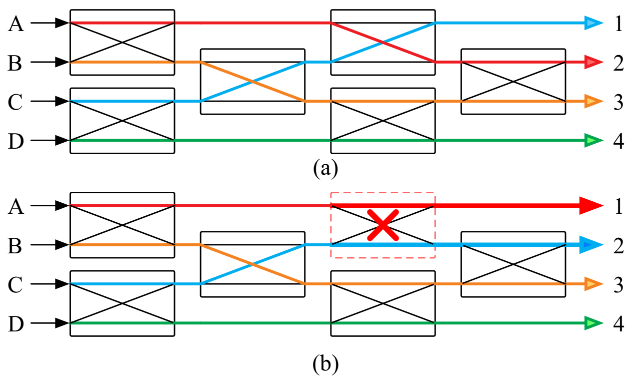


Fig. 3. Fluid redirection under attack. (a) Each color represents a different fluid flow under normal operation. (b) The transposer in the dashed outline is under attack, causing fluids intended for output ports 1 and 2 to swap places.

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. Techniques have been developed to increase the reliability of routing fabrics under disconnection fault models due valve failure [35].

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 quantifiable security guarantees.

## III. Problem Overview

Without the ability to synchronize and perfectly alter the serial pneumatic control signal, the attacker is essentially limited to guessing a random control vector. The physical routing that results is therefore randomly chosen from a sample space consisting of all the possible transposer states.

Several transposer states may map into equivalent physical routings. Therefore, some physical routings are more likely than others. The routing fabric architecture induces a probability distribution, and this distribution determines the performance of the fabric under attack (Fig. 4). Put another way, given a routing fabric, we would like to know *what will most likely occur if the operator of the microfluidic platform is to lose control of the system?* Then, we would like to know *how do we design a microfluidic routing fabric that mitigates actuation tampering attacks?* These are the analysis and design problems described in Section IV and VI, respectively.

## IV. Routing Fabric Analysis

In this section, we develop an efficient analysis framework for evaluating security properties of routing fabrics.

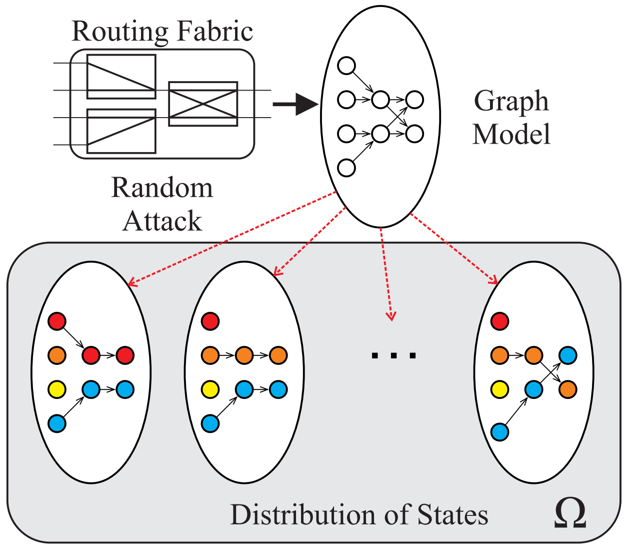


Fig. 4. Overview of the problem. A routing fabric admits several states as a function of a control vector. An attacker can only randomly choose control vectors, so the state space forms a probability distribution. We seek to analyze this distribution, and understand how to synthesize routing fabrics with desirable distributions.

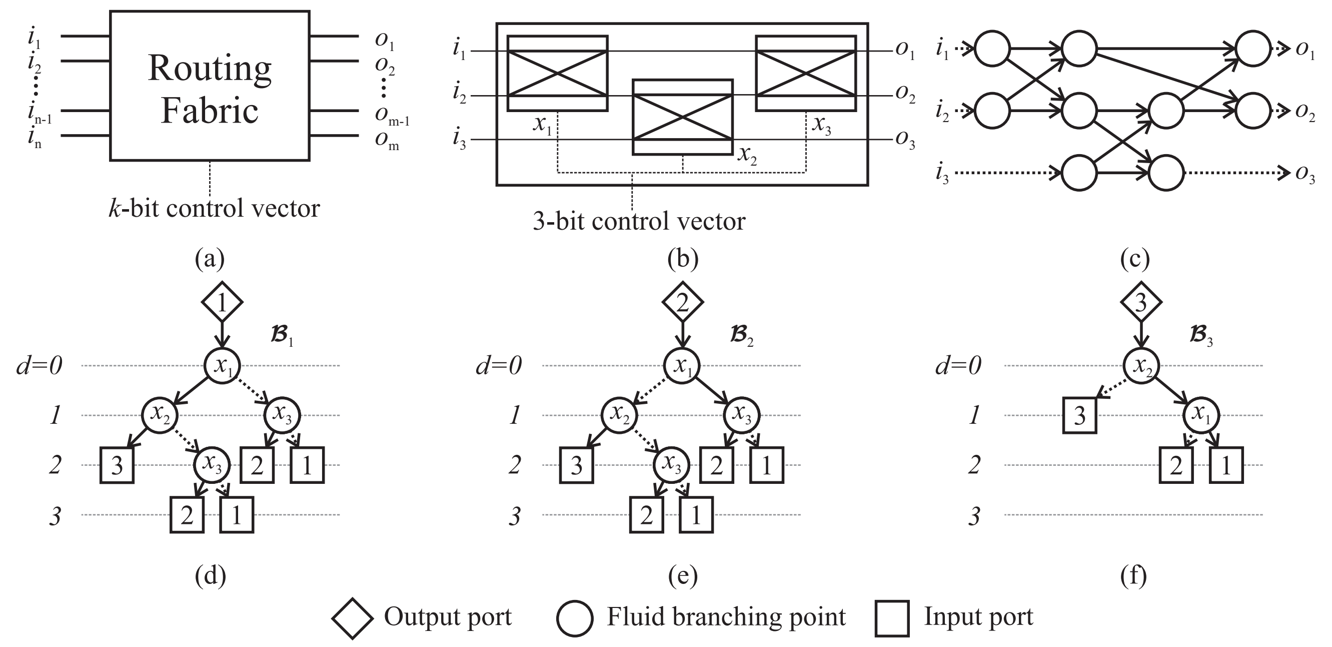
### A. Modeling Preliminaries

Fig. 5(a) illustrates a generic routing block with a set of $n$ input fluids $\{i_1, i_2, ..., i_n\}$ and $m$ output ports $\{o_1, o_2, ..., o_m\}$. The control port accepts a control vector $\boldsymbol{s} \in \{1, 0\}^k$ where $k$ is the number of binary reconfigurable primitives in the fabric. The result of applying a particular control vector can be observed at the output as a vector with $m$ entries, taking on values from the set $\{1, 2, ..., n\}$ to indicate which input fluid is present. If we assume that $m, n, k$ are fixed parameters, then the routing fabric can be interpreted as a function $f : \{1, 0\}^k \rightarrow \{0, 1, 2, ..., n\}^m$ where the domain is the set of all control vectors and the range is the set of all output vectors.

