IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 26 July 2022, accepted 18 August 2022, date of publication 5 September 2022, date of current version 19 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3204397

RESEARCH ARTICLE

Optimized Sequence Dataset Generation for a Two-Stage Pipelined Microcontroller Instruction-Level Electromagnetic Information Leakage Analysis

SHIH-YI YUAN, (Member, IEEE)
Department of Communication Engineering, School of Information and Electrical Engineering, Feng Chia University, Taichung 407, Taiwan
e-mail: syyuan@fcu.edu.tw

This work was supported in part by the Bureau of Standards, Metrology and Inspection, MOEA (BSMI) Project R-111038; and in part by the Electronics Testing Center (ETC), Taiwan.

ABSTRACT With the prevalence of the Internet of Things (IoT) and microcontrollers (MCUs), the security issues of IoT and MCUs have become increasingly important. Side-channel analysis (SCA) is a major threat to such problems. One of the non-invasive SCAs is through electromagnetic information leakage (EM-leak) analysis. The author has developed a machine-instruction-level EM-leak analysis by neural network (NN) model. The NN model analysis needs a large dataset for training and validation. And the dataset should be *complete* and *sufficient*. The dataset sufficiency can be achieved by acquiring more data from the already proposed EM-leak measurement platform. However, the completeness issue becomes a challenge due to the pipelined architecture in the target MCU. In this paper, the completeness issue of a NN model dataset for the EM-leak analysis is addressed. Experiments show that the proposed algorithm can find an optimal solution. The contributions of this paper include: it successfully reduces a complex and practical SCA problem into a two-stage pipelined MCU individual EM-leak analysis sequence (2-EMAseq) description, proves that the 2-EMAseq problem has at least one optimal solution, uses this proof to develop an algorithm, and uses this algorithm to find a complete dataset is the target two-stage pipelined MCU dataset for NN model training/validation. Currently, the algorithm can only generate the optimal EM-dataset for two-stage pipelined MCUs. However, there are many MCUs with 4∼8 pipeline stages. Whether there is an optimal solution when the stages are more than two is still an open question. And it needs to find proofs or derive heuristics in the future for such MCUs.

INDEX TERMS EM information leakage, NN dataset generation, directed graph Eulerian trail.

I. INTRODUCTION

Reverse engineering could be used to exploit vulnerabilities of digital systems. It has also become a critical security concern [2] for embedded systems and the Internet of Things (IoT). IoT envisions billions of wireless-connected end-nodes [3] and is used in many applications such as surveillance, health monitoring, agriculture, and robotics, among others. The microcontroller (MCU) is the heart of IoT. The primary task of an MCU is to perform predefined behaviors according to internal programs to acquire environment input or to provide output to networks or other devices. An MCU is a single-chip programmable sequential digital electronic component with integrated peripherals. The MCU is pervasive [1], [3], [4] in the contemporary information age because of the small size with many input/output peripherals.

An MCU program is a carefully designed sequence of machine instructions. Different instructions (load, store, addition, multiplication, comparison, jumps, etc.) control the on/off states of the sub-functional blocks (network, IO interface, bus arbitration, etc.) within the MCU and affect the timing, power consumption, and electromagnetic (EM) radiation of these blocks. The observable EM behaviors are called the

The associate editor coordinating the review of this manuscript and approving it for publication was Su Yan.

side-effects of the program. Although the side-effect signals are subtle, the internal behaviors of an MCU can be detected by the side-effect signal analysis. If these behaviors could be analyzed for reverse engineering, the security issues of these pervasive MCUs become important.

Some electronic attacks are based on the MCU's side effects. They are called side-channel attacks [1], [5]. The secret information of a victim program executed inside an MCU can be exposed by such attacks [5]. These side-effects may include system execution time, transient power consumption, or EM radiation of an MCU. The analysis of these effects may increase the security concerns that stem from such attacks.

In [5], Yuan designed a platform to analyze the EM radiation side-channel information leakage (EM-leak) signals via a neural network (NN) model (FIGURE 1). In [5], more advanced models such as LSTM [6] or ResNet [7] are not used but a fully connected NN (FCNN) model is adapted. The reason is that the author has tried many advanced NN models with different dataset types for SCA identification. When trading off among different factors (such as the training cost, overfitting, training time, and dataset sizes), the FCNN is determined. In FIGURE 1, the EM-leak within the collective EM-leak (cEM-leak) guarded by two sentential signals (begin/end-guard-signal) is called the individual EM-leak (iEM-leak). The iEM-leak corresponds to an individual machine instruction EM-leak of an MCU. Through the measurement and NN analysis of [5], the program execution sequence can be extracted from the EM-leak analysis.

