Distinguishing Users with Capacitive Touch Communication

Tam Vu, Akash Baid, Simon Gao, Marco Gruteser, Richard Howard, Janne Lindqvist, Predrag Spasojevic, Jeffrey Walling
WINLAB, Rutgers University
{tamvu, baid, gruteser, reh, janne, spasojev} @winlab.rutgers.edu
{simongao, jeffrey.s.walling}@rutgers.edu

ABSTRACT

As we are surrounded by an ever-larger variety of post-PC devices, the traditional methods for identifying and authenticating users have become cumbersome and time-consuming. In this paper, we present a capacitive communication method through which a device can recognize who is interacting with it. This method exploits the capacitive touchscreens, which are now used in laptops, phones, and tablets, as a signal receiver. The signal that identifies the user can be generated by a small transmitter embedded into a ring, watch, or other artifact carried on the human body. We explore two example system designs with a low-power continuous transmitter that communicates through the skin and a signet ring that needs to be touched to the screen. Experiments with our prototype transmitter and tablet receiver show that capacitive communication through a touchscreen is possible, even without hardware or firmware modifications on a receiver. This latter approach imposes severe limits on the data rate, but the rate is sufficient for differentiating users in multiplayer tablet games or parental control applications. Controlled experiments with a signal generator also indicate that future designs may be able to achieve datarates that are useful for providing less obtrusive authentication with similar assurance as PIN codes or swipe patterns commonly used on smartphones today.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communication*

General Terms

Design, Human Factors, Measurement, Experimentation, Performance, Algorithms

Keywords

SignetRing, Touchscreen Communication, User Identification, Capacitive Touch Communication, Distinguishing Users

MobiCom'12, August 22–26, 2012, Istanbul, Turkey.

1. INTRODUCTION

Mobile devices now provide us ubiquitous access to a vast array of media content and digital services. They can access our emails and personal photos, open our cars [41] or our garage doors [13], pay bills and transfer funds between our bank accounts, order merchandise, as well as control our homes [10]. Arguably, they now provide the de-facto single-sign on access to all our content and services, which has proven so elusive on the web.

As we increasingly rely on a variety of such devices, we tend to quickly switch between them and temporarily share them with others [26]. We may let our children play games on our smartphones or share a tablet with colleagues or family members. Sometimes a device may be used by several persons simultaneously, as when playing a multi-player game on a tablet, and occasionally, a device might fall into the hands of strangers.

In all these situations, it would be of great benefit for the device to know who is interacting with it and occasionally to authenticate the user. We may want to limit access to age-appropriate games and media for our children or prevent them from charging our credit card.[1] We desire to hide sensitive personal information from strangers, colleagues, or perhaps even an curious spouse [23, 26]. Or, we may simply want to enjoy an enhanced user experience from the multi-player game that can tell who touched the screen.

Unfortunately, user identification and authentication mechanisms available on today's mobile devices have been largely adopted from PC software and have not followed the versatility of the usage and sharing possibilities. For example, several mobile devices (e.g. iPad or iOS devices) do allow to restrict access to device functions, but the devices do not provide any easy way to quickly change, let alone authenticate, users. They provide PIN codes, passwords, for authentication, and a number of other techniques have been proposed by researchers [11]. Yet they remain cumbersome and very few people enable these security features on their phones.

In this paper, we will explore a form of "wireless" communication, that we term *capacitive touch communication* to address this issue. The key idea is to exploit the pervasive capacitive touch screen and touchpad input devices as receivers for an identification code transmitted by a hardware identification token. While the token can take many forms, we consider here an example realization as a ring, inspired by the signet rings used since ancient times. The token transmits electrical signals on contact with the screen, either direct contact or indirect contact through the human skin.

The major contributions of this paper are as follows:

- painting a vision to use the near-ubiquitous capacitative touch sensors to distinguish and possibly authenticate users.

[1]Apple is facing a law suit over children's in-app credit card purchases [18].