We model attacks using a set of random variables $\{X_1, X_2, ..., X_k\}$ where each $X_i \sim \mathit{Bernoulli}(0.5)$ corresponds to a transposer control line and $k$ is the number of control bits. This model is based on two assumptions: first, that an attacker randomly perturbs the pneumatic control line as a function of time and second, that the attacker must guess how many faults to inject. In order to know how many faults to inject, an attacker must know the current routing fabric state. Extracting this information is impractical given the physical tampering point-of-care threat model and the fact that multiple routing fabric states can be used to achieve the same fluid routing (i.e., the routing problem [19]). We can define a related random variable $Y_i = f_i(X_1, X_2, ..., X_k)$ for each output port $o_i$, that indicates which input fluid $i_i$ appears. The function $f_i$ describes how the routing fabric architecture behaves given a realization of the control vector.

### B. Physical Graph Model

Any routing fabric can be described in terms of an equivalent directed acyclic graph (DAG) $\mathcal{F}_{n \times m} = (\mathcal{D}_{n \times m}, \mathcal{S}_{n \times m})$ where each vertex $d_i \in \mathcal{D}_{n \times m}$ represents a decision point and



$i_1$ $i_2$ $i_{n-1}$ $i_n$
Routing Fabric
$o_1$ $o_2$ $o_{m-1}$ $o_m$
$k$-bit control vector
(a)
$i_1$ $i_2$ $i_3$ $x_1$ $x_2$ $x_3$ $o_1$ $o_2$ $o_3$
3-bit control vector
(b)
$i_1$ $i_2$ $i_3$ $o_1$ $o_2$ $o_3$
(c)
$\mathcal{B}_1$ $\mathcal{B}_2$ $\mathcal{B}_3$
d=0
1
2
3
(d)
(e)
(f)
Output port
Fluid branching point
Input port

Fig. 5. Routing fabric analysis. (a) Generic routing fabric with $n$ inputs and $m$ outputs, consisting of $k$ transposers. (b) A 3-to-3 routing fabric ($\mathcal{F}_{3\times3}$) composed of 3 transposers. (c) Physical graph model used for routing. (d) Routing graph models explicitly show how transposer states lead to fluid routings. $\mathcal{B}_1$ has five possible routings. (e) Sub-routing graph model $\mathcal{B}_2$. (f) Sub-routing graph model $\mathcal{B}_3$. Note that the arrows are opposite to the direction of fluid flow.

each edge $s_i \in \mathcal{S}_{n\times m}$ represents a fluid flow channel between the decision points [11], [19]. We call such a representation the *physical graph model* of the routing fabric, as the model can be uniquely mapped to a hardware implementation. This representation can be extended to be state-dependent in order to explicitly model the changes in topology that occur when fluids are actively routed [36], [21]. Consider the 3-to-3 routing fabric in Fig. 5(b). Its physical graph model is shown in Fig. 5(c), and shows all the possible fluid pathways. Edges in dashed lines are included for clarity, but are not part of the model used for routing [19].

## C. Routing Graph Model

While directed acyclic graphs provide a physical interpretation of a routing fabric, they obscure the relationship between control and the intended result. Previous studies investigated the effect of different control states using the state-dependent graph [36] representation, where each related sub-graph must be generated and evaluated [1]. This is computationally expensive and does not facilitate the design of new routing fabric architectures. To address this problem, we propose to transform the physical graph model into an equivalent graph which explicitly represents the fluid routing possibilities.

**Definition.** *A routing graph $\hat{\mathcal{B}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ consists of a set of $m$ rooted directed binary trees $\{\mathcal{B}_1, \mathcal{B}_1, ..., \mathcal{B}_m\}$, where each $\mathcal{B}_i = (\mathcal{V}_i, \mathcal{E}_i)$ and*

1) *Each $\mathcal{B}_i$ for $1 \leq i \leq m$ corresponds to fluid output $o_m$, and is called a sub-routing graph.*
2) *Each $v_i \in \mathcal{V}_j$ for $1 \leq j \leq m$ represents a transposer-based branching point in the routing fabric reachable by output $o_m$.*
3) *Each $e_i \in \mathcal{E}_j$ for $1 \leq j \leq m$ represents a physical pathway between transposers reachable by output $o_m$.*

The root of each sub-routing graph represents one of the output ports. Input ports are represented by leaf vertices (vertices with no outgoing edges) which are labeled to indicate which input port $i_n$ they flow from. Schematically, we will show input vertices as squares and output vertices as diamonds. All other vertices are labeled with $x_i$ where $i$ corresponds to a transposer and are shown as circles. As we will see, such a representation lends itself to efficient security analysis. The dependencies between states are encoded in the tree structure while the output states are explicitly represented for intuitive interpretation. Derivation of the routing graph from the physical graph can be done with complexity $\mathcal{O}(m \cdot (|V| + |E|))$, using depth-first search or breadth-first search for each of the $m$ outputs. Note that the routing graph model is constructed from the perspective of the output ports and that the directed edges are oriented opposite of fluid flow. This is because of the perspective of the security evaluation in the next section, which is concerned with what input fluid arrives at each output.

Fig. 5(c-f) shows the routing graph models of the 3-to-3 routing fabric. Note that each output is interpreted as a separate rooted binary tree. Isomorphic subgraphs could have been shared among the three inputs, forming a multi-rooted shared decision diagram [37], [38]. However, the focus of this work is not finding the most efficient implementation but rather on easily-understandable analysis, so we keep the trees separated. Also note that the routing graph model can be derived directly from the routing fabric rather than through the physical graph model. We have described the physical graph model for completeness and to set up a framework for the synthesis phase, where we build a routing fabric using routing

TABLE I
PROBABILITY MASS FUNCTIONS FOR THE 3-TO-3 ROUTING FABRIC.

| Variable | $p(y=1)$ | $p(y=2)$ | $p(y=3)$ |
|---|---|---|---|
| $Y_1$ | 3/8 | 3/8 | 1/4 |
| $Y_2$ | 3/8 | 3/8 | 1/4 |
| $Y_3$ | 1/4 | 1/4 | 1/2 |

graphs as building blocks.

### D. Evaluating Security

Given that an attack is probabilistic under the threat model described previously, we would like know which input fluid will likely be routed to each output. Therefore, *evaluating the security of a routing fabric is reduced to computing the probability distributions induced by the routing graph model.* Given the routing graph model $\hat{\mathcal{B}} = \{\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_m\}$ we can calculate the induced probability mass functions (PMFs) $\hat{\mathcal{P}} = \{p(Y_1), p(Y_2), ..., p(Y_m)\}$ as follows. Each sub-routing graph has $n$ different leaf vertex labels and therefore each of the $m$ PMFs has $n$ outcome probabilities $\{p_1, p_2, ..., p_n\}$. Each of the leaves on the tree has a probability $2^{-d}$ where $d$ is the depth of the leaf, since it is assumed that the transposers are i.i.d $Bernoulli(0.5)$. Therefore each of the $p_i$ is found by summing leaf probabilities with the same label. This can be written as a binary expansion:

$$p_i = \sum_{d \in D_i} 2^{-d} \tag{1}$$

where $D_i$ is the set of depths associated with leaf vertices labeled $i$. That is, each each probability is constructed out of "atoms" of the form $2^{-d}$. This bears similarity to the concept of a *generating tree* by Knuth and Yao [39]. However, generating trees simulate arbitrary distributions by decomposing them into dyadic atoms and assigning them to leaves of the tree (here we only evaluate the distributions). This can be interpreted as a procedure for simulating distributions through the use of a single unbiased coin flip.