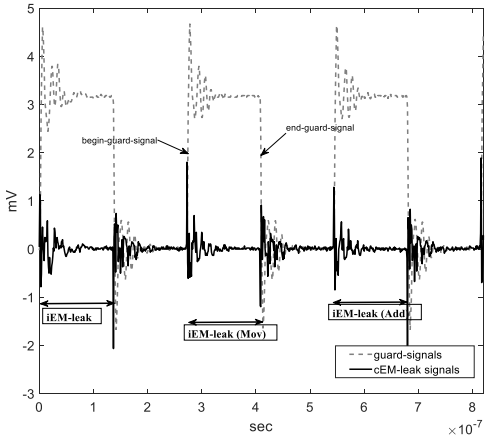


FIGURE 1. Program context within a testing program.

However, the NN models require a lot of data for NN training and validation. And these data, which are called EM-dataset in this paper, should be measured by a special EM-leak measurement platform [5].

There are many NN datasets, from small dataset (like MNIST [6] or CIFAR-10 [7]) to large dataset (like ImageNet [8], WebFace [9], or VoxPopuli [10]). An NN dataset is a mapping or labeling of finite samples from an interested research field. And the fields generally contain infinite samples. Taking the MNIST [6] dataset as an example, it is a database of handwritten digits that is used for image processing systems. The database contains only 60,000 training images and 10,000 testing images. It is clear that the handwriting digits of all people cannot be complete for the undocumented or the unborn children. And the MNIST is just a small group of handwriting samples. Generally, the completeness of these datasets is not important because NN models targeted to their problems are open questions and cannot be fully elaborated.

However, it is not the case in the proposed EM-dataset. The dataset is finite and should be complete because the DUT's SCA should be identified by the NN model even to the least possible corner cases [11]. Currently, this paper proposes a new problem to be solved due to the dataset completeness is not met by other types of dataset.

Thus, generating a ''sufficient and complete'' EM-dataset is very important for such studies. Here, sufficiency means the dataset size should be large enough for the training and validation of the NN models. And the completeness means the dataset should cover all instruction sequence cases. The sufficiency issue can be covered by iterating the measurement to increase the dataset sizes. However, completeness cannot be guaranteed by the iterations. The reason is described below.

Although [5] can identify the EM-leak to some extent, the dataset for the NN training/validation is far from completeness. And only partial instruction can be identified in [5]. The reason is detailed below.

If an MCU has n different instructions, all these instructions can be selected into a testing program at any sequence. The testing program is then programmed (burnt) into a target MCU (the device under test, DUT). Here, the verb ''programmed (burnt)'' means an action to download the compiled testing program into the DUT's memory (flash or RAM). After the programming, the DUT is powered up and the collective EM-leak signals (cEM-leak) can be measured. Some digital signal post-processing techniques [5] can be applied to the cEM-leak signals and the iEM-leak target to each instruction can be acquired. These iEM-leaks with their corresponding instruction labels can be collected into the EM-dataset. Since all of the n instructions' iEM-leaks can be measured and stored in the EM-dataset, the completeness is guaranteed.

A possible EM-dataset collection and EM-leak analysis of an unknown program procedure may have the following steps.

1) A ''program context'' (FIGURE 2) within a testing program is built. All the n instructions of a DUT are selected into the program context for EM-leak feature extraction. The sequence of these instructions can be randomly assigned. If the instructions are jump or conditional jumps, they should be treated carefully. Two 'guarded-signals' are inserted at the beginning and end of the program context. An empty delay loop is inserted at the bottom of the end-guard-signal. These blocks constitute the whole testing program. After the

fixed time delay, the testing program executes from the beginning repeatedly.

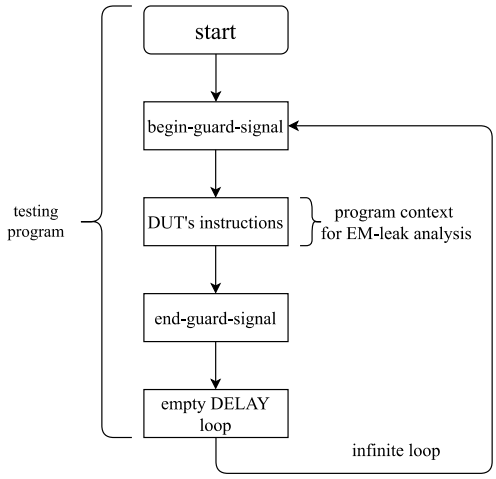


FIGURE 2. Side-channel EM-leak (iEM-leak separated from the cEM-leak) and the guard-signals.

2) The testing program is compiled into instructions (machine codes). And these instructions are programmed into the target DUT.
3) The DUT is powered up and the cEM-leak is repeatedly measured by EM-leak measurement setups FIGURE 4).
4) After the cEM-leak is acquired, every iEM-leak can be divided according to FIGURE 1. Every iEM-leak represents its own individual instruction's EM-leak behavior. These iEM-leaks (with the corresponding instruction labels) are collected as the EM-dataset. The details are described in [5].
5) After the dataset is acquired, the NN model can be trained and validated to identify the relationships between the iEM-leaks and their corresponding instructions.
6) Given any unknown program executed in the DUT, by measuring its cEM-leaks and mapping each EM-leak into the corresponding instruction through the trained NN model, the execution sequence of the given program can be restored. This means the context of any program can be identified by the EM-leak.

