PowerTouch: A Security Objective-Guided Automation Framework for Generating Wired Ghost Touch Attacks on Touchscreens

Huifeng Zhu, Zhiyuan Yu, Weidong Cao, Ning Zhang, and Xuan Zhang
Washington University in St.Louis, MO, USA

ABSTRACT

The wired ghost touch attacks are the emerging and severe threats against modern touchscreens. The attackers can make touchscreens falsely report nonexistent touches (i.e., ghost touches) by injecting common-mode noise (CMN) into the target devices via power cables. Existing attacks rely on reverse-engineering the touchscreens, then manually crafting the CMN waveforms to control the types and locations of ghost touches. Although successful, they are limited in practicality and attack capability due to the touchscreens' black-box nature and the immense search space of attack parameters. To overcome the above limitations, this paper presents PowerTouch, a framework that can automatically generate wired ghost touch attacks. We adopt a software-hardware co-design approach and propose a domain-specific genetic algorithm-based method that is tailored to account for the characteristics of the CMN waveform. Based on the security objectives, our framework automatically optimizes the CMN waveform towards injecting the desired type of ghost touches into regions specified by attackers. The effectiveness of PowerTouch is demonstrated by successfully launching attacks on touchscreen devices from two different brands given nine different objectives. Compared with the state-of-the-art attack, we seminally achieve controlling taps on an extra dimension and injecting swipes on both dimensions. We can place an average of 84.2% taps on the targeted side of the screen, with the location error in the other dimension no more than 1.53mm. An average of 94.5% of injected swipes with correct directions is also achieved. The quantitative comparison with the state-of-the-art method shows that a better attack performance can be achieved by PowerTouch.

1 INTRODUCTION

The capacitive touchscreen technology has been widely used in every sphere of our daily life, ranging from smart devices and automobiles to medical equipment and industrial control centers [8, 24]. Its popularity stems from the inherent advantage in providing a more convenient human-computer interaction interface by directly using fingers than conventional methods through auxiliary mediums, such as keyboards and mouses. Given its increasingly prominent role in modern technologies, the essential capability of touchscreens to reliably and correctly recognize touch events (e.g., taps and swipes) is vital. Otherwise, once compromised, attacks on touchscreens could not only significantly deteriorate the usability of the device but also severely threaten user privacy.

Unfortunately, capacitive touchscreens have been shown to be susceptible to electromagnetic interference (EMI) due to their intrinsic electromagnetic characteristics. The capacitive property of the touchscreens can be easily manipulated with the presence of electromagnetic fields. Attackers are thus able to take advantage of this vulnerability to conduct **Ghost Touch Attack** by injecting EMI to the capacitive touchscreen such that it falsely recognizes nonexistent touch events (i.e., ghost touches). This ghost touch is then identified as the authenticated behaviors of users who in fact do not physically touch the screen using fingers, thereby leading to unintended and even malicious operations. One tangible ghost touch attack is injecting fake touches to smartphones to click the answering buttons without the users' attention when an eavesdropping phone call dials in [25]. Further, attackers can alter the actual touch of a user to authenticate a malicious operation even though the user intends to click "CANCEL" button [18].

Despite its great threat potential, ghost touch attack is an emerging topic in security and remains largely unexplored. So far, only few works have demonstrated the attacks [11, 18, 22, 25]. Initial attempts , including prior works such as Tap'n Ghost [18], GhostTouch [25], and Invisible Finger [22], use radiated EMI for attacks. These early explorations assume that attackers can stealthily retrofit the table under the phone to install equipment (e.g., antenna array or device locator circuits) for EMI injection. However, such sophisticated setups are nowhere practical in real-world scenarios. A parallel work (i.e., WIGHT [11]) has proposed a wired ghost touch attack, which is an improved method using conducted EMI. Specifically, the authors inject *common-mode noise (CMN)* to target devices via the power cable (e.g., USB charging cable) and control the locations of ghost touches by adjusting the CMN waveform (see Figure 3 for an example of CMN waveform).

Although the wired ghost touch attack significantly lowers the barrier of entry for attackers, it still suffers from several challenges. First, *the method exhibits low controllability.* Based on injecting CMN with simple patterns, it can only randomly place taps on one horizontal or vertical line. To achieve higher attack capabilities (e.g., swipe) and finer-granular controllability, CMN with more complex patterns is required. Second, *manually crafting effective CMN waveform is a cumbersome procedure and time-consuming.* Previous efforts are largely based on manually sweeping the search space in a trial–and-error manner, while automatically designing such CMN given arbitrary security objective remains largely unexplored. Besides, experiments in previous work typically involve multiple instruments. Even a slight change in the CMN waveform requires adjusting excessive configuration parameters of instruments. Third, *the black-box nature of touchscreens results in an immense search space.* For touchscreens, the design details of both their internal circuits and noise filtering algorithms are not publicly available and vary significantly among different vendors. Attackers can rarely attain practical constraints to reduce the search space unless they entirely reverse engineering against touchscreens.

In this paper, to overcome the limitations of existing wired ghost touch attacks, we present *PowerTouch*–a genetic algorithm (GA) based framework that can automatically generate wired ghost touch

ICCAD '22, October 30-November 3, 2022, San Diego, CA, USA
© 2022 Copyright is held by the owner/author(s).
ACM ISBN 978-1-4503-9217-4/22/10.
https://doi.org/10.1145/3508352.3549395

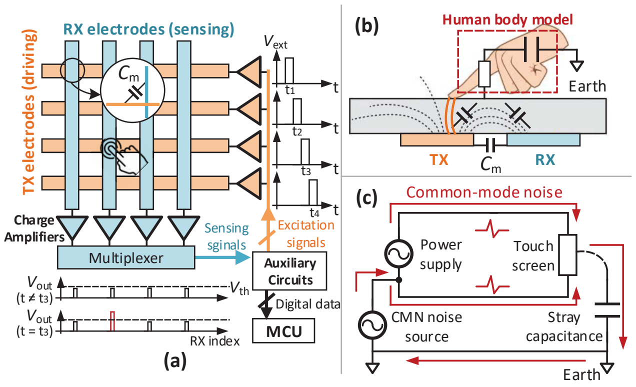

Figure 1: (a) A typical system architecture of capacitive touchscreens with the scan driving method. (b) Mutual capacitive sensing. (c) Schematic diagram of common-mode noise.

attacks and find optimal CMN waveform to maximize attack performance given a security objective[1]. To the best of our knowledge, this is the first framework to automate ghost touch attacks. PowerTouch is developed by using a software-hardware co-design approach. A domain-specific GA is adopted to be a black-box optimizer, which treats the target touchscreen system as a black box. Unlike standard GA, we propose a tailored genetic encoding scheme and evolution operators. The complexity of the CMN waveform is thus also optimized to attain a complete search. To efficiently evaluate CMN waveform, we develop a software infrastructure to bridge the gap between the software and hardware. It takes high-level CMN parameters as inputs and controls the custom-designed arbitrary CMN injector at the hardware level to generate corresponding CMN. The software infrastructure coordinates instruments and configures amplitude and frequency modulation to enable generating complex CMN waveforms. This evaluation procedure is thus automated. The generality and effectiveness of PowerTouch are demonstrated through optimizing CMN waveform towards different objectives, such as taps on different regions, and swipe up/down, swipe left/right. Aided by PowerTouch, the advanced capability, such as injecting swipes using CMN, is presented for the first time. The contributions of this work are summarized as below:

- **Domain-specific Genetic Algorithm**. We adopt a GA specifically to generate CMN waveform for launching wired ghost touch attack given a security objective. We design a specialized genetic encoding scheme, a specialized crossover operator, and a new genetic operator. The complexity of CMN waveform at different levels can thus be optimized efficiently.
- **Automated Workflow.** Our framework is devised using an software-hardware co-design approach to automate the search space exploration. Specifically, all procedures of evaluating CMN waveform including sensing the touchscreen, generating complex signals, injecting CMN, and analyzing ghost touch, are automated. Users can focus on designing fitness functions without knowing the details at the lower level.
- **Improved Ghost Touch Attack Capability.** We conducted extensive experiments to demonstrate the effectiveness of PowerTouch given nine different objectives on two different smartphones. Compared with previous attacks based on CMN, we achieve a more powerful attack by showing additional capability of controlling taps on the other dimension and injecting swipes. For example, an average of 84.2% taps are placed on the targeted side of the screen, with the location error in the other dimension no more than 1.53mm. An average of 94.5% of injected swipes with correct directions is also achieved.

[1]We have open sourced the tool with the source code available at: https://github.com/xz-group/PowerTouch

2 BACKGROUND

This section introduces the background of wired ghost touch attacks by discussing the mechanism of capacitive touchscreens and the common-mode noise. The basic concept of the genetic algorithm is also presented.

2.1 Capacitive Touchscreen

Figure 1(a) shows a typical system architecture of capacitive touchscreens, which consists of touch sensors, analog auxiliary circuits, and micro-controller unit (MCU) [14]. The touch sensor is a two-dimensional crossbar array of two-layer transparent conductive electrodes: the transmitting (TX) electrodes (the orange lines) and the receiving (RX) electrodes (the blue lines). There is an insulator layer between the two layers of electrodes (e.g., indium-tin-oxide (ITO) [10]). Thus, parasitic mutual capacitors are formed at each cross point of TX and RX electrodes. When a finger touches the screen, it absorbs a partial electric field at this point due to the additional capacitive coupling (see Figure 1(b)), affecting the mutual capacitance C_m. The auxiliary circuits and the MCU monitor such changes to recognize the touch events. Note that in this paper, we specify the two dimensions of the touchscreens as TX and RX dimensions instead of vertical and horizontal dimensions. The orientation of TX electrodes varies among different devices.

To measure the mutual capacitance of the whole touchscreen, several scanning method have been proposed [1, 15, 19, 23]. In this paper, we consider the most classical one, i.e., the scan driving method (SDM). As shown in Figure 1(a), the auxiliary circuits sequentially send excitation signals V_{ext} (e.g., square wave signals) to every TX electrode in each touchscreen refresh. When driving one TX electrode, the charges can be sensed on all RX electrodes, which is determined by $Q_s = C_m \times V_{ext}$. Then the charge amplifiers simultaneously converts Q_s to voltages V_{out}. These voltages are processed one-by-one after passing the multiplexer, and finally are converted to digital readouts. Theoretically, a touch event will be reported if V_{out} is higher than the pre-defined threshold V_{th}. In practice, customized noise mitigating circuits and anti-mistouch algorithms are implemented to improve the reliability of touchscreens [16, 29]. Since such techniques are the critical intellectual property of the vendors, the design details are not publicly available, making a touchscreen a black-box system.

2.2 Common-mode Noise on Power Cables

The ghost touch attacks are fundamentally resulted from electromagnetic interference (EMI). According to the different paths that EMI is propagated to the target devices, EMI can be further categorized into two major types: radiated EMI and conducted EMI [31]. This work focuses on one notable source of conducted EMI: the common-mode

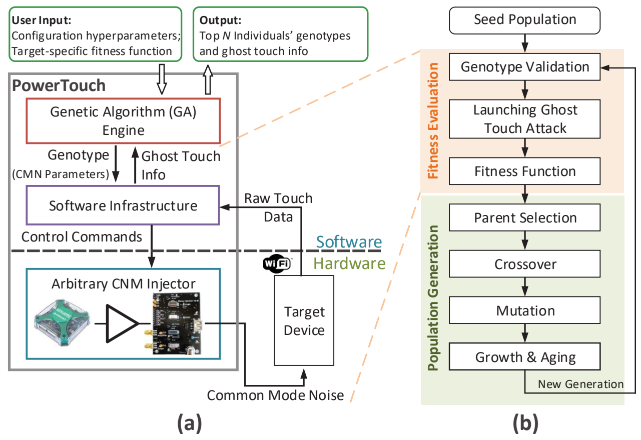

Figure 2: (a) The overview of PowerTouch framework. It consists of three main parts: genetic algorithm (GA) engine, software infrastructure, and arbitrary CMN injector. (b) The workflow of GA-based wire ghost touch attack generation.

noise (CMN) on power cables. CMN could severely compromise the normal functionalities of various devices including capacitive touchscreens [7, 13]. As illustrated in Figure 1(c), CMN is a type of noise due to the ubiquitous stray capacitance between the device and the earth. It is caused by current leakage through the stray capacitance, which then returns to the noise source via the earth [30]. Since the currents in both VDD and GND are in the same direction, the voltage fluctuations equally appear on the two lines, making CMN sneaky yet harmful. For instance, a large CMN can be injected into a USB charger while keeping its differential output voltage at a clean 5V power supply. However, once connecting the charger to a device, the large current induced by CMN may damage the system.

The capacitive touchscreens are intrinsically vulnerable to the CMN. The intuition is that the touchscreens fundamentally leverage the current in the device-human-earth path to detect touch events, which is a common-mode path. Based on this vulnerability, in a recent work (i.e., WIGHT [11]), the authors proposed a wired ghost touch attack. The adversaries can implement malicious charging stations in public places and inject CMN into the target device via power cables (e.g., USB charging cable). The CMN can be mistakenly recognized as ghost touches. To control the locations of ghost touches, attackers need to identify the timing when a specific TX electrode is being scanned, then inject a carefully-crafted CMN. However, although successful, the black-box nature of touchscreens and the immense parameter search space limit the attack capability based on manual heuristic approaches. The crafted waveform also does not readily port over to a different new device.