Referring back to the examples in Fig. 5(c-f), we see that each leaf at depth level 2 has probability $1/4$ while those at depth 3 have probability $1/8$. Summing common leaf vertices, the PMFs are evaluated in Table I. Interpretation of the probability distribution depends on the intent of the designer. We will revisit interpretation and security metrics in Section VI, after describing applications with interesting desired behaviors. We note that Knuth and Yao's generating tree represents a procedure, or an algorithm, for a simulating an arbitrary distribution, while the routing graph in this work is a representation of fluid flow paths. The practical implication of this is that generating trees can be more expressive, as they can have feedback and be infinite. This concept has been extended to show that arbitrary rational distributions can be generated in stochastic flow networks [40].

## V. Routing Fabric Synthesis

In routing fabric analysis, we start with a complete architecture and decompose it into its constituent parts to evaluate the induced probability distribution. In routing fabric synthesis, we proceed in the opposite direction. We start with a routing graph model and assign transposers to each vertex such that it represents a unique architecture. Constructions of routing graphs will be provided in Section VI. For now, we assume they are given. In Fig. 6(a,c) we have two alternate transposer assignments for the same routing graph models. In Fig. 6(a), the root vertices have been assigned different transposers, while in Fig. 6(b), they are the same. While this is a small difference, it leads to considerable changes in the resultant architecture. Fig. 6(b,c) illustrate a synthesized design that uses four transposers, while Fig. 6(e,f) uses only three. The routability of both designs are also drastically different, as Fig. 6(c) allows any input-output permutation while Fig. 6(f) cannot simultaneously route $i_1$ with $i_2$ nor $i_3$ with $i_4$.

Transposer assignment can be represented as a vertex coloring problem. By merging pairs of vertices together, we take advantage of the transposer primitive's ability to route two fluids for one control input. We seek to reduce control complexity by minimizing the number of transposers, while ensuring that the resulting design is physically meaningful.

### A. Problem Statement

The synthesis problem statement is defined as follows:
**Input**: A routing graph model $\hat{\mathcal{B}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$.
**Output**: A synthesized routing graph $(\hat{\mathcal{B}}, c(v))$, where $c(v)$ is a vertex coloring function $v_i \in \hat{\mathcal{V}} \rightarrow \mathbb{Z}^+$.
**Objective**: Minimize the number of transposers.
**Constraints**: Ensure the design is physically realizable.

### B. ILP-Based Synthesis

We propose to solve the transposer assignment problem using integer linear programming (ILP). Such a formulation naturally permits the description of synthesizability constraints in a graph coloring optimization problem. We will use the term color and transposer interchangeably. We define our model as follows. We take a routing graph as input and divide the vertices into three sets: $V_L$ is the set of leaf vertices, $V_R$ is the set of root vertices, and $V_I$ is the set of all other vertices. That is, $V = V_R \cup V_I \cup V_L$. We denote the number of vertices as $N = |V|$, and the maximum number of colors as $K = |V_I|$.

Let $x_{n,k}$ be a binary variable that equals 1 if vertex $v_n$ has been assigned color $k$, for $1 \leq n \leq N$ and $1 \leq k \leq K$. Let $c_k$ be a binary variable that is equal to 1 if color $k$ has been assigned to at least one vertex for $1 \leq k \leq K$. This can be modeled as a logical OR of the $x_{n,k}$ color assignment variable, across all vertices:

$$c_k = \bigvee_{1 \leq n \leq N} x_{n,k}, \quad 1 \leq k \leq K \tag{2}$$

The optimization goal is to reduce the number of transposers, which means assigning 1 to as few of the $c_k$'s as possible. Therefore, we state the objective function as:

$$\min: \sum_{1 \leq k \leq K} c_k \tag{3}$$

Subject to the following constraints:

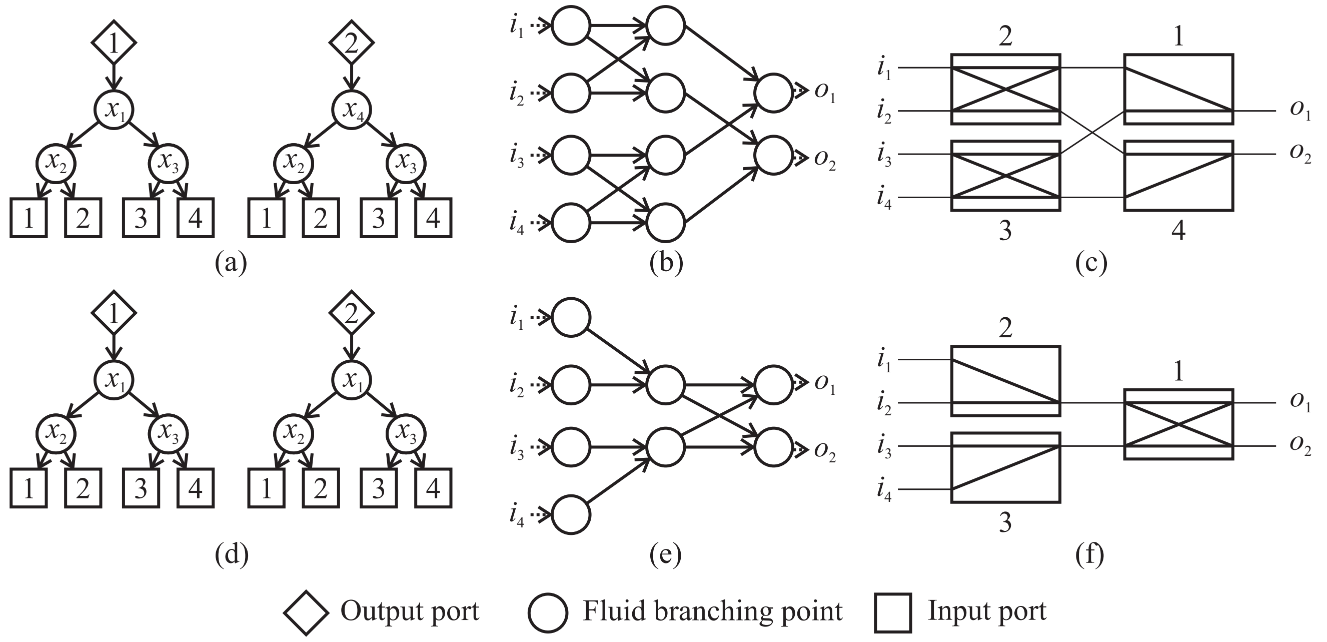


Fig. 6. Routing fabric synthesis. (a) Non-optimal coloring of vertices for two routing graphs. (b) Physical graph model of the non-optimal design. (c) Schematic representation of the synthesized routing fabric. Four control lines are required to drive this design. (d) Optimal coloring. (e) Physical graph model of the optimal design. (f) Schematic representation of synthesized optimal routing fabric. Only three control lines are required. However, this design has less routing flexibility than (c). Saving even a single transposer has practical benefits, as pneumatic control lines add significant bulk and expense to microfluidic systems.

1) *Limit on transposer input ports:* Each transposer has two distinct inputs and as such can only accept two unique fluid flow paths. This means the set of predecessor colors for each transposer color cannot exceed two. Define an additional binary variable $w_{i,j}$ which equals 1 if color $i$ has color $j$ as a predecessor.

$$\sum_{1\leq j\leq K} w_{i,j} \leq 2, \quad 1 \leq i \leq K \tag{4}$$

2) *Limit on transposer output branching paths:* A transposer can toggle an input fluid between two output ports. Both input ports must obey the same restriction. Therefore each color cannot have more than two different colors as children.

$$\sum_{1\leq i\leq K} w_{i,j} \leq 2, \quad 1 \leq j \leq K \tag{5}$$

3) *Unintentional mixing:* Leaf vertices must be driven by only a single transposer. To do otherwise would imply that two independently controlled paths are connected to the same output port, meaning there exists a configuration of transposers that would result in fluids mixing.

$$\sum_{1\leq j\leq K} w_{i,j} \leq 1, \quad \forall i \in V_L \tag{6}$$

This constraint can be removed to include mixing states.