It should be noticed that the iEM-leak acquisition procedures are tedious and time-consuming. Generally, the number of instructions of an MCU is fixed and rather small (43 for Intel 8051 MCU and 30∼400 for different RISC-type MCUs). It is estimated that an iEM-leak signal needs an average of 0.5 seconds to be acquired from the measurement platform [5] into the EM-dataset. And a dataset needs at least n*1000 iEM-leak samples for sufficient NN training and validation.

It seems easy to acquire an EM-dataset. If the conceptual EM-dataset collection procedure above is feasible and assuming a DUT has only 100 instructions (n=100), the EM-dataset collection procedure may take only about

inst.	execution cycle						
	0	1	2	3	4	5	6
A	P1	P2					
A		P1	P2				
A			P1	P2			
B				P1	P2		
A					P1	P2	
C						P1	P2
iEM-leak	-	AA	AA	AB	BA	AC	...

FIGURE 3. 2-stage Pipelined instruction cycle vs. iEM-leak signals (yellow: naïvely expected, gray: accompany generated). PX: pipeline stage X.

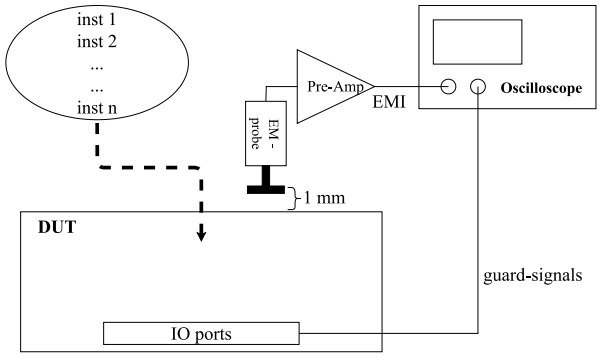


FIGURE 4. EM-leak measurement platform (adapted from [5]).

13.9 hours (100*1000*0.5 seconds) to generate a suitable EM-dataset.

However, the completeness of an EM-dataset cannot be generated by such a naïve procedure. The reason is that modern MCUs contain pipeline designs [12]. A pipelined MCU design can increase the MCU's throughput, clock frequency, and thus, the overall MCU performance. However, this makes the EM-dataset generation in [5] to be very difficult.

For the analysis simplicity, assuming an MCU contains only 3 instructions A, B, and C; and assuming the MCU has a 2-stage pipeline (stage 1 and stage 2), it means that the MCU contains 2 consecutive instructions inside it to control an iEM-leak signal in one execution cycle. Thus, there are a total $3^2 = 9$ different pairs of iEM-leak patterns. They are A_1A_2, A_1B_2, A_1C_2, B_1A_2, …, C_1B_2, and C_1C_2. Without loose of clearance, the combinations are abbreviated to AA, AB, AC, BA,…, CB, and CC. If the program context is designed as AA AB AC BA BB …CC, the program needs total 9*2 = 18 instructions. However, since the instructions are pipelinedly executed, the first three pairs of instructions (six instructions) selected could be AA BA CA (FIGURE 3, orange). The six instructions not only generate the expected AA, BA, and CA iEM-leaks (FIGURE 3, yellow) but are also accompanied by an extra AA and new BA iEM-leaks are generated (FIGURE 3, gray). Because the BA iEM-leak is generated elsewhere, the dataset contains the extra AA and BA iEM-leak makes the EM-dataset unbalanced [13]. Thus, a more efficient instruction sequence may be re-arranged as A A B A C B B C C (and loopback). It can be verified that all the iEM-leaks are generated in the new sequence.

Not only the completeness is guaranteed but also the dataset is unbalanced. Since all the iEM-leaks can be acquired from the new sequence, the total instructions within the program context are reduced from 18 to 9 instructions.

The DUT used in this paper is dsPIC33EP64MC202 [16]. It is a mature and popular 16-bit 2-stage pipelined RISC-type MCU produced by Microchip Technology. The total number of instructions is 242 (not considering the DSP instructions). If the EM-dataset generation is based on the naïve concept, it needs a total of 0.93 years ($242^2 * 1000 * 0.5$ seconds $=$ 0.9279 year)! Not only the generation takes a long time, but the dataset is unbalanced [13] depending on the instruction sequence order.

Thus, an efficient and balanced EM-dataset generation algorithm with an optimal or nearly optimal complete instruction sequence for the NN model program context is very important. If a near-optimal instruction sequence is developed, not only the dataset completeness can be guaranteed but the collection time and the dataset unbalance can be both reduced. Thus, the EM-dataset generation can be sufficient and complete if the instruction sequence within the program context is properly arranged.

In this paper, the author tries to find an optimal instruction sequencing algorithm to increase the dataset generation efficiency. The paper is organized as follows. In section II, the paper tries to define the optimality of the instruction sequence of the program context. After the definition, the author tries to prove that the optimal solution exists and, eventually, proposes an algorithm for building the sequence. In section III, the experiment results are conducted according to the proposed algorithm. Section IV is the conclusion. An abbreviation table is given after the Section IV.