2.3 Genetic Algorithm

In PowerTouch, we develop a genetic algorithm (GA) based black-box optimization technique to tackle the above limitation and automatically generate CMN injecting ghost touches. GA is a popular artificial intelligence (AI) algorithms due to its simplicity and generality [3, 12, 17]. It optimizes a target function $F(\mathcal{G})$ by applying genetic operators (e.g., crossover and mutation) to input sequence $\mathcal{G}$, mimicking the natural evolution. $\mathcal{G}$ here is called the genotype and every parameter in $\mathcal{G}$ is called the gene. Standard GA starts with a randomly sampled

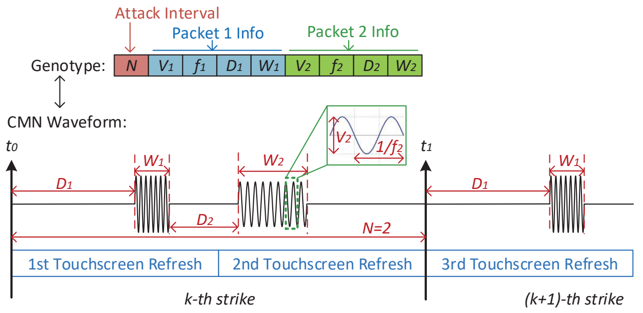

Figure 3: One example of CMN waveform and its genotype representation.

seed population (i.e., the initial generation). Every generation consists of individuals, and each individual is represented by a genotype $\mathcal{G}$. For each generation, individuals' fitness scores will be evaluated first. Then the best-fitted individuals will be selected as parents which generate a new generation after crossover and mutation. The process of evaluating fitness and creating the next generation will be repeated. As the population evolves, a particular bias is introduced in terms of that new points in the search space will be sampled [9, 26].

In standard GA, the search space is rigid, meaning the number of parameters in $\mathcal{G}$ is fixed to fit in genetic operators. However, for generating ghost touch attacks, a complete search is desired, meaning the complexity of CMN waveform is an essential dimension worth exploring and optimizing. This requirement makes the lengths of $\mathcal{G}$ variable as the CMN of different levels of complexity require a various number of parameters to describe. One innovation in our work is to customize the evolution mechanism to make it compatible with flexible-length $\mathcal{G}$.

3 POWERTOUCH FRAMEWORK

A high-level overview of PowerTouch framework is shown in Figure 2(a). PowerTouch is based on a software-hardware co-design approach to automatically find optimal CMN waveform for ghost touch attacks. PowerTouch is written in Python 3 and takes inputs via a Python script which defines configuration parameters and objective-specific fitness function. There are three major components in PowerTouch: genetic algorithm engine (Section 3.1), software infrastructure (Section 3.2), and arbitrary common-mode noise (CMN) injector (Section 3.3). The GA engine is the core of PowerTouch and coordinates its execution. The software infrastructure and arbitrary CMN injector work together as a ghost touch attack analyzer, which takes as input genotype $\mathcal{G}$ (i.e., high-level parameters of the CMN waveform) and returns the information of injected ghost touches. We describe each component in detail as below.

3.1 Genetic Algorithm Engine

In our work, we formulate the ghost touch attack (GTA) as a black-box optimization problem:

> ***GTA problem:*** *find the optimal common-mode noise waveform that efficiently, precisely, and exclusively injects a specific type of ghost touch to the capacitive touchscreen without prior knowledge of the circuits of touchscreens or the design of touch recognition algorithms.*

Figure 3 illustrates one example of the CMN waveform, which is an analog signal consisting of several packets of sinusoidal wave pulses. The implementation of GA typically includes three parts: the problem encoding, which defines the input variables to be optimized and the method for representing an individual; the fitness evaluation, which establishes the method for measuring the quality of an individual; and the procedure of population generation.

Problem Encoding. In our initial exploration, we broadly investigate which properties of the CMN waveform can impact the ghost touch attack performance. Based on the results, we define the parameters of CMN waveform (i.e., genotype $\mathcal{G}$) to be optimized as:

$$\mathcal{G} := \{N, V_i, f_i, D_i, W_i, ..., V_n, f_n, D_n, W_n\}. \tag{1}$$

Here N is the attack interval, defined as injecting one CMN strike per N frames of touchscreen refreshes. We find N can influence the type of injected ghost touches. For example, for a smaller N, the touchscreen tends to identify the noises as the swipes rather than taps. One CMN strike consists of n packets of sine wave pulses, each of which is determined by four parameters: amplitude (V_i), frequency (f_i), delay (D_i) and width (W_i). Among these parameters, V_i and f_i are related to the attack efficiency. For instance, one can inject a large number of ghost touches by properly setting V_i and f_i. Meanwhile, since the touchscreens are based on sequentially scanning the TX electrodes, D_i and W_i are vital for controlling the location of ghost touches. At last, the number of packets n can be viewed as the measure of the complexity of CMN waveform. And the number of parameters to describe a CMN waveform varies with different n. We find there typically is an optimal n for different levels of ghost touch controllability. In the GA engine, all the above properties will be simultaneously explored and optimized.

Figure 3 illustrates an exemplary CMN waveform with $N = 2$ and $n = 2$. At t_0, the touch screen starts to scan a new frame and the oscilloscope is triggered. After a delay D_1, the first packet with a width of W_1 is injected. After another delay D_2, the second packet with a width of W_2 is injected. Since the attack interval is $N = 2$, the oscilloscope is not triggered until the start of the third frame scan at t_1. In this work, we focus on injecting Sine wave signal into the devices as its spectrum is well controlled.

Fitness Evaluation. The overall workflow of GA-based optimization is illustrated in Figure 2(b). In fitness evaluation, the GA engine first validates the genotype $\mathcal{G}$ described in Equation (1), where parameters are verified by satisfying user- or hardware constraints (e.g., maximum available noise frequency). Then the genotype is given to the software infrastructure. The software infrastructure and arbitrary CMN injector corporately launch the ghost touch attack. For each individual, we inject CMN strikes multiple times at the attack interval N to collect the statistics of the individual's performance. After injecting the desired number of strikes, the GA engine receives the reported ghost touch information (i.e., three `Pandas` dataframes containing data of taps, swipes, and comprehensive statistics, respectively) from the software infrastructure. Based on this information, the fitness function scores each individual.

Note that fitness function is objective-specific and different objectives typically require unique fitness functions. We discuss fitness function design principles here and introduce fitness functions for each objective in Section 4. For the GTA problem, we evaluate CMN waveform based on three metrics.

- **Efficiency.** The number of ghost touches of the desired type (e.g., swipe upwards) within a certain number of strikes. With higher efficiency, the attack performance of the CMN waveform is more reliable, and the average time between the start of an attack to the moment it succeeds is shorter.
- **Precision.** The distribution of ghost touch locations, such as the mean absolute error (MAE) to the target location and the standard deviation (SD). This metric affects the success rate when attackers intend to tap on a button.
- **Exclusiveness.** The proportion of the desired type of ghost touches among all ghost touches. Since other types of ghost touches may invalidate the attack, exclusiveness also impacts the success rate.