4) *All paths must have an assignment:* Each vertex in the routing graph must be assigned exactly one transposer.

$$\sum_{1\leq k\leq K} x_{n,k} = 1, \quad 1 \leq n \leq N \tag{7}$$

We summarize the notation used in our model in Table II.

TABLE II
NOTATION USED IN ILP FORMULATION.

| Variable | Description |
|---|---|
| $V_R$ | set of root vertices |
| $V_I$ | set of intermediate vertices |
| $V_L$ | set of leaf vertices |
| $K$ | number of possible color assignments |
| $N$ | number of vertices in the routing graph |
| $x_{n,k}$ | a 0-1 variable indicating if vertex $v_n$ is assigned color $k$ |
| $c_k$ | a 0-1 variable indicating if color $k$ has been used |
| $w_{i,j}$ | a 0-1 variable indicating if color $i$ has $j$ as predecessor |

## C. Fast Synthesis

The ILP model is optimal, however, the use of a general-purpose ILP solver may require unacceptable computational time as they explore a search space without exploiting any domain-specific knowledge. Here, we show that the special restricted variant of the synthesis problem as described in the previous section is solvable in polynomial time. The ILP formulation is still useful for precisely specifying the problem, checking for correctness, and accommodating alternate constraints that may or may not yield a polynomial time solution.

We propose a fast synthesis algorithm that proceeds by sequentially evaluating unassigned vertices and deciding whether to add them to the current transposer or start a new transposer (Fig. 7). At the beginning of the routine, we form a new transposer and iterate through unassigned vertices to find a suitable candidate for linking. Linking is performed if a vertex's two children already have a color assignment. At startup, the only vertices that satisfy this property are leaf vertices. After the first assignment, we iterate through the remaining unassigned vertices and see if they can be shared.

```
Input: Routing graph $\hat{\mathcal{B}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$
Output: Map $c(v)$ from vertices to colors
 1: $c(v) \leftarrow \varnothing, \forall v \in \hat{\mathcal{V}}$
 2: $currColor \leftarrow 1$
 3: while $\exists \varnothing \in c(v)$ or new assignments made do
 4:     for each $\{v_i \in \hat{\mathcal{V}} : c(v) = \varnothing\}$ do
 5:         if $v_i$'s children have assigned color then
 6:             $c(v_i) \leftarrow currColor$
 7:             break
 8:         end if
 9:     end for
10:     for each $\{v_i \in \hat{\mathcal{V}} : c(v) = \varnothing\}$ do
11:         if $v_i$'s children colors match currColor's children colors
            then
12:             if $currColor$ already has two parent colors then
13:                 $currColor \leftarrow currColor + 1$
14:                 $c(v_i) \leftarrow currColor$
15:                 break
16:             end if
17:             $c(v_i) \leftarrow currColor$
18:         end if
19:     end for
20:     $currColor \leftarrow currColor + 1$
21: end while
22: if $\{\exists \varnothing \in c(v)\}$ then
23:     throw $InfeasibleSolutionException$
24: end if
25: return $c(v)$
```

Fig. 7. Fast routing graph synthesis pseudocode.

Sharing is performed if a vertex's child vertices have the same colors as the ones already associated with the transposer. Once all vertices have been enumerated, a new transposer is formed. This repeats until all vertices are assigned. Note that an infeasible solution occurs when no new assignments have been made between iterations of the loop and there exist vertices without a color assignment.

**Theorem 1.** *The fast synthesis algorithm solves the transposer assignment problem optimally in $\mathcal{O}(N^2)$.*

*Proof.* We prove optimality in two parts.
*(i)* Assume a routing graph consisting of complete binary trees with colored leaf vertices. Consider any arbitrary set $\mathcal{D}$ of vertices such that the children of these vertices have a color assignment. Let $m$ be the number of colors assigned to $d \in \mathcal{D}$. $m$ is minimized when all vertices $d \in \mathcal{D}$ with the same set of children colors are assigned the same color. This shows how to minimally color a set of vertices.
*(ii)* Let $n$ be that the number of colors assigned to the children of $\mathcal{D}$, and assume that initially $n = n_0$ and $m = m_0$. If we change the color assignments of the child vertices such that $n > n_0$, then $m \geq m_0$. That is, the number of colors used in $\mathcal{D}$ cannot decrease as introducing new child colors will no longer permit sharing of colors in $\mathcal{D}$. For any set of vertices, the number of colors is optimized when the number of children colors are minimized.
Therefore, assigning colors according to *(i)* minimizes the colors for a given set of vertices, and also sets up the minimizing conditions *(ii)* for the parent vertices of this set.

For worst-case complexity, we observe that the algorithm consists of one outer loop and two inner loops. The first inner loop iterates through all unassigned vertices until a suitable candidate is found for adding to the current color, which is $\mathcal{O}(|\hat{\mathcal{V}}|)$. The second inner loop iterates through all unassigned vertices, excluding the one that was just assigned, which is $\mathcal{O}(|\hat{\mathcal{V}}|)$. At each iteration of the outer loop, the worst-case behavior is that only a single color is assigned to each vertex, so this completes in $O(|\hat{\mathcal{V}}|)$ as well. Therefore, the overall worst-case complexity is $\mathcal{O}(|\hat{\mathcal{V}}|^2)$. □

```
Input: Routing graph $\hat{\mathcal{B}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$, map $c(v)$
Output: Physical graph $\mathcal{F}_{n \times m}$
1: for each $\{e = \{s, t\} \in \mathcal{E}\}$ do
2:     $s \leftarrow c(s)$, $t \leftarrow c(s)$
3: end for
4: delete redundant edges in $\mathcal{E}$
5: for each $\{v \in \mathcal{V}$ with two incoming edges$\}$ do
6:     create duplicate of vertex $\hat{v} = v$
7:     redirect one of the incoming edges of $v$ to $\hat{v}$
8: end for
9: return $\mathcal{F}_{n \times m}$ = modified $\hat{\mathcal{B}}$
```

Fig. 8. Routing graph reduction pseudocode.

**Theorem 2.** *The routing fabric synthesis problem is polynomial-time solvable in $\mathcal{O}(|\hat{\mathcal{V}}|^2)$.*

*Proof.* This follows directly from the preceding theorem. □

## *D. Routing Graph Reduction*

After the routing graph has been synthesized, we can convert it into a physical graph model using a reduction algorithm, similar to the reduce process used in binary decision diagrams [41]. Leaf vertices map to unique output ports, so all leaf vertices with the same label must be merged together, with all incoming edges redirected to the merged vertex. Intermediate vertices with the same label and with matching predecessors can be merged together. Note that vertex labels can appear twice in the final physical graph model, as they represent unique input ports on a single transposer. The common label refers to the shared control. This can be performed quickly using the algorithm in Fig. 8 with complexity $\mathcal{O}(|\hat{\mathcal{V}}| + |\hat{\mathcal{E}}|)$. The unique colors in the map $c(v)$ defines vertices in the physical model. We then enumerate the edges in the routing graph and translate them into edges in the physical model based on the color of the edge's vertices. Then we delete redundant edges, and expand vertices with two incoming edges. The transformed graph is a physical graph model. This physical graph model represents a completed routing fabric design with desirable security properties when driven by a serial control line.