II. PROPOSED METHOD

This paper tries to define the complete and balanced sequence of a two-stage pipelined MCU instruction generation for EM-dataset as a "two-stage pipelined MCU iEM-leak analysis sequence" problem (2-EMAseq) and map it to a directed graph (digraph) Eulerian trail problem. This paper also claims the proposed 2-EMAseq solution is optimal.

A. THE TWO-STAGE PIPELINED MCU IEM-LEAK ANALYSIS SEQUENCE PROBLEM (2-EMASEQ) CAN BE MAPPED TO A COMPLETE DIGRAPH

Assuming there are n instructions in a DUT, without loss of generality, these instructions can be labeled by numbers $\{0, 1, \ldots, n-1\}$. All the labeled numbers are mapped to nodes of a graph. Every arc arc_{ij} is the directed link from node i to node j. Since the 2-EMAseq problem contains all the instruction pairs (0,0), (0, 1), (0, 2), …, $(n-1, 0)$, $(n-1, 1)$, …, $(n-1, n-1)$, it is trivial to map all the instruction pair to the arc. The mapping is:

$$\text{instruction pair } (i, j) \longmapsto arc_{ij}, \quad where\ i, j \in \{0, \ldots, n-1\}$$

It is easy to find that the mapped n nodes digraph G is a complete graph with n^2 arcs. In this paper, each arc has a number label from 1 to n^2. The arc label represents its corresponding line position (instruction sequence) in the program context.

For example (FIGURE 5), if an MCU has only 3 instructions: load, store, and add, and they are labeled as {0 (load), 1 (store), 2 (add)}. Assuming the program context instruction sequence is randomly assigned as FIGURE 5(a). The mapped digraph G=(N, A) is defined as N being the node-set, and A is the arc-set. The corresponding arcs can be shown in FIGURE 5(b). It is described above that the total iEM-leak count is 9. Thus, the arc labels are defined as the program context's line numbers (labeled as {1, 2, 3, …, 9}). Here, the 3 nodes are labeled as {a, b, c} corresponding to the instruction {0, 1, 2} for clarity (not to be confused by the arc label number 1…9) and called the node labels. The mapped digraph G(N, A) is shown in FIGURE 5(c).

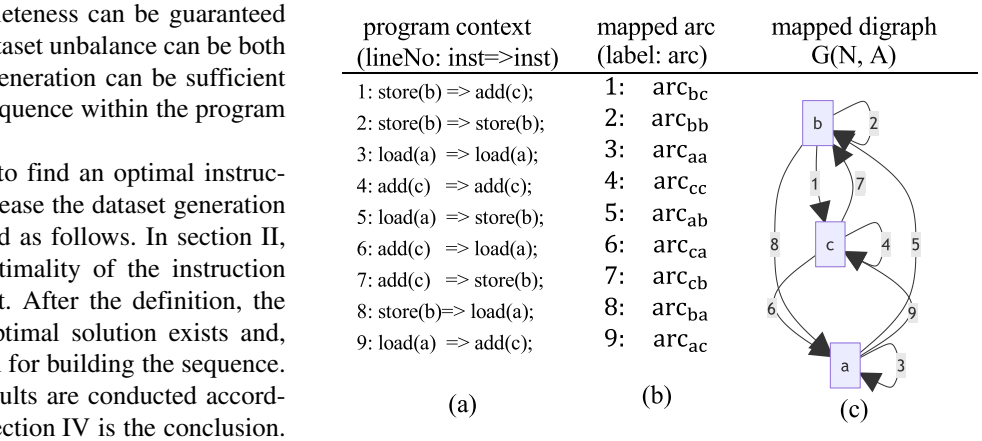


FIGURE 5. The (randomly assigned) two-stage pipelined MCU iEM-leak analysis sequence problem (2-EMAseq) case is mapped to a complete digraph: (a) the instruction sequence assigned by a program context, (b) the mapped arcs and their labels, and (c) the mapped digraph.

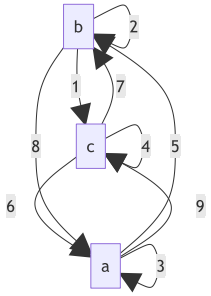
B. REPRESENTATION OF THE MAPPED GRAPH AND EXISTENCE OF THE OPTIMAL SOLUTION

The graphs in this paper follow the adjacency matrix [14] representations. For example, if a complete digraph is FIGURE 5 (c), the adjacency matrix can be represented as FIGURE 6. The integer numbers inside the matrix are the arc labels. If we order the arcs by their labels, the result is shown in FIGURE 7 which shows the node travel sequence according to the arc label. Here, we define:

$$\begin{cases} from(a_{XY}) \equiv X \equiv \text{the arc label } a\text{'s head node} \\ to(a_{XY}) \equiv Y \equiv \text{the arc label } a\text{'s tail node} \end{cases}$$

We define two adjacent labels of arcs as an "arc-pair". For an arc-pair $\{arc_i, arc_{i+1}\}$, if the node of to (arc_i) and from(arc_{i+1}) is the same, it means the adjacent arc-pair need not change node when traveling from arc_i to arc_{i+1}. This also means the instruction can be reduced by 1 without losing any completeness. In this paper, it is defined an "efficient" arc-pair as: to (arc_i) = from (arc_{i+1}). For example, in FIGURE 7,

the arc-pair {5, 6} is not efficient because the to(arc_5) = b ≠ c = from (arc_6). Arc-pair{8, 9} is efficient because the to(arc_8) = a = from(arc_9).