Note that all above metrics should be included in fitness functions with equal importance. Otherwise, we may succumb to pitfalls. For example, by considering only precision, the GA engine may find CMN waveform that can precisely but barely inject successful ghost touches, resulting in unreliable CMN waveform. Another challenge here is to balance these three metrics to avoid the overall fitness being solely dominated by one metric with an extremely high score. To resolve this issue, we propose to reshape the metrics using a logistic function [27] defined below:

$$S(x \mid x_0, k) = [1 + e^{-k(x - x_0)}]^{-1}, \tag{2}$$

where sufficiently good metrics can be soft-capped. Here, x is the input metric value. x_0 is the x value of the curve's midpoint and k is the logistic growth rate. They determine the shape of the curve.

Population Generation. As shown in Figure 2(b), once the fitness evaluation is completed, genetic operators are applied to create a new generation (i.e., offspring). To make our method take any-length $\mathcal{G}$ as inputs, we modify the Crossover operator. One new operator, Growth & Aging, is also introduced to mutate n (i.e., the complexity of CMN). Each operator are discussed as follows:

- **Parent Selection.** We empirically use the Tournament selection method. In a K-way tournament selection, we select k-individuals and run a tournament among them. Only the fittest candidate among those selected candidates is chosen and passed to the next generation. This process will repeat until the desired number of parents is selected.
- **Crossover.** The parents are first randomly paired. Then the parents' genotypes $\mathcal{G}$ are re-combined to produce the offspring. We adopt uniform crossover, where for each gene, the offspring has an equal chance of inheriting it from either of the parents. Since the genotypes of two parents may have different lengths, the shorter genotype will be padded with placeholders to the end to align with the longer one. Note that every four parameters of one packet (V, f, D, and W) are viewed as one gene to be inherited. After crossover, we remove the placeholders in the offspring's genotype, if any.
- **Mutation.** We adopt random mutation here. Each parameter defined in Equation (1), with a certain probability, will be mutated by adding a random value (determined by the level of mutation) to the original parameter.
- **Growth & Aging.** With a certain probability, the gene of one packet (including V, f, D, and W) will be added to the end of the original genotype (i.e., Growth) or removed from the end of the original genotype (i.e., Aging) if the afterward number of packets does not exceed the user-defined range.

3.2 Software Infrastructure

The software infrastructure works as an intermediate layer for interacting with the hardware part of PowerTouch. It consists of three parts: instruments driver, attack template, and touchscreen monitor. Using the former two, the software infrastructure automatically sets up the hardware based on the input CMN waveform parameters (i.e., genotype $\mathcal{G}$). Meanwhile, the touchscreen monitor collects and analyzes the information of injected ghost touches.

Instrument Driver. The experimental setup in previous work [11, 18, 22, 25] includes diverse instruments and is highly intricate. To address this limitation, we render a much more concise and low-cost setup based on Analog Discovery 2 (AD2), which is a USB multi-function instrument [6]. AD2 provides application programming interfaces (APIs) for users to program AD2 by interacting with AD2’s registers. We combine and wrap the APIs to automate configuring instruments based on our application. Based on our application, four instrument drivers are implemented: Oscilloscope, Waveform Generator (Analog), Logic Analyzer, and Pattern Generator (Digital), and DC Power Supply.

Attack Templates. Even with instrument drivers, launching an attack can still be cumbersome. A slight change in the CMN waveform could require adjustment of more than 20 instrument setup parameters as all the instruments must be tightly synchronized. To this end, we further abstract the interfaces as attack templates to automate instrument cooperation. Therefore, users can conveniently input the CMN waveform parameters without knowing the details of AD2.

Touchscreen Monitor. The touchscreen monitor is used to collect raw touch information from the target device, such as the location, duration, and type of the touch. The raw touch information refers to the `ABS_MT` events, which is the information that the touchscreen controller reports to the kernel by following the multi-touch (MT) protocol [21]. Nowadays, most touchscreen controllers use the type-B MT protocol to simultaneously track different contacts (i.e., different fingers). Each contact will be assigned with a unique tracking ID. We capture `ABS_MT` events through Android Debug Bridge (ADB) and implement a MT protocol parser to convert raw `ABS_MT` events into touch events.

3.3 Arbitrary Common-mode Noise Injector

Figure 4 shows the schematic of the arbitrary CMN injector. It consists of an AD2 instance, a high-voltage amplifier (PiezoDrive MX200 [20]), and our customized noise injection PCB. The MX200 can generate high-voltage signals with a maximum amplitude of 100V and bandwidth of 500KHz[2]. Two relays (CH1 and CH2) in noise injection PCB are used to isolate the low-voltage (blue) and high-voltage (red) paths. Thus AD2 can be protected from being damaged by high-voltage common-mode noise. Under the control of the software infrastructure, AD2 continuously senses the touchscreen TX excitation signals emitted from the target device. Once triggered, AD2 generates a waveform at the desired timing. The waveform is further amplified as a high-voltage signal, then converted to CMN and injected into the target device aided by the noise injection PCB. We will discuss the key techniques involved in this process next.

[2]The original bandwidth is 200KHz, but a 500KHz equivalent bandwidth can be achieved after compensating the decayed gains.

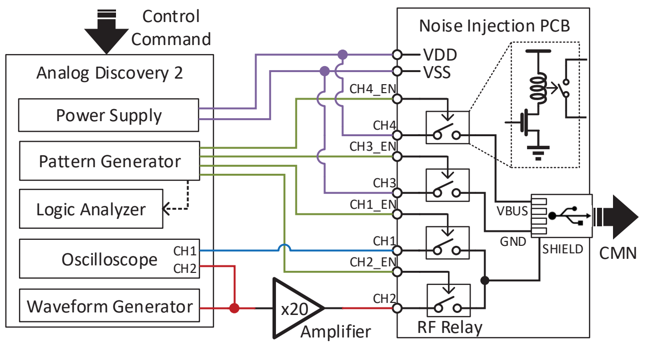

Figure 4: The schematic of arbitrary CMN injector.

Extended Noise Waveform Generation. We make use of the Waveform Generator and Oscilloscope instruments in AD2 to generate and examine the CMN waveform respectively. The key technical challenge in generating CMN specified in Equation 3 is its high complexity. The requirements for separately adjusting each packet (including V, f, D, and W) force one to adopt expensive commercial Arbitrary Waveform Generators. Otherwise, CMN waveform with more than one packet can not be generated. To address this challenge, we adopt both amplitude modulation (AM) and frequency modulation (FM) techniques. And the modulation signals are generated using direct digital synthesis (DDS) technique [5]. The AM is responsible for setting amplitudes of the packets and converting continuous signals into multiple separated packets. Then FM adjusts the frequencies of Sine wave signal in each packet. This process, including generating corresponding modulation signals, is automated in the software infrastructure.