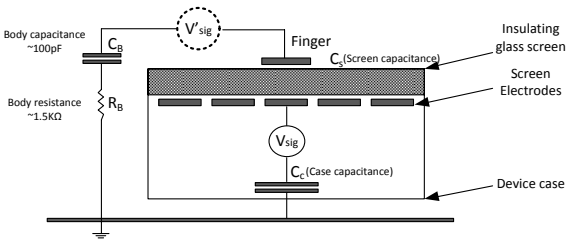


Figure 1: Schematic of a basic capacitive touchscreen

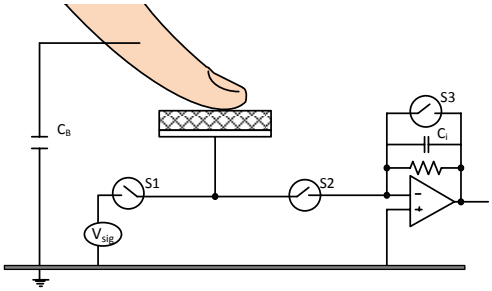


Figure 2: Internal touch detection circuit

- introducing and exploring the concept of capacitive touch communication as one mechanism to distinguish users.
- showing how the output of an off-the-shelf touchscreen system can be affected by electrical signals generated in a token that is in contact with the screen. We also show how such signals can be transmitted through the human skin.
- designing and implementing a prototype transmitter in the form of a signet ring and receiver software for communicating short codes through an off-the-shelf capacitative touch screen

2. BACKGROUND

Touchscreen technology was first developed in the 1960's for air traffic control systems [25] and is now a popular user interface technology on devices ranging from ATMs and self-service terminals in grocery stores or airports, to cars, smartphones, and tablets. Even the touchpads used in laptops are based on similar technology. These products employ different touchscreen implementations, including analog resistive, surface capacitive, projected capacitive, surface acoustic wave, infrared and optical technology to mention a few. On mobile devices, however, capacitive touchscreens have emerged as the main technology and we focus our work on those.

2.1 Capacitive Touchscreen Technology

A capacitive screen in most commercial tablets and smart phones consists of an array of conducting electrodes behind a transparent, insulating glass layer which detects a touch by measuring the additional capacitance of a human body in the circuit. Figure 1 shows a schematic of one possible realization of such a system [47]. When a user touches the screen, her finger acts as the second electrode in a capacitor with the screen as the dielectric. The touchscreen electrodes are driven by an AC signal (V_{sig}) which sends a current through the screen capacitance C_s passing through the body capacitance C_B, and then back into the tablet through the case capacitance C_c. This change in voltage measured at one or more screen electrodes is then passed to the screen controller for processing. Because all of the relevant capacitance values are small (hundreds of picofarads [19]) environmental noise makes direct measurement of this current impractical. Instead, the charge integration circuitry in Figure 2 is used to measure the excess capacitance associated with a finger touch. In this case, a digital signal, V_{sig}, is synchronized with a pair of switches and a charge integrator. Switch S_3 is first closed to discharge capacitor C_i and then opened. Next, switch S_1 is closed and S_2 opened while V_{sig} is high. This charges the series combination of the C_B, C_c, and C_s. Then S_1 is opened and S_2 closed, transferring this charge to C_i. After a fixed number of cycles, the voltage on C_i is directly proportional to the ratio between C_i and the series combination of C_B, C_c, and C_s. This voltage is then used to detect touch and, through the matrix addressing of the electrodes, position of the touch. Hence, even when a finger moved across the screen surface without lifting it, the finger triggers this detection at different positions on the electrode array.