From/To	a	b	c
a	3	5	9
b	8	2	1
c	6	7	4

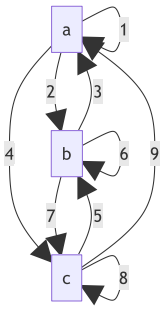
FIGURE 6. Digraph of FIGURE 5(c) and its adjacency matrix representation (matrix numbers are arc labels).

$$b \xrightarrow{1} c, b \xrightarrow{2} b, a \xrightarrow{3} a, c \xrightarrow{4} c, a \xrightarrow{5} b, c \xrightarrow{6} a, c \xrightarrow{7} b \xrightarrow{8} a \xrightarrow{9} c$$

FIGURE 7. The rearrangement of FIGURE 5(c) according to the arc-pair labels.

If an arc pair (arc_i, arc_{i+1}) is not efficient, we define there is a "redundant node" that happens in to(arc_i) and from(arc_{i+1}). The node-redundancy is defined as the ratio of the sum of redundant nodes to the number of arcs. From FIGURE 7, there are many non-efficient arc-pairs: a total of 6 redundant nodes out of the 9 arcs are counted; the node-redundancy is 6/9=0.67.

In this paper, the author tries to propose an algorithm to find the most efficient (optimized) and complete arc-pair sequence. It means the node-redundancy = 0 and all edges are traveled (see FIGURE 8 as an example compared to FIGURE 5).



(a)

$$a \xrightarrow{1} a \xrightarrow{2} b \xrightarrow{3} a \xrightarrow{4} c \xrightarrow{5} b \xrightarrow{6} b \xrightarrow{7} c \xrightarrow{8} c \xrightarrow{9} a$$

(b)

FIGURE 8. An optimal 2-EMAseq solution (node-redundancy = 0) of FIGURE 6 and FIGURE 7: (a) the mapped graph and (b) rearranged according to arc labels.

From the example above, it can be generalized into the following lemmas.

Lemma 1: If there exists an Eulerian trail [15], the node-redundancy is zero.

Proof: By the redundant node definition, if an Eulerian trail (a walk through the graph which uses every edge exactly once) exists, it is trivial that there will be no redundant node and the node-redundancy is zero.

Lemma 2: For a digraph G mapped by the 2-EMAseq problem, if the node-redundancy of G is zero, it is an optimal solution to the 2-EMAseq problem.

Proof: if the node-redundancy is zero, it implies no redundant node of a given arc sequence. If there is no redundant node, the efficiency is optimized by definition. Since there is always an edge from node i to node j in the 2-EMAseq mapped diagraph, it is clear that G is connected. If G is efficient and connected, the corresponding 2-EMAseq problem is optimized.

Lemma 3: there is at least one optimal solution to the 2-EMAseq problem.

Proof: From [17], a digraph has an Eulerian trail if and only if the 4 conditions exist:

1) At most one vertex has out-degree − in-degree = 1.
2) At most one vertex has in-degree − out-degree = 1.
3) Other vertex (in-degree) = (out-degree)
4) All of its vertices with nonzero degrees belong to a single connected component of the underlying undirected graph.

In the 2-EMAseq problem, the mapped digraph is a complete digraph. Thus, all nodes' (in-degree)=(out-degree)= n (condition 3). No nodes have unbalanced in-degree and out-degree (conditions 1 & 2). All the nodes are connected to the same component (condition 4). It is clear that the digraph mapped from 2-EMAseq problem meets all the conditions. Thus, the Eulerian trail exists in the mapped digraph.

Since the Eulerian trail exists, by Lemma1, the node-redundancy is zero. Since at least one solution that the arc sequence's redundancy is zero, by Lemma2, it is an optimal solution. This means at least one optimal solution to the 2-EMAseq problem exists.

C. PROPOSED ALGORITHM

Because the optimal solution exists, the author tries to derive an algorithm to find the solution.

The algorithm's pseudo code is shown in FIGURE 9.