Common-mode Noise Injection. To inject the noise, we choose to attach the noise source (i.e., amplifier) to the `SHIELD` of the USB power cable, which is a common approach to testing EMI tolerance [4]. According to the USB standard, a USB cable (including all 14 different connector types [28]) must have `SHIELD`, which is made of a stranded copper bread. It surrounds all other wires on the whole length of the cable and is connected to the plug shells at both ends. By connecting the positive and negative ends of the noise source to `SHIELD` and the earth, respectively, the noise is converted to the common-mode noise from the target perspective.

Synchronization with Touchscreen Refresh. PowerTouch fundamentally synchronizes with touchscreen refresh to determine the timing to launch one new round of strike. The Oscilloscope in AD2 monitors the TX excitation signals unintentionally transmitted from the touchscreen through the power cable and is triggered when the touchscreen starts scanning a new frame. Once triggered, the Waveform Generator outputs noise signals. Meanwhile, the Pattern Generator outputs control signals that turn on/off CH1 and CH2 relays according to defined attack timings. It is worth mentioning that the synchronization techniques in previous work [11, 25] are based on first reverse-engineering the devices to interpret TX excitation signals, and then determining when to inject noises. This process is expensive and time-consuming. Besides, because different devices typically use unique touchscreen designs, their TX excitation signals can be completely different (see Figure 5). Such facts make techniques proposed in previous work [11] impractical for new devices. However, in PowerTouch, aided by GA-based automated flow, users only need to identify the periodical touchscreen refresh signal.

Table 1: Summary of Evaluated Objectives.

	Nexus 5 (Horizontal TX electrodes)	Redmi 5A (Vertical TX electrodes)
T1	Taps on 10th TX electrode	Taps on 2nd TX electrode
T2	Taps on 15th TX electrode	Taps on 5th TX electrode
T3	Taps on 20th TX electrode	Taps on 8th TX electrode
T4	Taps on 15th TX electrode (left side)	Taps on 5th TX electrode (upper side)
T5	Taps on 15th TX electrode (right side)	Taps on 5th TX electrode (lower side)
S1	Swipe Up	Swipe Left
S2	Swipe Down	Swipe Right
S3	Swipe Left	Swipe Up
S4	Swipe Right	Swipe Down

Nexus 5 has 27 TX electrodes [25]. For Redmi 5A, we assume the number to be 10.

4 EXPERIMENTAL RESULTS

We evaluate the performance of PowerTouch using nine different objectives. For each objective, PowerTouch is evaluated on two smartphones from different brands: LG Nexus 5 and Xiaomi Redmi 5A. The effectiveness of PowerTouch is demonstrated by automatically generating CMN waveform to inject desired ghost touches.

4.1 Experimental Methodology

Objectives and Fitness Functions. The summary of the nine evaluated objectives is listed in Table 1, with five for injecting taps (T1~T5) and the other four for injecting swipes (S1~S4). For both types of touches, the controllability of two dimensions (i.e., TX and RX) is included. Based on the principles introduced in Section 3.1, for injecting taps (T1~T5), we devise the fitness function as:

$$Fitness = \log_{10}(N_{tap}) \times S(P_{tap}|0.7,20) \times S(P_{hit}|0.5,10) \times [S(SD_{TX}|100,-0.05) + S(MAE_{TX}|250,-0.02)],$$

where N_{tap} is the number of injected taps; P_{tap} is the proportion of taps among all injected touches; and SD_{TX} and MAE_{TX} are the standard deviation and the mean absolute error of taps to the location of the target TX electrode; P_{hit} is the proportion of taps hitting the targeted half side (e.g., left) of the screen (for T4 and T5). We make $P_{hit} = 1$ when targeting T1~T3. Similarly, we design the fitness function for injecting swipes (S1~S5) as:

$$Fitness = \log_{10}(N_{swipe}) \times [S(P_{swipe}|0.55,10) + 10S(P^{c}_{swipe}|0.7,20)],$$

where N_{swipe} is the number of injected swipes; P_{swipe} is the proportion of swipes among all injected touches; P^{c}_{swipe} is the proportion of swipes with the correct direction among all injected swipes.

PowerTouch Configuration. In this work, we follow existing work on ghost touch attacks [22, 25] and assume the device is placed face-down on the table. To achieve a better signal quality, we further assume the main part of the table is conductive. For the experiments in this paper, we place a metal sheet on a wood-made table, and the metal sheet is connected to EARTH of a power socket on the wall. A glass plate is placed above the metal sheet. Figure 5 shows the TX excitation signals of two devices collected by PowerTouch, where the touchscreen refresh can be easily recognized (see the red labels). We found the refresh rates of Nexus 5 and Redmi 5A are around 120Hz and 115Hz, respectively. To synchronize with Nexus 5, we set the oscilloscope to trigger on rising edges with the trigger level at 25mV. For Redmi 5A, the oscilloscope is triggered at falling edges with the trigger level at −540mV. The sampling rate of the oscilloscope is configured at 400KHz. For GA related configurations, we set

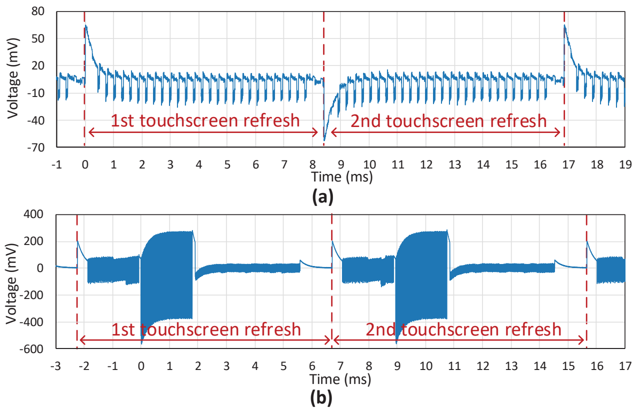

Figure 5: The waveform of TX excitation signals on (a) Nexus 5 and (b) Redmi 5A.

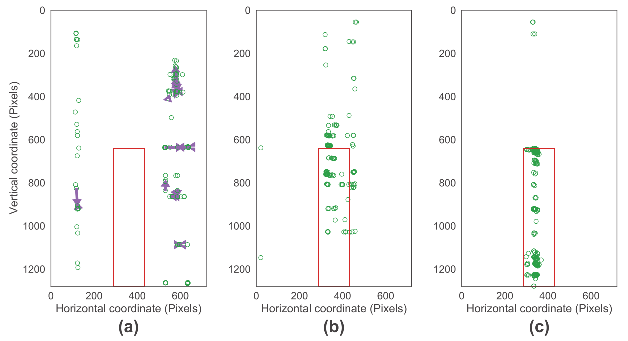

Figure 6: An illustration of changes in distributions of injected ghost touches as evolution continues: the best individual after (a) 5, (b) 10, and (c) 25 generations. The green circles are the taps, and the purple arrows are the swipes. The red box is the target area. The objective is T5, and the experiment is conducted on Redmi 5A.

population_size = 50, and number_of_generations = 25. In parent selection, 20 individuals will be selected as parents (tournament_size = 3). The mutation probability and growth&aging probability are both 0.2. We set the maximum number of packets as 4.