2.2 Related Work

The most closely related projects to our work are Touché [39], DiamondTouch [14], Signet [44], IR Ring [38], Magkey/Mickey [12]. Proposed in 2001 as one of the first efforts toward differentiating touches of different users interacting with the same surface, DiamondTouch uses a table to transmit capacitively coupled signals through users, chairs, and finally to the receiver. This approach requires extensive hardware infrastructure which make it impossible to apply to mobile scenarios. Touché proposes a technique, called Swept Frequency Capacitive Sensing, that can recognize human hand and body configurations. While the technique could enable a new way of human computer interaction, it would require additional special hardware component to be manufactured onto the devices. Signet uses physical patterns of conductive material as unique inputs for authentication through a capacitive touch screen. In contrast, our work focuses on using arbitrary programmable sequences of bits through direct use of the user's fingers. As such, it makes the solution non-intrusive and applicable to wider classes of applications.

There are several ways to authenticate a user, which in general can be divided into 1) what you know, 2) what you have, and 3) who you are. PINs, passwords and swipe patterns are the most widely spread authentication mechanism for mobile phones [11, 16]. These methods are easy to implement and require no special hardware, but are easily observable by an adversary and usually have very low information entropy. For example the usual 4 bit numeric PINs used in most phones have a theoretical potential entropy of $\log_2(10^4) = 13.3$ bits. Practical entropy for 4-digit PINs is likely to be much lower, as is the case with passwords [46]. The second type of authentication mechanisms ("what you have") are often also referred to as authentication tokens, examples include Magkey/Mickey [12], RFID or other wireless tokens such as transient authentication [36], and IR Ring [38]. Magkey and Mickey are tokens that use magnetic fields and acoustic signals that are received by the phone's compass and microphone respectively. RFID, NFC and other wireless-based techniques are prone to eavesdropping and suffer from interference among multiple radio signal sources. One example technique belonging to that category is RingBow [37], a wearable hardware token in the form of a ring, which communicates with the mobile device using Bluetooth. This type of communication is insecure during the pairing period and does not allow touchscreen-enabled devices to associate touch events to their users. And finally, IR Ring demonstrated the possibility to use infra-red and IR video cameras to authenticate users

on a multitouch display, which is not directly applicable to today's mobile devices due to its additional hardware requirement.

Examples of "who you are" include iris recognition, face recognition and voice recognition all of which are being actively prototyped and tested on mobile devices. Motorola Atrix claims to be the first phone in the western market to have a fingerprint sensor [34] while Sony is developing a novel finger-vein pattern matching technique [42]. Both these techniques require specialized hardware which adds to the cost and form-factor of handheld devices and are prone to known vulnerabilities [30, 20]. On the other hand, face, iris and voice recognition utilizes the in-built sensors and most of the feature set required are already implemented in mobile devices for other applications [27, 33]. While these techniques can leverage the abundance of past research in the respective fields, they also suffer from the well known spoofing mechanisms [17, 3]. For example both high-quality photograph of the eye and printed contact lenses have been used to achieve close to 100% spoof acceptance rates for iris recognition systems [45].[2] Similar results hold for face detection and voice detection although large strides are also being made for spoof detection in biometric authentication systems (see Jain et al. [24] and the references therein). More recently, innovative uses of the various sensors available in most smart phones have led to a number of *unconventional techniques*. For example, there are proposals [8, 28] for in-air gesture based authentication mechanism which uses the accelerometer sensors of the mobile device. Being easily visible to an adversary, such a scheme suffers from an unpleasant tradeoff between coming up with complex gestures and being susceptible to copy attacks, and can also be socially awkward. Implicit authentication is a similar approach which aims to authenticate mobile users based on everyday actions such as number/duration of calls, location, connectivity pattern, etc. and keeps a multi-variable continuous authentication score of the user. As is obvious, this requires a continuous modelling and logging of data from a variety of sensors and has a high energy cost.

Today's consumer electronic devices often include some form of parental control mechanisms, which are usually limited to locking out some functionalities of the device or service, e.g. adult content. Parental control mechanisms are an overlooked area of research, however, recent studies indicate that there would be demand for flexible access control mechanisms at home [32]. Our presented work can be seen as an easy to use enabler for parental access control mechanisms.