## *E. Caveats*

Note that only the specific instance of transposer assignment as described in the ILP model is polynomial-time solvable. Changing the ILP model to accommodate alternate primitives or to permit mixing of fluids may not yield fast solutions. It is an open question as to what effect the ordering of leaf vertices has on optimality of transposer assignment. Furthermore, the synthesis methodology presented here does not specify how the input routing graphs are generated. The routing graphs can satisfy any functionality (within some practical limitations) desired by the system designer, as we will demonstrate in the following design example.

## VI. Design of Tamper-Mitigating Routing Fabrics for Forensic DNA Barcoding

We now turn to the development of some simple constructions to generate routing fabrics with desirable security properties, targeted for a forensic DNA barcoding application.

### A. Forensic DNA Barcoding

Forensic DNA analysis is a major application of microfluidics and is ripe for tampering at the point-of-care [42]. Microfluidic technology has made high-throughput DNA analysis and barcoding a reality, allowing scientists to study gene expression as a function of cell type. Cellular analysis is a widely used procedure in clinical diagnostics, pharmaceutical research, and forensic science [43]. 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 [44]. 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 [45]. Barcoded samples can 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 [19].

### B. Security Implications

The routing fabric utilized in [19] has 6 levels, and permutes 8 inputs to 2 outputs. (Fig. 9(a,b)). 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. 9(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 through the serial control line such that all transposer states happen to be swapped, 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

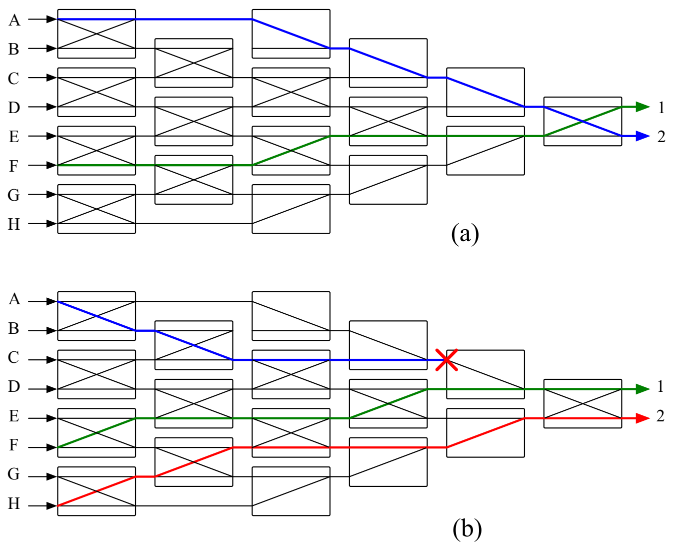


Fig. 9. (a) Routed fabric used in a DNA barcoding application under normal operation. Fluid A (blue) is routed to output port 2 while fluid F (green) is routed to output port 1. (b) After all transposers are attacked, fluid H (red) ends up at output port 2 while fluid F (green) moves to port 1. Fluid A (blue) is blocked at an intermediate transposer.

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.

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. As we can see, an attacker could potentially skew the barcoding process by launching attacks on the control vectors for routing fabrics. To secure the barcoding platform against such attacks, we can use the methodology presented in this work. First, ensure that the mapping between control vectors and control valves cannot be determined by the attacker, and then design the routing fabric in such a way that random perturbations cannot produce a meaningful result.

### C. Tamper-Mitigating Routing Graph Constructions

We demonstrate how our synthesis methodology can be used with routing graphs designed from scratch to secure against actuation tampering attacks. The simple constructions developed here can be adapted for other uses, but more interestingly, the general formulation means that new constructions can be tailored to the application.

In the DNA barcoding application, we would like to prevent the attacker from biasing the barcoding process using the concept of *uncontrollability* as follows (Fig. 10(a)):

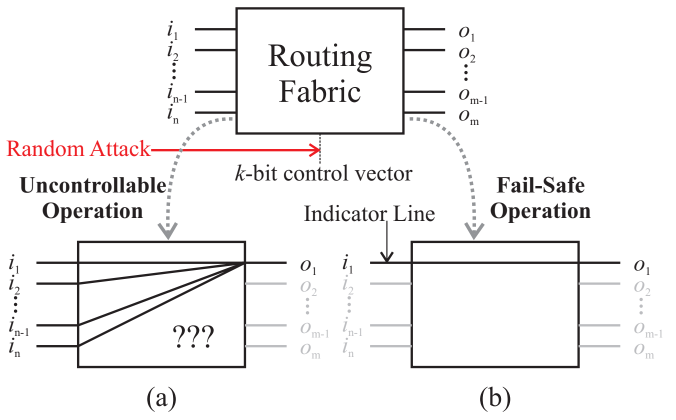


Fig. 10. (a) Uncontrollable operation. Under attack, it is unknown which input fluid will be directed to a particular output. (b) Fail-safe operation. One routing is more likely than all others, and can be connected to an "indicator line" where some inert low-cost fluid is easily detected at the output.

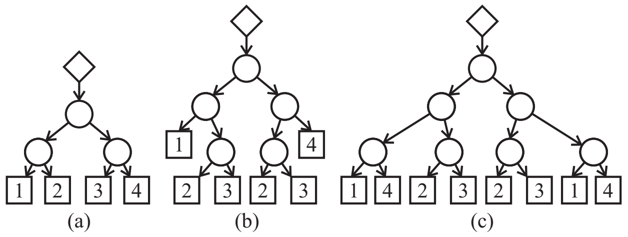


Fig. 11. (a) Optimal balanced tree construction for $n = 4$ outcomes. (b) $r$-1 expansion of the optimal tree. (c) $r$-2 expansion of the optimal tree. Expansions can continue indefinitely by selecting two leaf vertices of the same depth and converting them into branching vertices with two children with the same labels.

**Definition 1.** *A routing fabric is* ***uncontrollable*** *if an attacker is unable to reliably route a specific input fluid to specific output port.*

In other words, the PMF $p(Y_i)$ is a uniform distribution. We extend the definition of uniform distribution to mean that the PMF will either take on uniform values or 0. This is to accommodate fabric designs where certain outcomes are not possible.

**Security Metric: Normalized Entropy.** The information-theoretic concept of entropy naturally captures the security of the distribution:

$$\hat{H}(Y_n) = -\sum_{y_i \in S} \frac{p(y_i)\log_2(p(y_i))}{\log_2(|S|)} \tag{8}$$

where $S$ as the set of all desired outcomes and the $\log_2$ term expresses the entropy in units of bits. By "desired outcomes" we mean to exclude inputs that we may wish to behave in a biased manner, as we will see when we introduce the concept of indicator lines. It is well-known that for discrete distributions, entropy is maximized when the distribution is uniform [46]. We use the normalized form of entropy so that the quantity can be compared across routing fabrics irrespective of the number of input ports $n$.

**Construction:** If we assume that $n$ is a power of 2 (i.e., the number of input ports is dyadic), then a balanced binary tree of depth $\log(n)$ implements a uniform distribution with the minimum number of vertices. However, it may be desirable to realize a tree of greater depth in case the synthesis algorithm returns an infeasible solution. We define a parameter $r$ as the redundancy of an uncontrollable routing graph, which increases the number of branching points while keeping the induced probability distribution constant. We construct an $r$-redundant uncontrollable routing graph by duplicating the $r$-1 routing graph and connecting the two subgraphs with one branching point (Fig. 11).