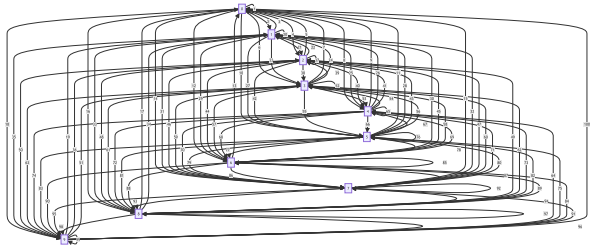
Theorem: Given a complete digraph G(N, A), where the N = $\{0, 1, \ldots, n\}$ is the node-set, $|N| = n + 1 \geq 3$, and A = $\{a_1, \cdots, a_{(n+1)^2}\}$ is the arc set. The proposed algorithm can find an optimal solution to the 2-EMAseq problem of G.

Proof: For $|N| \leq 2$, it is trivial to find the Eulerian trail in G.

Without loss of generality, the algorithm selects node n and makes G′(N′, A′) as the sub-graph of G where N′ = $\{0, 1, \ldots, n-1\}$, A′ contains only the arcs within node set N′. It is clear that G and G′ are all complete digraphs.

```
1   Input: A complete digraph G(N, A),
2        where N={0, 1, …, n}, and A={1, …, (n+1)²}
3
4   (labelSet, labelCnt) = Function label(G, labelCnt)
5   {
6       Let G' = G\{n}, A'=A\{(i,j)|i==n or j==n};
7       labelSet Labels = {};
8       for i in {0, ..., n-2}
9       {
10          Labels += (labelCnt, (n, i));
11          labelCnt ++;
12          Labels += (labelCnt, (i, n));
13          labelCnt ++;
14      }
15      Labels += (labelCnt, (n-2, n));
16      labelCnt ++;
17      Labels += (labelCnt, (n, n-1));
18      labelCnt ++;
19      (ls, lc) = label(G', labelCnt);
20      labelSet += ls;
21      labelCnt = lc ++;
22      return (Labels, labelCnt)
23  }
24
25  (labelSet, _) = label(G, 1)
```

FIGURE 9. The proposed algorithm (pseudo code) of the 2-stage pipelined iEM-leak optimal solution.



(a)

$$0 \xrightarrow{1} 0 \xrightarrow{2} 1 \xrightarrow{3} 0 \xrightarrow{4} 2 \xrightarrow{5} 0 \xrightarrow{6} 3 \xrightarrow{7} 0 \xrightarrow{8} 4 \xrightarrow{9} 0 \xrightarrow{10} 5 \xrightarrow{11} 0 \xrightarrow{12} 6 \xrightarrow{13} 0 \xrightarrow{14} 7 \xrightarrow{15}$$
$$0 \xrightarrow{16} 8 \xrightarrow{17} 0 \xrightarrow{18} 9 \xrightarrow{19} 1 \xrightarrow{20} 1 \xrightarrow{21} 2 \xrightarrow{22} 1 \xrightarrow{23} 3 \xrightarrow{24} 1 \xrightarrow{25} 4 \xrightarrow{26} 1 \xrightarrow{27} 5 \xrightarrow{28} 1 \xrightarrow{29} 6 \xrightarrow{30}$$
$$1 \xrightarrow{31} 7 \xrightarrow{32} 1 \xrightarrow{33} 8 \xrightarrow{34} 1 \xrightarrow{35} 9 \xrightarrow{36} 2 \xrightarrow{37} 2 \xrightarrow{38} 3 \xrightarrow{39} 2 \xrightarrow{40} 4 \xrightarrow{41} 2 \xrightarrow{42} 5 \xrightarrow{43} 2 \xrightarrow{44} 6 \xrightarrow{45}$$
$$2 \xrightarrow{46} 7 \xrightarrow{47} 2 \xrightarrow{48} 8 \xrightarrow{49} 2 \xrightarrow{50} 9 \xrightarrow{51} 3 \xrightarrow{52} 3 \xrightarrow{53} 4 \xrightarrow{54} 3 \xrightarrow{55} 5 \xrightarrow{56} 3 \xrightarrow{57} 6 \xrightarrow{58} 3 \xrightarrow{59} 7 \xrightarrow{60}$$
$$3 \xrightarrow{61} 8 \xrightarrow{62} 3 \xrightarrow{63} 9 \xrightarrow{64} 4 \xrightarrow{65} 4 \xrightarrow{66} 5 \xrightarrow{67} 4 \xrightarrow{68} 6 \xrightarrow{69} 4 \xrightarrow{70} 7 \xrightarrow{71} 4 \xrightarrow{72} 8 \xrightarrow{73} 4 \xrightarrow{74} 9 \xrightarrow{75}$$
$$5 \xrightarrow{76} 5 \xrightarrow{77} 6 \xrightarrow{78} 5 \xrightarrow{79} 7 \xrightarrow{80} 5 \xrightarrow{81} 8 \xrightarrow{82} 5 \xrightarrow{83} 9 \xrightarrow{84} 6 \xrightarrow{85} 6 \xrightarrow{86} 7 \xrightarrow{87} 6 \xrightarrow{88} 8 \xrightarrow{89} 6 \xrightarrow{90}$$
$$9 \xrightarrow{91} 7 \xrightarrow{92} 7 \xrightarrow{93} 8 \xrightarrow{94} 7 \xrightarrow{95} 9 \xrightarrow{96} 8 \xrightarrow{97} 8 \xrightarrow{98} 9 \xrightarrow{99} 9 \xrightarrow{100} 0$$

(b)

FIGURE 10. A 10-instruction optimal tEMseq solution by the proposed algorithm: (a) the digraph and (b) the arc sequence of (a).