4.2 Effectiveness

In this subsection, we demonstrate the effectiveness of PowerTouch by showing the performance of CMN waveform generated by our framework for each objective listed in Table 1. Figure 6 illustrates the changes in the distribution of injected ghost touches as evolution continues. The green circles are the taps and the purple arrows are the swipes. The red box is the target area. In this example, we aim at injecting pure taps to Redmi 5A, where the taps should be placed on the lower side of the 5th TX electrode (i.e., the objective T5). Figure 6(a), (b), and (c) present the best individual after 5, 15, 25 generations. For the ultimate CMN waveform generated by PowerTouch, the majority of injected ghost touches are located in the target area. Similar results are achieved when targeting other objectives on both devices. Next, we analyze the detailed metrics of generated CMN waveform for each objective. Specifically, the performance of CMN waveform is evaluated from efficiency, exclusiveness, and precision perspectives.

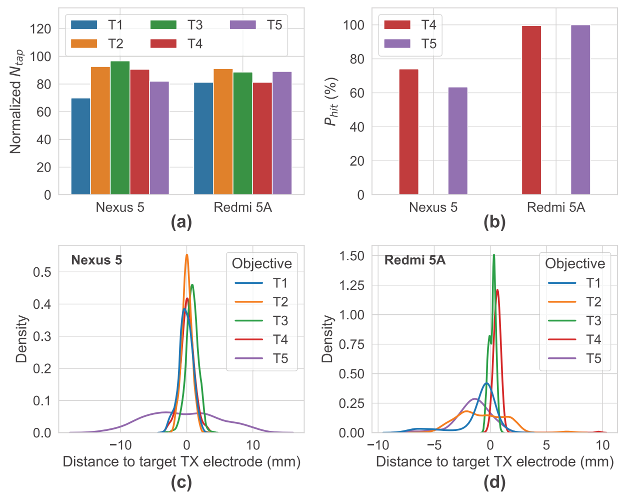

Figure 7: The results of objectives T1∼T5: (a) Normalized number of injected taps per 100 strikes; (b) the proportion of taps that hit the target region; (c)(d) The distribution of distance from injected taps to the target TX electrode.

Injecting Taps. Here the objectives T1∼T3 require placing taps on a specific TX electrode. For T4 and T5, besides controlling touches on desired TX electrode, they further require controlling the locations on the RX dimension. We found PowerTouch can successfully achieve all objectives. Figure 7(a) illustrates the efficiency, where we plot the normalized N_{tap}. It is the number of injected taps per 100 strikes. The overall averaged N_{tap} over two devices is 86.3, and the N_{tap} for individual objectives reaches up to 96.7 on Nexus 5 for the objective T3. For exclusiveness, we focus on two metrics: the proportion of taps among all injected touches P_{tap}, and the proportion of taps that hit the desired region on the screen P_{hit}. For all objectives, 100% of P_{tap} is achieved, meaning no swipes are injected. Figure 7(b) shows the comparison of P_{hit} for objective T4 and T5. On Nexus 5, P_{hit} are 74% and 63% for the two objectives. While on Redmi 5A, P_{hit} are 99% and 100%.

Figure 7(c) and (d) show the precision of the injected taps. The distance distribution from injected taps to the target TX electrodes is plotted. Due to the different TX orientations, we calculate the distance (number of pixels) in the vertical dimension for Nexus 5 and the horizontal dimension for Redmi 5A. The number of pixels is further converted into the absolute distance (i.e., millimeter) by considering the physical dimensions and resolution of the screen. For Nexus 5, the maximum mean error is no more than 0.8mm (11.5 pixels). The maximum standard deviation is 5.27mm (73.4 pixels) for objective T5, with the average standard deviation of other objectives being 0.9mm (13.1 pixels). For Redmi 5A, the mean error is no more than 1.5mm (16.3 pixels), with the standard deviation no more than 2.1mm (22.3 pixels). Compared with the size of an average fingerprint (17.8×12.7mm) [2], we can achieve reasonable precision.

Injecting Swipes. Similarly, objectives S1 and S2 are based on the controllability of placing swipes on TX electrodes, while S3 and S4 are based on the controllability of RX electrodes. PowerTouch is shown to successfully achieve the four objectives, and the performance of optimal CMN waveform for each objective is illustrated in Figure 8.

Figure 8(a) presents the number of injected swipes per 100 strikes N_{swipe}, where the N_{swipe} of two devices can be up to 94.9 and 92.5 respectively. Regarding the exclusiveness, the proportion of swipes among all injected swipes P_{swipe} is plotted in Figure 8(b). The P_{swipe} up to 99% can be achieved, meaning the swipes can be almost exclusively injected. On both devices, we find injecting swipes along the TX direction (i.e., S3 and S4) is more difficult than swipes perpendicular to the TX direction (i.e., S1 and S2). For example, on Nexus 5, the average P_{swipe} for S1 and S2 is 79%, whereas the average P_{swipe} for S3 and S4 being 37%. Correspondingly, the average N_{swipe} for S3 and S4 is 67.9 lower than the one for S1 and S2. It is relatively more efficient to inject swipes on Redmi 5A compared with Nexus 5, with the average N_{swipe} on the former is 85.6. Note that for all objectives on both devices, PowerTouch ensures the majority of the injected swipes is in the correct direction. Figure 8(c) shows the proportion of swipes with the correct direction among all injected swipes P^{c}_{swipe}. The average P^{c}_{swipe} for Nexus 5 and Redmi 5A are 93% and 96%, respectively.

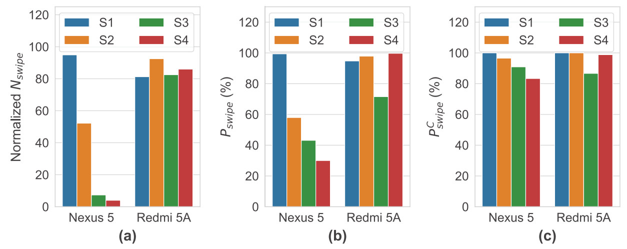

Figure 8: The results of objectives S1∼S4: (a) Normalized number of injected swipes per 100 strikes; (b) The proportion swipes among all injected ghost touches; (c) Among injected touches, the proportion of swipes with correct direction.

4.3 Security Scenario Evaluation

In this subsection, we focus on evaluating the attack performance achieved by PowerTouch by considering the real security scenarios. Four example attacks are listed below, each requiring a different level of ghost touch controllability. Note that since the required controllability is highly affected by the user interface (UI), we show the UIs of the targets in Figure 9 for clarification.