As a secondary countermeasure, we would also like to detect if an attack has occurred. This can be achieved by using an indicator fluid. In normal operation, the microfluidic platform will never route the indicator fluid to any of the output ports. Under attack, the routing fabric will dispense indicator fluid with high probability, which can easily be detected by on-board sensors. An indicator fluid can be as simple as water with dye. Detection can be implemented with a *fail-safe* routing fabric (Fig. 10(b)).

**Definition 2.** *A routing fabric is* ***fail-safe*** *if, under a random control vector, one fluid routing is more likely than all others.*

**Security Metric: Relative Likelihood.** We introduce factor $\epsilon_n$ as the ratio between the most probable outcome and the second most probable outcome corresponding to $Y_n$. Let $p_m$ be the $m$-th highest probability defined by the probability mass function (PMF) $p(Y_n)$. Then,

$$\epsilon_n = \frac{p_1(Y_n)}{p_2(Y_n)} \tag{9}$$

A PMF with a uniform distribution will thus have $\epsilon = 1$, while on the other extreme, a $Bernoulli(1)$ distribution will have $\epsilon = \infty$. It is desirable to maximize this quantity.

**Construction:** The requirement that one outcome be much more likely than all other outcomes places little restriction on the distribution of the unlikely outcomes. Therefore, instead of attempting to provide an optimal construction, we simply provide a simple method for augmenting an existing routing graph to become fail-safe: define a new root vertex, and connect the old root vertex to one of outgoing edges, and connect the fail-safe outcome to the other outgoing edge. The relative likelihood is thus $\epsilon \geq 2$, since all of the original outcome probabilities are now bounded by $1/2$ (Fig. 12). Intuitively, connecting an extra transposer in this manner extends the state space such that when this extra transposer is active, the routing fabric functions normally, and when it is inactive, the routing fabric is diverted to an indicator line.

We then propose an iterative design procedure, where we start with an minimal construction and then perform $r$-expansion until a feasible solution is found (Fig. 13).

## D. Experimental Results

Using the constructions derived here, we generated two identical sub-routing graphs. First, we generate an uncontrollable routing graph to prevent biasing the droplet barcodes towards any particular label. Then we apply the augmentation operation to provide fail-safe functionality. The construction is shown in Fig. 14(a).

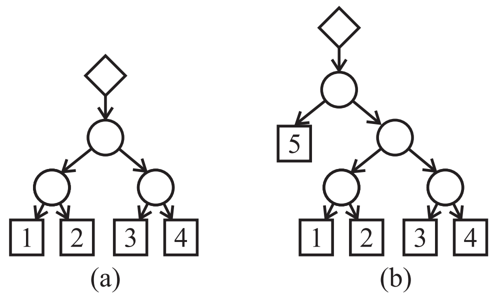


Fig. 12. (a) Original routing graph taken as input. (b) Augmented routing graph where a new fail-safe state 5 has been introduced. Root-level augmentation guarantees that $\epsilon \geq 2$.

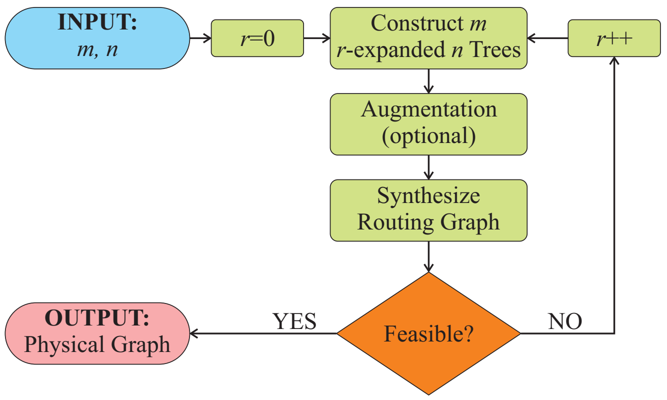


Fig. 13. A simple iterative procedure to generate $n - to - m$ routing fabrics with uniform distributions and optionally, a fail-safe mode. Start with the minimal construction achieving the design goals, and incrementally $r$-expand the routing graphs until a feasible solution is found.

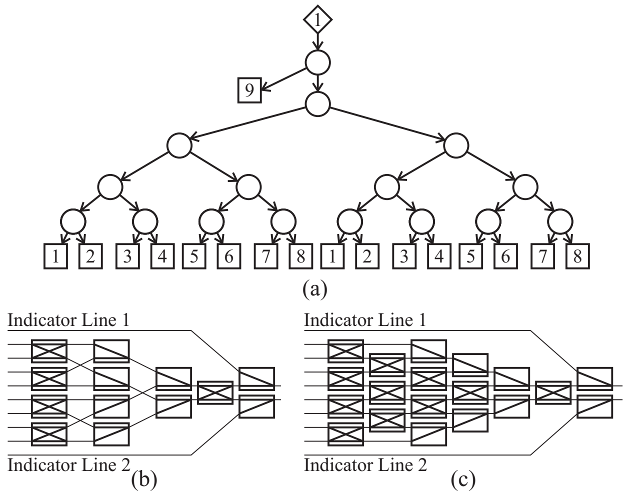


Fig. 14. (a) Routing graph construction. Two identical graphs are used as input to the synthesis routine. (b) Result from synthesis. The tamper-mitigating routing fabric achieves a controlled state distribution, with the same routing flexibility, while using fewer transposers. (c) Original 8-to-2 routing fabric design as described in [47].

TABLE III
PROBABILITY MASS FUNCTIONS AND SECURITY METRICS FOR 8-TO-2 ROUTING FABRICS.

| | Augmented CoSyn [19] | | This Work | |
|---|---|---|---|---|
| $n$ | $p(Y_1 = n)$ | $p(Y_2 = n)$ | $p(Y_1 = n)$ | $p(Y_2 = n)$ |
| 1 | 0.1875 | 0.1875 | 0.0625 | 0.0625 |
| 2 | 0.1875 | 0.1875 | 0.0625 | 0.0625 |
| 3 | 0.1563 | 0.1563 | 0.0625 | 0.0625 |
| 4 | 0.1563 | 0.1563 | 0.0625 | 0.0625 |
| 5 | 0.0938 | 0.0938 | 0.0625 | 0.0625 |
| 6 | 0.0938 | 0.0938 | 0.0625 | 0.0625 |
| 7 | 0.0625 | 0.0625 | 0.0625 | 0.0625 |
| 8 | 0.0625 | 0.0625 | 0.0625 | 0.0625 |
| 9 | 0.5000 | 0.0000 | 0.5000 | 0.0000 |
| 10 | 0.0000 | 0.5000 | 0.0000 | 0.5000 |
| $\hat{H}(Y_n)$ | 0.48 | 0.48 | 1.00 | 1.00 |
| $\epsilon_n$ | 5.33 | 5.33 | 8.00 | 8.00 |

The result of applying either the ILP-based or heuristic synthesis algorithms on these routing graph constructions is shown in Fig. 14(b). By design, the probability distribution induced by this routing fabric is well-defined. The only downside to this design is that pipelining operations are not defined. The induced probability mass functions are shown in Table III. We see that this work is both fail-safe and uncontrollable, while the original routing fabric is ambiguous as to what guarantees it can provide. We evaluate relative likelihood $\epsilon = 8$ and $\epsilon = 5.33$ for this work and the augmented CoSyn fabric, respectively. Ignoring the fail-safe states induced by the indicator lines, we evaluate normalized entropy as $\hat{H} = 1$ and $\hat{H} = 0.48$, showing that the synthesized fabric maximizes uncertainty for the attacker.