According to the algorithm, the Labels of the arcs are:

$$n \xrightarrow{1} 0 \xrightarrow{2} n \xrightarrow{3} 1 \xrightarrow{4} n \xrightarrow{5} 2 \xrightarrow{6} n \xrightarrow{7} 3 \xrightarrow{8} n \xrightarrow{9} 4 \cdots \text{ (line 6)}$$
$$n \xrightarrow{2k+1} k \xrightarrow{2k+2} n \xrightarrow{2k+3} \cdots \text{ (line 8-14)}$$
$$(n-2) \xrightarrow{2n-2} n \xrightarrow{2n-1} (n-1) \to G' \text{ (line 15-19)}$$
$$\xrightarrow{2n-1+n^2+1} n \xrightarrow{n^2+2n+1} n \text{ (line 20-21)}$$

Since $|\mathrm{N}'| = n$, the algorithm recursively calls the subgraph labeling algorithm to label all the arcs with the Eulerian trail sequence of G', where $|\mathrm{A}'| = n^2$. Before the subgraph G' traveling, the arc$(n, n-1)$ is labeled as $2n-1$. When the G' is travel completely, the outgoing arc is labeled as:

$$2n - 1 + |A'| + 1 = n^2 + 2n$$

The final arc is $n \to n$ which is labeled as:

$$n^2 + 2n + 1 = (n+1)^2 .$$

Thus, the Eulerian trail can be derived from the algorithm by the mathematical induction (MI). Since the Eulerian trail can be found by the proposed algorithm, the optimal solution to the 2-EMAseq problem can be found following Lemma 3.

III. EXPERIMENTAL RESULT

For the DUT's 242 instructions (without DSP instructions), all the instructions can be inserted into the program context for acquiring the iEM-leak signals according to the proposed algorithm. Currently, only partial iEM-leak signals are measured and used to train a DNN because of the time-consuming measurements. Since the algorithm is now developed, the author can derive all the instructions' iEM-leak signals based on the 2-EMAseq solution.

The algorithm is designed by python. A 10-instruction iEM-leaks are shown here for validation. The result is shown in FIGURE 10. The algorithm can find an optimal iEM-leak program context sequence (node-redundancy=0) as expected.

About the execution time of the algorithm, for the target DUT's instruction, the algorithm takes 0.49 seconds to derive the Eulerian trail and 0.2 seconds to prepare the program context. However, the iEM-leak signal acquisition will take a longer time. And the training time would be even longer. However, the completeness of the iEM-leak signal generation is guaranteed. The detailed instructions [10] of the DUT are skipped to save space. Currently, the algorithm can only deal with the 2-stage pipelined iEM-leak program context preparation. However, modern MCUs may have 4∼7 pipeline stages. It will be the future work to derive suitable algorithms for such MCUs.

ABBREVIATIONS

2-EMAseq	two-stage pipelined MCU individual EM-leak analysis sequence problem
digraph	Directed graph
DUT	Design under test
EM-dataset	Dataset used in 2-EMAseq problem for the NN model training
EM-leak	Electromagnetic information leakage
cEM-leak	collective EM-leak
iEM-leak	individual EM-leak
FCNN	fully connected NN
IoT	Internet of Things
LSTM	Long Short-Term Memory
MCU	Microcontroller

NN	Neural network
program context	Core testing program inside the test program loop
ResNet	Residual NN
SCA	Side-channel analysis
sentential signals	Signals indicating iEM-leak's begin and end

IV. CONCLUSION

The Internet of Things (IoT) related microcontroller (MCU) is now pervasive. The neural network (NN) models for the electromagnet (EM) side-channel analysis of these MCU needs a large training/validation dataset (EM-dataset).

The EM information leakage (EM-leak) EM-dataset completeness generation for a 2-stage pipelined MCU problem is firstly observed. The EM-dataset that can solve the EM-leak analysis needs to be complete and sufficient. Due to the pipeline architecture design, dataset generation becomes a difficult issue. The sufficiency issue can be solved by increasing the EM-leak signal measurements. But the completeness of the dataset cannot be solved easily.

This paper proposes an efficient algorithm and implantation for the EM-dataset generation. The generation of EM-dataset is proved to be complete.

In this paper, the completeness of EM-dataset generation is defined as a "two-stage pipelined MCU iEM-leak analysis sequence" problem (2-EMAseq) problem. The 2-EMAseq is then proved that it can be mapped to a special directed graph (digraph) Eulerian trail problem. The 2-EMAseq problem is then proved through the special digraph that the optimal solution exists. And an algorithm is proposed and proved to solve the optimal 2-EMAseq problem which makes the EM-dataset generation complete. The experiment result shows the proposed algorithm can generate a complete EM-dataset efficiently.