The problem of device pairing is also closely related to secure authentication and solution approaches often overlap. The general objective in this case is to enable two devices with no prior context to securely associate with each other in the presence of man-in-the-middle adversary. The short-range and frequency hopping nature of Bluetooth makes it a robust authentication mechanism, however several recent works expose a key vulnerability - passive sniffing of the PIN during the pairing process [40]. Similarly, for near-field communications (NFC) [2] based pairing, eavesdropping using directional antennas has been shown to be a critical security threat [22]. Novel use of the accelerometer sensor in mobile devices have recently been shown to provide a secure method of device pairing [31]. While robust for two equipped mobile devices, the requirement of shaking prevents its use from cases which require pairing of a mobile device with a fixed device. Further, replication of the movement by an adversary is possible through careful observation of the pairing process. Finally, a recent approach uses public RF signals such as TV and FM broadcasts to derive cryptographic keys for secure pairing between close-by devices [29].

[2]The face recognition system available in the new Google Android based Galaxy Nexus platform can be compromised just by showing a picture taken with another smartphone [1].

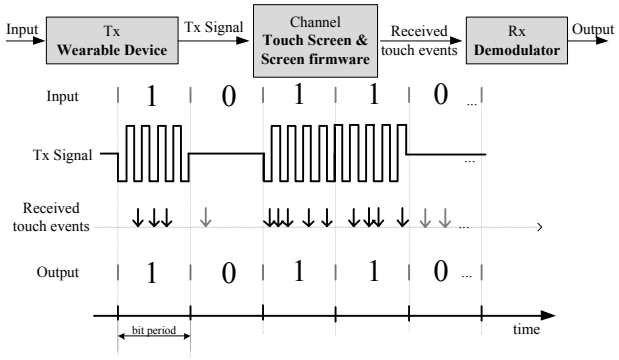


Figure 3: Capacitive touch screen communication showing OOK modulation and variations in number and timing of generated events

Auxiliary channels to established shared secrets have been studied extensively in the domain of secure pairing since the resurrecting duckling model [43]. Examples include using infrared [9] or humans [21]. More recently secure pairing efforts have focused on using the same channel for authentication and data, and deriving the keying material based on the local environment, e.g [29]. In contrast, our approach provides a seamless way to both securely pair the device and authenticate later.

3. CAPACITIVE TOUCH COMMUNICATION

To allow mobile devices to identify their users in a less obtrusive manner, we explore a novel form of "wireless" communication in which a touch panel acts as a receiver and a small ring-like device worn by the user serves as the transmitter. This type of communication, which we term *capacitive touch communication*, could have wide applicability since touch panels are now ubiquitous.

While it would be interesting to also consider modifications to the touch sensor hardware and firmware to facilitate such communication, we focus this first study on exploring to what extend the communication can be achieved with off-the-shelf touch sensor systems. This means we will only have access to the touch events exported by the screen's driver, not the raw voltages measurements. It imposes very stringent requirements on the communication protocols, as we will see in the next sections. We believe, however, that this is a useful point solution within the design space of capacitive touch communication, since this approach would allow more rapid deployment on existing devices.

3.1 Creating Artificial Touch Events

Motivated by this goal, we discovered a technique for "spoofing" the screen detection algorithm by causing the system to alternately register touch/no touch conditions even when the finger is not moving. This allows us to send a digital signal into the touchscreen.

Referring again to Figure 1, one possible method for artificially creating touch events is by injecting a synchronized signal (V'_{sig}) into the circuit with the proper amplitude and phase to increase or decrease the charge integrated on C_i. Unfortunately, the signal in the device, V_{sig}, is not available to the external user, so such synchronization would be extremely difficult. As such, we use an unsynchronized lower frequency signal of high amplitude which

Figure 4: Touch event structure retrieved by touch screen controller

charges and discharges C_i asynchronously, leading to repetitive, but irregular, touch/no touch events captured by the touchscreen controller. This process essentially "spoofs" the touch detection mechanism by injecting high level repetitive signals and introduces a technique to send a low bit rate signal into the tablet. With precise knowledge of the proprietary touch sensor systems it should be possible to create much more fine-grained signaling methods. For the purpose of this feasibility study, however, we will now consider how this coarse technique can be leveraged for designing a user identification system.