What this means for an attacker is that with probability equal to a fair coin flip, any attempt to control the routing fabric will result in their attack being detected. If they happen to succeed in activating the fail-safe mode, then they can only select a droplet barcode with uniform probability. That is, they will be unable to bias the outcome through repeated attacks. Furthermore, repeated attacks will exponentially increase the likelihood of inducing the failure more. In comparison with the original 8-to-2 routing fabric, the attacker would be able to bias the barcoding operation toward barcodes located at ports 1 and 2. This will introduce false biases into the cell study, leading to incorrect results.

## VII. Relation to Prior Work

This work is the first to address both the analysis and synthesis of secure microfluidic routing fabrics. However, our concepts share similarities with many prior works, which we mention here for the interested reader.

- *Graphical Models*: Algebraic decision diagrams [38] and their precursors, ordered binary decision diagrams [41], have lent themselves to efficient manipulation of boolean functions and as such have had a tremendous impact on the CAD industry since their introduction in the late 70s. This work does not leverage features that have become synonymous with BDDs, such as canonicity through graph reduction and ordering of variables.
- *Probabilistic Graphical Models*: The physical graph model could be interpreted as a Bayesian network, and the act of determining most probable outcomes as Bayesian inference [48]. However, the analysis of transposers is slightly more complicated due to the in-

terdependence of fluid routing paths and their control; physical graph models contain repeated vertices.

- *Scattering-based Analysis*: [1] analyzed security performance in terms of the probability of scattering, where scattering is defined as a fluid being redirected more than $d$ coordinate spaces away from its desired destination. This was also evaluated under the assumption of a certain attack class, organized by the number of bit flips induced by the attack. Such a model presumed the attacker was limited in attack capability while still being forced to randomly choose control lines to tamper with. Here, we argue that such a limitation is unrealistic in practice and that introducing randomness leads to more feasible analysis and design.
- *Fault-Tolerant Design*: This work differs from general fault-tolerant architecture research in several important ways [49]. First, there is no known transposer failure mode that manifests itself as a flipped state; doing so would require the simultaneous transient failure of several valves [35]. Only an attacker is capable of inducing such a state. Second, while many fault-tolerant systems incorporate redundancy that can be leveraged in case of failure, many of these systems require the detection of a fault and active redirection of resources. During an attack, no such advantage may be conferred since the attacker may be actively tampering with these systems.
- *Interconnection Networks*: And lastly, we note that the concept of a microfluidic routing fabric has historical precedent in interconnection networks. The study of interconnection networks began with the introduction of the Clos network [50], which was designed for telephony applications. In Clos' original paper, conditions were proved for ensuring that the network is non-blocking. Other interconnection toplogies include Banyan and Omega networks [51]. Later, these concepts were adopted for networking and computer architectures, and analyzed for their fault-tolerant behavior. The routing fabrics considered in this work are of a mason-brick topology and are not employed in these related fields.

## VIII. Conclusions

We presented a security assessment of an emerging microfluidic hardware primitive: the transposer-based routing fabric driven through a serial control line. We then formulated an analysis and design methodology under a random control vector attack model. Two classes of security, fail-safe and uncontrollable, were defined and case studies were presented to show how such characteristics could be leveraged in practice as a tamper mitigation mechanism. Such a design-time tamper mitigation technique eliminates one of the simplest methods for tampering with a physically vulnerable device employed at the point-of-care.

One major limitation of this work is the restriction on distributions being dyadic. As shown in recent works, feedback greatly increases the expressivity of flow networks, permitting arbitrary rational distributions in compact form [40]. Without feedback, an infinite number of branching points are required [39]. As feedback is prohibited in microfluidic flow networks, it would be interesting work to see if arbitrary distributions could be efficiently approximated for implementation in a routing fabric. Other limitations include: the fact that pipelining cannot be specified as a design criterion [19], and the number of transposers in the final design is not guaranteed to be minimized.

If we were to consider DoS attacks, then the performance of the routing fabrics in this work would be degraded. For instance, an attacker could perturb only a single transposer in the hope of causing a failure. The fail-safe mechanism would then be activated with probability $1/K$ instead of $1/2$, where $K$ is the number of transposers. Evaluating the probability that a fluid input would deviate from its correct path must take the current transposer state into account and would thus require scattering-based analysis as presented in [1].

Future work could address the problem of designing a tamper-mitigating routing fabric with security guarantees at the *joint* probability distribution level. This is especially challenging, given that the joint probability distributions we consider are actually functions of probability distributions defined through multiple graphical models. We also note that our work is a *passive* mitigation technique, as it requires no active input or control to work. Alternate mitigation techniques could use active feedback to correct anomolous behavior. New microfluidic routing primitives could be investigated; it was noted that in [11] that multiplexer-demultiplexer pairs could have been implemented instead of the transposer primitive.

Finally, we note that this work is among one of the first hardware architecture-based defenses against tampering in microfluidics. The intent is not to replace well-designed cyberphysical systems implementing standard security measures such as encryption. Rather, this work aims to bolster security in the physical domain, which is often overlooked.

## References

[1] J. Tang, M. Ibrahim, K. Chakrabarty, and R. Karri, "Security trade-offs in microfluidic routing fabrics," in *Proc. IEEE Int. Conf. Comput. Des.*, Newton, MA, Nov. 2017, pp. 25–32.

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

[3] D. Mark, S. Haeberle, G. Roth, F. Von Stetten, and R. Zengerle, "Microfluidic lab-on-a-chip platforms: requirements, characteristics and applications," in *Microfluidics Based Microsystems*. Dordrecht, Netherlands: Springer, 2010, pp. 305–376.

[4] M. A. Unger, H.-P. Chou, T. Thorsen, A. Scherer, and S. R. Quake, "Monolithic microfabricated valves and pumps by multilayer soft lithography," *Science*, vol. 288, no. 5463, pp. 113–116, 2000.

[5] J. Melin and S. R. Quake, "Microfluidic large-scale integration: the evolution of design rules for biological automation," *Annu. Rev. Biophys. Biomol. Struct.*, vol. 36, pp. 213–231, 2007.

[6] L. R. Volpatti and A. K. Yetisen, "Commercialization of microfluidic devices," *Trends Biotechnol.*, vol. 32, no. 7, pp. 347–350, 2014.

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

[8] S. Bhattacharjee, S. Poddar, S. Roy, J.-D. Huang, and B. B. Bhattacharya, "Dilution and mixing algorithms for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 4, pp. 614–627, 2017.

[9] K. Hu, T. A. Dinh, T.-Y. Ho, and K. Chakrabarty, "Control-layer routing and control-pin minimization for flow-based microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 1, pp. 55–68, 2017.

[10] K. Hu, F. Yu, T.-Y. Ho, and K. Chakrabarty, "Testing of flow-based microfluidic biochips: fault modeling, test generation, and experimental demonstration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 10, pp. 1463–1475, 2014.

[11] R. Silva, S. Bhatia, and D. Densmore, "A reconfigurable continuous-flow fluidic routing fabric using a modular, scalable primitive," *Lab. Chip*, vol. 16, no. 14, pp. 2730–2741, 2016.