- **Establish malicious connection.** Attacker can initialize a connection request (e.g., Bluetooth) to victim's device and tap the "ALLOW" button (see Figure 9(a)). This connection can be further used to fully control the victim's phone. The attacker can inject swipes when injecting taps since the current UI is not sensitive to swipes.
- **Answer an eavesdropping call.** Suppose the attacker priorly attains the victim's phone number. The attacker can call the victim and answer it without noticing. This call can thus be used for eavesdropping. Under the UI shown in Figure 9(b), the attacker needs to inject swipes to answer the call. Taps can also be injected since the current UI is not sensitive to taps.
- **Click malicious link.** The attacker sends the victim a message containing a malicious link. By clicking the link, malware can be installed on the victim's phone. As shown in Figure 9(c), the attacker taps the message from a list to open the link. Meanwhile, swipes should not be injected to avoid the message being scrolled to an unpredicted location.
- **Disable user accessing data.** Some smartphones (e.g., Redmi 5A) facilitate users to set a password to protect the privacy of their data (e.g., message). The user can swipe down the message list to

Table 2: The performance of four exemplary attacks on two different smartphones.

Attack Scenario	Required Exclusiveness	Required Controllability		Attack Rate		Success Rate	
		Nexus 5 (Horizontal TX)	Redmi 5A (Vertical TX)	Nexus 5	Redmi 5A	Nexus 5	Redmi 5A
Establish malicious connection	Nonexclusive	Tap on RX	Tap on TX	67.1%	91.1%	74.0%	100%
Answer an eavesdropping call	Nonexclusive	Swipe along with TX	Swipe perpendicular to TX	3.3%	92.5%	83.3%	100%
Click malicious link	Tap-only	Tap on TX	Tap on RX	96.7%	89.0%	100%	100%
Disable user accessing data	Swipe-only	Swipe perpendicular to TX	Swipe along with TX	50.4%	85.0%	55.9%	98.5%

Table 3: Comparison with the state-of-the-art.

	WIGHT [11]	This Work
Free of reverse engineering	No	Yes
Complexity of CMN waveform	Single packet	Multiple packets
CMN waveform design method	Manually crafted	Automatically generated
Control taps on TX electrodes	Yes	Yes
Control taps on RX electrodes	No	Yes
Inject swipes	No	Yes
Attack Performance (Taps on TX electrodes)		
Average SD_{TX}	21.8 pixels	12.5 pixels
Response Time	0.5~1s	<200ms
Success Rate	83%	100%

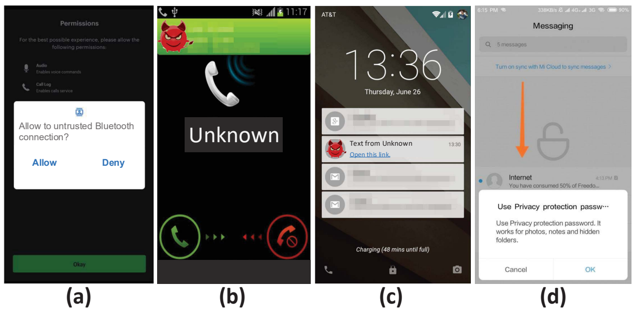

Figure 9: Example user interfaces for four different ghost touch attack scenarios: (a) establish malicious connection, (b) answer an eavesdropping call, (c) click malicious link, and (d) disable user accessing data.

pop up the password setting window shown in Figure 9(d). The attacker can utilize such functions to disable users from accessing data by setting a random password. When injecting the swipes, no taps can be injected. Otherwise, the attacker will exit the message list and enter into one message dialog.

Table 2 lists the attack performance of the four examples when conducting attacks on the two devices. The attack performance is measured with the attack rate and success rate. The former is the percentage rate of CMN strikes recognized as the desired ghost touches. The latter is the ratio of the number of the desired ghost touches to the number of all ghost touches that may affect the attack performance in real scenarios. We identified two factors that affect the attack performance: the UI sensitivity, and TX electrode orientation. In the attack scenarios, there are two levels of UI sensitivity (i.e., only sensitive to taps or swipes, and sensitive to all touches). Correspondingly, to successfully launch an attack, different exclusiveness levels of injected ghost touches are required. The orientation of TX electrodes mainly affects attack performance on different devices. For example, for the first attack scenario, because the two phones have different TX electrode orientations, the required ghost touch controllability is different, leading to different attack performance. For all the scenarios using PowerTouch, an acceptable attack performance can be achieved with an attack rate up to 96.7%, and the success rate up to 100%.

4.4 Comparison with Prior Arts

We compare our work with the state-of-the-art attack [11] as listed in Table 3. Note that for the quantitative metrics comparison, since WIGHT [11] only achieves injecting taps on specific TX electrodes, we compare both works' best performance in attacking smartphones using such capability (i.e., the performance targeting T1~T3 for PowerTouch). The overall comparison shows that we do not need reverse engineering the target device, and can automatically generate the CMN waveform given a specific security objection. In addition, we develop an arbitrary CMN injector and customized GA evolution schemes, making PowerTouch support complex CMN waveform with multiple packets. Therefore, a more fine-granular control is achieved in our work by facilitating control taps on RX electrodes. Note that to the best of our knowledge, there are no existing approaches to controlling ghost touches on RX electrodes. As a result, we found that the attacks demonstrate in previous work [11] are limited to phones with a specific TX orientation, which is overcome by our work. Moreover, we demonstrate the capability of injecting swipes, which is not achieved in the prior work [11]. A better attack performance is also achieved in terms of response time and success rate. In summary, the comparison shows that PowerTouch can significantly improve the attack performance.

5 DISCLOSURE

We have reported PowerTouch to Google, Xiaomi, and LG, including a detailed description of the vulnerability and proof of concept. All three companies have not requested an embargo for the vulnerabilities described in this paper. However, we will add identity checking to the open source process to avoid misuse of our framework. Furthermore, here we outline an overview of potential defense approaches.

6 CONCLUSION

We present PowerTouch, a GA-based framework that enables automatic generation of wired ghost touch attacks via power cables. To the best of our knowledge, this is the first security objective-guided framework to automate ghost touch attacks. Our framework can optimize CMN waveform towards a specific region probabilistically given the objective. A more fine-granular control on the ghost touches and a more advanced attack capability than the state-of-the-art can be achieved without reverse-engineering the touchscreens. We envision that the CMN waveform found by PowerTouch can be interpreted to guide the community to a better understanding of the attack and inspire potential defenses in the future.

ACKNOWLEDGMENTS

This work is supported in part by US National Science Foundation under grants CNS-1837519, CNS-1916926, CNS-2038995, and CNS-1739643, and by Army Research Office under contract W911NF-18-1-0305.

REFERENCES

[1] Jae-Sung An, Sang-Hyun Han, Ju Eon Kim, Dong-Hyun Yoon, Young-Hwan Kim, Han-Hee Hong, Jae-Hun Ye, Sung-Jin Jung, Seung-Hwan Lee, Ji-Yong Jeong, et al. 2017. 9.6 A 3.9 kHz-frame-rate capacitive touch system with pressure/tilt angle expressions of active stylus using multiple-frequency driving method for 65 104× 64 touch screen panel. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 168–169.