3.2 Communication System Overview

The communication scheme we are proposing can be modeled as a classical communication system with a transmitter, a receiver and a complex channel connecting the two, as shown in Figure 3.

Transmitter: The transmitter in our system is a wearable battery-powered hardware token. One possible form that such a token could take is that of a ring, essentially a digital version of the signet rings carried by nobility in earlier times[3]. While many other forms of tokens are possible, we will use the *ring* concept as a running example throughout the paper.

The ring would contain a small flash memory that stores a bit sequence or a message, which could be a user identifier or a secret key that authenticates a user. It also has a simple processor that reads the bit sequence and generates a On-Off keying (OOK) [15] modulated signal. That is, bit *one* is represented by turning *on* a carrier signal; and bit $zero$ by switching *off* that carrier signal, as the *Tx Signal* shown in Figure 3. When the ring is pressed against the screen, it acts as a voltage source(V'_{sig} in figure 1) which creates a set of touch events with timestamps following the bit sequence being transmitted.

Channel: Since the events generated follow the bit sequence being transmitted, these events can be used to reconstruct the original bit sequence, which is unknown to the screen otherwise. Thus, in this setting, the channel can be thought of as the combination of all hardware and software components that affect the relationship between the transmitted bit sequence and the events registered: (i) the series of capacitances, (ii) the firmware that comes with the screen, and (iii) the proprietary driver that is a part of the device's operating system.

Unfortunately, due to the internal switching frequency inside the touch panel, non-deterministic amount of charge accumulation and the firmware/driver artifacts, the number and the timing of the events do not directly follow the input sequence. For example, in figure 3, when the first bit one is transmitted, three touch events are triggered, while in the succeeding $ones$ five and four events are produced. Furthermore, even though transmission of a $zero$ should not trigger touch events, one and two events are registered in the two $zeros$ presented in this example respectively. In addition, the channel adds a variable and unknown delay between the transmitted sequence and the touch event registered.

Receiver: The *Tx Signal* transmitted by the ring generates touch events represented by the 6-tuple structure depicted in Figure 4 (a

[3]A finger ring bearing a hard-to-fake engraved pattern, which serves as a seal of authority, a signet.