[12] L. M. Fidalgo and S. J. Maerkl, "A software-programmable microfluidic device for automated biology," *Lab. Chip*, vol. 11, no. 9, pp. 1612–1619, 2011.

[13] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. Sastry, "Challenges for securing cyber physical systems," in *Proc. Workshop Future Dir. Cyber-physical Syst. Security*, Newark, NJ, Jul. 2009, p. 5.

[14] J. Giraldo, E. Sarkar, A. Cardenas, M. Maniatakos, and M. Kantarcioglu, "Security and privacy in cyber-physical systems: A survey of surveys," *IEEE Design & Test*, 2017.

[15] S. S. Ali, M. Ibrahim, J. Rajendran, O. Sinanoglu, and K. Chakrabarty, "Supply-chain security of digital microfluidic biochips," *Computer*, vol. 49, no. 8, pp. 36–43, 2016.

[16] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[17] S. S. Ali, M. Ibrahim, O. Sinanoglu, K. Chakrabarty, and R. Karri, "Security assessment of cyberphysical digital microfluidic biochips," *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 13, no. 3, pp. 445–458, 2016.

[18] R. Garver and C. Seife. (2013, Apr.) FDA Let Drugs Approved on Fraudulent Research Stay on the Market. [Online]. Available: https://www.propublica.org/article/fda-let-drugs-approved-on-fraudulent-research-stay-on-the-market

[19] M. Ibrahim, K. Chakrabarty, and U. Schlichtmann, "CoSyn: efficient single-cell analysis using a hybrid microfluidic platform," in *Proc. Conf. Des. Autom. Test Europe*, Lausanne, Switzerland, Mar. 2017.

[20] S. Potluri, A. Schneider, P. Pop, J. Madsen *et al.*, "Synthesis of on-chip control circuits for mVLSI biochips," in *Proc. Conf. Des. Autom. Test Europe*, 2017, pp. 1799–1804.

[21] J. Tang, M. Ibrahim, K. Chakrabarty, and R. Karri, "Security implications of cyberphysical flow-based microfluidic biochips," in *Proc. IEEE Asian Test Symp.*, Taipei, Taiwan, 2017, pp. 110–115.

[22] G. J. Kost, "Preventing medical errors in point-of-care testing: security, validation, performance, safeguards, and connectivity," *Arch. Pathol. Lab. Med.*, vol. 125, no. 10, pp. 1307–1315, 2001.

[23] I. E. Araci and P. Brisk, "Recent developments in microfluidic large scale integration," *Current Opinion in Biotechnology*, vol. 25, pp. 60–68, 2014.

[24] S.-J. Kim, D. Lai, J. Y. Park, R. Yokokawa, and S. Takayama, "Microfluidic automation using elastomeric valves and droplets: reducing reliance on external controllers," *small*, vol. 8, no. 19, pp. 2925–2934, 2012.

[25] B. Mosadegh, T. Bersano-Begey, J. Y. Park, M. A. Burns, and S. Takayama, "Next-generation integrated microfluidic circuits," *Lab. Chip*, vol. 11, no. 17, pp. 2813–2818, 2011.

[26] P. N. Duncan, S. Ahrar, and E. E. Hui, "Scaling of pneumatic digital logic circuits," *Lab. Chip*, vol. 15, no. 5, pp. 1360–1365, 2015.

[27] M. Rhee and M. A. Burns, "Microfluidic pneumatic logic circuits and digital pneumatic microprocessors for integrated microfluidic systems," *Lab. Chip*, vol. 9, no. 21, pp. 3131–3143, 2009.

[28] M. Ibrahim, A. Sridhar, K. Chakrabarty, and U. Schlichtmann, "Sortex: Efficient timing-driven synthesis of reconfigurable flow-based biochips for scalable single-cell screening," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2017, pp. 623–630.

[29] H. Fereidooni, J. Classen, T. Spink, P. Patras, M. Miettinen, A.-R. Sadeghi, M. Hollick, and M. Conti, "Breaking fitness records without moving: Reverse engineering and spoofing fitbit," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2017, pp. 48–69.

[30] D. G. Abraham, G. M. Dolan, G. P. Double, and J. V. Stevens, "Transaction security system," *IBM Systems Journal*, vol. 30, no. 2, pp. 206–229, 1991.

[31] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: theory, practice, and countermeasures," *Proc. IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[32] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, "The sorcerer's apprentice guide to fault attacks," *Proc. IEEE*, vol. 94, no. 2, pp. 370–382, 2006.

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

[34] H. Chen, S. Potluri, and F. Koushanfar, "Biochipwork: Reverse engineering of microfluidic biochips," in *Proc. IEEE Int. Conf. Comput. Des.*, Newton, MA, Nov. 2017, pp. 9–16.

[35] Y. Moradi, M. Ibrahim, K. Chakrabarty, and U. Schlichtmann, "Fault-tolerant valve-based microfluidic routing fabric for droplet barcoding in single-cell analysis," in *Proc. Conf. Des. Autom. Test Europe*, 2018.

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

[37] S.-i. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient boolean function manipulation," in *Proc. IEEE/ACM Des. Autom. Conf.*, 1990, pp. 52–57.

[38] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Algebric decision diagrams and their applications," *Formal methods in system design*, vol. 10, no. 2-3, pp. 171–206, 1997.

[39] D. E. Knuth and A. C. Yao, "The complexity of non-uniform random number generation," in *Algorithms and Complexity: New Directions and Recent Results*, J. F. Traub, Ed. New York, NY: Academic Press, Inc., 1976, pp. 357–428.

[40] H. Zhou, H.-L. Chen, and J. Bruck, "Synthesis of stochastic flow networks," *IEEE Trans. Comput.*, vol. 63, no. 5, pp. 1234–1247, 2014.

[41] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Comput.*, vol. 100, no. 8, pp. 677–691, 1986.

[42] K. M. Horsman, J. M. Bienvenue, K. R. Blasier, and J. P. Landers, "Forensic dna analysis on microfluidic devices: a review," *J. Forensic Sci.*, vol. 52, no. 4, pp. 784–799, 2007.

[43] J. El-Ali, P. K. Sorger, and K. F. Jensen, "Cells on chips," *Nature*, vol. 442, no. 7101, pp. 403–411, 2006.

[44] S. Hosic, S. K. Murthy, and A. N. Koppes, "Microfluidic sample preparation for single cell analysis," *Anal. Chem.*, vol. 88, no. 1, pp. 354–380, 2015.

[45] A. M. Klein, L. Mazutis, I. Akartuna, N. Tallapragada, A. Veres, V. Li, L. Peshkin, D. A. Weitz, and M. W. Kirschner, "Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells," *Cell*, vol. 161, no. 5, pp. 1187–1201, 2015.

[46] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 2012.

[47] M. Ibrahim, K. Chakrabarty, and K. Scott, "Synthesis of cyberphysical digital-microfluidic biochips for real-time quantitative analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 5, pp. 733–746, 2017.

[48] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[49] L. Xing and S. V. Amari, *Binary Decision Diagrams and Extensions for System Reliability Analysis*. Beverly, MA: Scrivener Publishing, 2015.

[50] C. Clos, "A study of non-blocking switching networks," *Bell Labs Technical Journal*, vol. 32, no. 2, pp. 406–424, 1953.

[51] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection Networks: An Engineering Approach*. San Francisco, CA: Morgan Kaufmann Publishers, 2003.