The contribution of this paper includes: it successfully introduces a practical problem to a 2-EMAseq description, proves the 2-EMAseq problem has an optimal solution through a special digraph, proposes an algorithm to find the optimal solution, proves the algorithm's correctness, and uses the proposed algorithm to generate a complete EM-dataset for a targeted MCU.

Currently, the algorithm can only generate the optimal EM-dataset for two-stage pipelined MCUs. However, there are many MCUs with 4∼8 pipeline stages. Whether there is an optimal solution when the stages are more than two is still an open question. And it needs to find proofs or derive heuristics. The author will continue to study these problems.

REFERENCES

[1] L. Batina, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *Proc. 28th USENIX Secur. Symp.*, Santa Clara, CA, USA, Aug. 2019, pp. 513–532.

[2] O. Shwartz, Y. Mathov, M. Bohadana, Y. Elovici, and Y. Oren, "Reverse engineering IoT devices: Effective techniques and methods," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4965–4976, Dec. 2018.

[3] A. Burrello, A. Garofalo, N. Bruschi, G. Tagliavini, D. Rossi, and F. Conti, "DORY: Automatic end-to-end deployment of real-world DNNs on low-cost IoT MCUs," *IEEE Trans. Comput.*, vol. 70, no. 8, pp. 1253–1268, Aug. 2021, doi: 10.1109/TC.2021.3066883.

[4] M. T. Banday, "A study of current trends in the design of processors for the Internet of Things," in *Proc. ICFNDS*, Jun. 2018, pp. 1–10, doi: 10.1145/3231053.3231074.

[5] S.-Y. Yuan, "Identification of microcontroller unit instruction execution using electromagnetic leakage and neural network classification," *IEEE Trans. Electromagn. Compat.*, vol. 64, no. 4, pp. 930–940, Aug. 2022, doi: 10.1109/TEMC.2022.3159868.

[6] Y. Ji, "A novel CNN+LSTM classification model based on fashion-MNIST," *Proc. SPIE*, vol. 12258, pp. 178–184, Jul. 2022.

[7] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," 2016, *arXiv:1603.08029*.

[8] J. Deng, W. Dong, R. L. Socher, L. Li-Jia, L. Kai, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, May 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.

[9] M. Al-Marri, A. Al-Qayedi, and R. Benlamri, "WEBFACE: A web-based facial animation system," in *Proc. Int. Conf. Web Technol., Appl., Services (WTAS)*, Calgary, AB, Canada, Jul. 2005, pp. 18–22.

[10] C. Wang, M. Riviere, A. Lee, A. Wu, C. Talnikar, D. Haziza, M. Williamson, J. Pino, and E. Dupoux, "VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics, 11th Int. Joint Conf. Natural Lang. Process.* Stroudsburg, PA, USA: Association for Computational Linguistics, 2021, pp. 993–1003.

[11] T. Ouyang, V. S. Marco, Y. Isobe, H. Asoh, Y. Oiwa, and Y. Seo, "Corner case data description and detection," in *Proc. IEEE/ACM 1st Workshop AI Eng., Softw. Eng. AI (WAIN)*, May 2021, pp. 19–26, doi: 10.1109/WAIN52551.2021.00009.

[12] H. F. Li, "Pipeline architecture," *ACM Comput.*, vol. 9, no. 1, pp. 62–102, 1977.

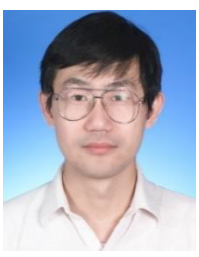
[13] P. Shukla and K. Bhowmick, "To improve classification of imbalanced datasets," in *Proc. Int. Conf. Innov. Inf., Embedded Commun. Syst. (ICIIECS)*, Mar. 2017, pp. 61–102, doi: 10.1109/ICIIECS.2017.8276044.

[14] R. B. Bapat, *Graphs and Matrices*. London, U.K.: Springer, 2011.

[15] J. Bang-Jensen and G. Z. Gutin, *Digraphs: Theory, Algorithms and Applications*, 2nd ed. Berlin, Germany: Springer, 2010.

[16] *16-Bit MCU and DSC Programmer's Reference Manual (DS70000157G)*, Microchip Technol., Chandler, AZ, USA, 2005. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/70000157g.pdf

[17] J. Jonsson, "On the number of Euler trails in directed graphs," *Mathematica Scandinavica*, vol. 90, pp. 191–214, Jan. 2002, doi: 10.7146/MATH.SCAND.A-14370.



SHIH-YI YUAN (Member, IEEE) received the M.S. and Ph.D. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1993 and 1997, respectively. He joined the Department of Communications Engineering, Feng Chia University, Taichung, Taiwan, where he is currently an Associate Professor and a member of the ICEMC Center. His research interests include compiler design, ICEMC modeling, and software solution for EMC problems.