[2] Biometrika. 2015. A technical evaluation of fingerprint scanners. http://www.biometrika.it/eng/wp_scfing.html#:~:text=The%20size%20of%20an%20average,area%20smaller%20than%200.5%22x0.. (2015).

[3] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2019. GenUnlock: An automated genetic algorithm framework for unlocking logic encryption. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.

[4] International Electrotechnical Commission. 2014. Testing and measurement techniques - Immunity to conducted disturbances, induced by radio-frequency fields. In *International Standard IEC 61000-4-6: Electromagnetic compatibility (EMC) Part 4-6*. IEC.

[5] Lionel Cordesses. 2004. Direct digital synthesis: A tool for periodic wave generation (part 1). *IEEE Signal processing magazine* 21, 4 (2004), 50–54.

[6] Mircea Dabacan. 2018. Analog Discovery 2 Reference Manual. *Analog Discovery 2 Reference Manual-Digilent Reference* (2018).

[7] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. 2012. Electromagnetic transient faults injection on a hardware and a software implementations of AES. In *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 7–15.

[8] Gerald P Douglas, Oliver J Gadabu, Sabine Joukes, Soyapi Mumba, Michael V McKay, Anne Ben-Smith, Andreas Jahn, Erik J Schouten, Zach Landis Lewis, Joep J van Oosterhout, et al. 2010. Using touchscreen electronic medical record systems to support and monitor national scale-up of antiretroviral therapy in Malawi. *PLoS medicine* 7, 8 (2010), e1000319.

[9] Zacharias Hadjilambrou, Shidhartha Das, Paul N Whatmough, David Bull, and Yiannakis Sazeides. 2019. GeST: An automatic framework for generating CPU stress-tests. In *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 1–10.

[10] Chan-Hwa Hong, Jae-Heon Shin, Byeong-Kwon Ju, Kyung-Hyun Kim, Nae-Man Park, Bo-Sul Kim, and Woo-Seok Cheong. 2013. Index-matched indium tin oxide electrodes for capacitive touch screen panel applications. *Journal of nanoscience and nanotechnology* 13, 11 (2013), 7756–7759.

[11] Yan Jiang, Xiaoyu Ji, Kai Wang, Chen Yan, Richard Mitev, Ahmad-Reza Sadeghi, and Wenyuan Xu. 2022. WIGHT: Wired Ghost Touch Attack on Capacitive Touchscreens. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 1537–1537.

[12] Sheng-Chun Kao and Tushar Krishna. 2020. Gamma: Automating the hw mapping of dnn models on accelerators via genetic algorithm. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–9.

[13] Hans Klein. 2013. Noise Immunity of Touchscreen Devices. In *White Paper*. Cypress Semiconductor.

[14] Oh-Kyong Kwon, Jae-Sung An, and Seong-Kwan Hong. 2018. Capacitive touch systems with styli for touch sensors: A review. *IEEE Sensors journal* 18, 12 (2018), 4832–4846.

[15] Sang-Gug Lee. 2010. A charge-share-based relative read-out circuit for capacitance sensing. In *The SID (Society Information Display) International Symposium*. The SID (Society Information Display) International Symposium, 1458.

[16] Bo Liu, Zaniar Hoseini, Kye-Shin Lee, and Yong-Min Lee. 2015. On-chip touch sensor readout circuit using passive sigma-delta modulator capacitance-to-digital converter. *IEEE Sensors Journal* 15, 7 (2015), 3893–3902.

[17] Yun Long, Edward Lee, Daehyun Kim, and Saibal Mukhopadhyay. 2020. Q-pim: A genetic algorithm based flexible dnn quantization method and application to processing-in-memory platform. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[18] Seita Maruyama, Satohiro Wakabayashi, and Tatsuya Mori. 2019. Tap'n ghost: A compilation of novel attack techniques against smartphone touchscreens. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 620–637.

[19] M Miyamoto. 2013. How to realize high SNR projected capacitive touch systems with very large format. In *Proc. International Display Workshop*. 1630–1633.

[20] PiezoDrive. 2022. MX200 – High Performance Piezo Driver. https://www.piezodrive.com/modules/mx200-high-performance-piezo-driver/. (2022).

[21] Henrik Rydberg. 2010. Multi-Touch Protocol - The Linux Kernel Archives. https://www.kernel.org/doc/Documentation/input/multi-touch-protocol.txt. (2010). Accessed: 2022-03.

[22] Haoqi Shan, Boyi Zhang, Zihao Zhan, Dean Sullivan, Shuo Wang, and Yier Jin. 2022. Invisible Finger: Practical Electromagnetic Interference Attack on Touchscreen-based Electronic Devices. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 1548–1548.

[23] Hyungcheol Shin, Seunghoon Ko, Hongjae Jang, Ilhyun Yun, and Kwyro Lee. 2013. A 55dB SNR with 240Hz frame scan rate mutual capacitor 30x24 touch-screen panel read-out IC using code-division multiple sensing technique. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*. IEEE, 388–389.

[24] Gilbert Tang and Phil Webb. 2018. The design and evaluation of an ergonomic contactless gesture control system for industrial robots. *Journal of Robotics* 2018 (2018).

[25] Kai Wang, Richard Mitev, Chen Yan, Xiaoyu Ji, Ahmad-Reza Sadeghi, and Wenyuan Xu. 2022. GhostTouch: Targeted Attacks on Touchscreens without Physical Touch. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association.

[26] Darrell Whitley. 1994. A genetic algorithm tutorial. *Statistics and computing* 4, 2 (1994), 65–85.

[27] Wikipedia. 2022. Logistic function. https://en.wikipedia.org/wiki/Logistic_function#cite_note-verhulst1838-1. (2022).

[28] Wikipedia. 2022. USB. https://en.wikipedia.org/wiki/USB. (2022). Accessed: 2022-05.

[29] Jun-Hyeok Yang, Seung-Chul Jung, Young-Suk Son, Seung-Tak Ryu, and Gyu-Hyeong Cho. 2013. A noise-immune high-speed readout circuit for in-cell touch screen panels. *IEEE Transactions on Circuits and Systems I: Regular Papers* 60, 7 (2013), 1800–1809.

[30] Huifeng Zhu, Xiaolong Guo, Yier Jin, and Xuan Zhang. 2020. PowerScout: A security-oriented power delivery network modeling framework for cross-domain side-channel analysis. In *2020 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. IEEE, 1–6.

[31] Huifeng Zhu, Haoqi Shan, Dean Sullivan, Xiaolong Guo, Yier Jin, and Xuan Zhang. 2022. PDNPulse: Sensing PCB Anomaly with the Intrinsic Power Delivery Network. *arXiv preprint arXiv:2204.02482* (2022).