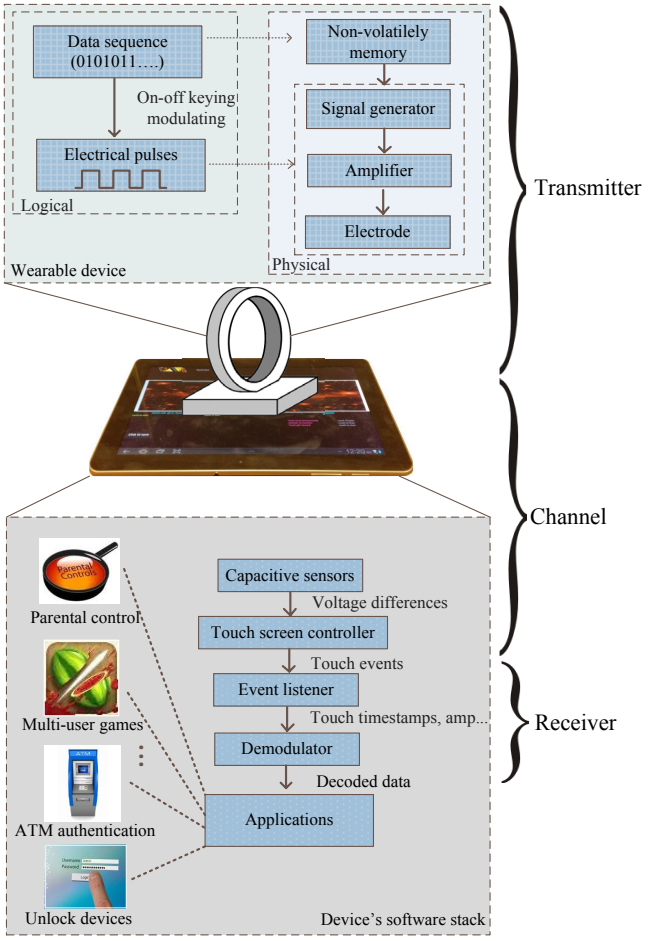


Figure 5: Overall architecture of the capacitive touch communication system

detailed description of this structure is presented in Section 5). Because the only information we can use is the timestamps of the events registered by the screen driver, the system requires an unconventional receiver design. Instead of the usual practice of looking at the amplitude (*Touch Amplitude* field) of the received signal, which in this case is not related to the transmitted data, we use the *number of event registered* for demodulating. That is, the software component receives a bit 1 if the number of events which appeared in that bit period is greater than a certain threshold and receives a bit 0 otherwise.

We note that there is a variable delay from the moment that touch events were registered to the kernel until it is handed to the application-level software, which in our case is the application-level demodulator. This delay makes demodulating less accurate. The time variance, we suspect, is due to the queueing and processing delays incurred when the event information travels up the software stack, from the touch-event handler in the Android kernel to the application level. To mitigate this inaccuracy, our demodulator looks at the touch event timestamps at the kernel level (using a few *printk* commands in the touchscreen driver of our prototype).

The key challenge is to handle the variance in the number and timing of the events that is introduced by the channel. To address this issue, we characterize the expected behaviour of the channel, reflected in terms of event counts, for decoding of the received se-

Bit Rate (bps)	4	5	8	10	12	15
Expected no. of events in $ones$ (1e)	11.3	9.2	5.8	4.5	3.6	3.3
Expected no. of events in $zeros$ (0e)	2.0	1.6	1.2	1.0	0.7	0.7
One-Zero threshold	7	6	4	4	2	2

Table 1: One-Zero threshold and expected number of events in bit one and $zero$ for different bit rates

quence, as described in Section 4.1. Specifically, we apply a joint decoding-synchronization technique that uses a threshold-based and distance-based method to simultaneously synchronize and decode the received sequence.

Figure 5 shows the high level architecture of the system and how components interact. Note that since we do not have access to the touchscreen controller, the touch events, not the underlying physical voltage differences, are the input to the receiver.

Indirect Communication: Even without direct contact with the screen, the ring can communicate with the touchscreen device as long as the ring bearing finger is in contact with the screen. In particular, the electrical pulses that are transmitted through a human finger's skin from the ring create the same effect of changing the screen capacitance to register artificial touch events. However, we found that due to the skin resistance, the number of events generated through this type of indirect contact is only enough for detecting the presence of the ring, but not stable and regular enough for reliably decoding the data being transmitted. We can leverage this capability of the communication system to enable a novel technique to differentiate two users simultaneously interacting with the same touchscreen, for example in a shared-screen two player game. The detection algorithm used for this mode of communication is described in Section 4.3.

4. DECODER DESIGN

The proposed capacitive touch communication system allows users to send messages to the application layer of the device. This unconventional use of the touchscreen, especially under the constraint of using commercial off-the-shelf devices without lower layer access, poses a number of challenges:

1. We observed that the receiver responds differently to the same input following a different bit pattern; this could be due either to the physical layer or the software that is optimized for detecting touch events from a human finger. For example, the number of events registered to the screen when bit 1 is sent after a long sequence of 0s is different from that of a bit 1 that comes after a sequence of 1s. The normal solution is to code the data to avoid this pattern dependent effect. Rather than adopting a typical bit-by-bit decoding solution, our data rate is already so limited that we developed our own code optimized specifically for the observed pattern dependence.

2. There is a variable delay between the transmission of a symbol and its reception at the receiver after processing through all layers of firmware and software. This jitter significantly increases the difficulty of detection. Since the communication channel has low bandwidth and high jitter, no traditional symbol synchronization schemes can be directly applied. We overcome the bit synchronization challenge by simultaneously synchronizing and demodulating the signal.

3. The channel adds an unknown delay between receiver and transmitter; this problem is classically solved using a frame synchronization which requires using a preamble. Since we have a low bandwidth channel and would like to transmit the message in only a few seconds, the message can only include limited number of bits. Thus, we can not afford to add the preamble. Instead, we use constrained bit patterns that are unique under cyclic shifts caused by unsynchronized frames.

The conversion from touch events to a sequence of binary digits is based on the principle of On-Off keying; the touchscreen driver produces several events when a binary one is transmitted and only a few events when a $zero$ is transmitted. The key challenge is to handle the variance in the number of events associated with $ones$ and $zeros$. In the coming sections, we describe an off-line calibration procedure to characterize the expected behaviour of the channel, which is then used in the online phase to classify touch responses as $zero$ or one transmissions. Once a sequence of bits is decoded, we use a "closeness" metric to determine the distance of the received message from the set of all possible messages of the same length. This process corrects for uncertainty in timing and event number. Details about the design of the closeness metric and the decoding process are presented in the next sections.

Algorithm 1: Threshold selection algorithm

input : $E_{discrete}$ - Event sequence in time domain
$TxBitSeq$ - Original transmitted bit sequence
$BitRate$ - Transmission bit rate (bps)
output: $1e$ and $0e$ - Expected number of events in $ones$ and $zeros$

bitPeriod $\leftarrow \frac{1000}{BitRate}$
oneC $\leftarrow$ 0 // Event counter for all ones
zeroC $\leftarrow$ 0 // Event counter for all zeros
//Convert discrete events to event vector in time series
for $i = 1 \rightarrow max(E_{discrete})$ **do**
  if *exist* $E_{discrete}[j] == i$ **then** $E_t[i]$= 1
  else $E_t[i]$ = 0
//Find the starting position that gives the max $1e0e$ Ratio
for startPos =1 $\rightarrow$bitPeriod **do**
  for j =1 $\rightarrow$ *length*$(TxBitSeq)$ **do**
    eCount = sum(E_t[startPos + (j $-$ 1) $*$ bitPeriod, startPos + j $*$ bitPeriod])
    if $TxBitSeq(j) == 1$ **then**
    oneC = oneC + eCount
    else zeroC = zeroC + eCount
  // Update $1e0e$ Ratio
  current$1e$ = oneC/$no.\ bit\ 1\ in\ TxBitSeq$
  current$0e$ = zeroC/$no.\ bit\ 0\ in\ TxBitSeq$
  if *current*$1e$/*current*$0e >$ maxRatio **then**
  maxRatio = current$1e$/current$0e$
  $1e$ =current$1e$; $0e$= current$0e$
return $1e$ and $0e$;

4.1 Determination of Expected Number of Events for $ones$ and $zeros$

To determine the number of touch events associated with a one or $zero$, it is necessary to calibrate the device at each data rate before use. This calibration to determine thresholds only needs to be performed once per device, at initialization; thereafter it can be stored in a lookup table and adjusted during self calibration depending on an estimate of the data rate of the incoming data sequence or fetched as an input from applications. To determine the counting threshold for each data rate, a sequence of $ones$ and $zeros$ is repeatedly transmitted in a prescribed pattern. On the receiver side, event sequence is detected and recorded to a log file. Threshold selection algorithm, algorithm 1, takes the log file and the prescribed pattern as input to compute the two expected counter thresholds $1e$

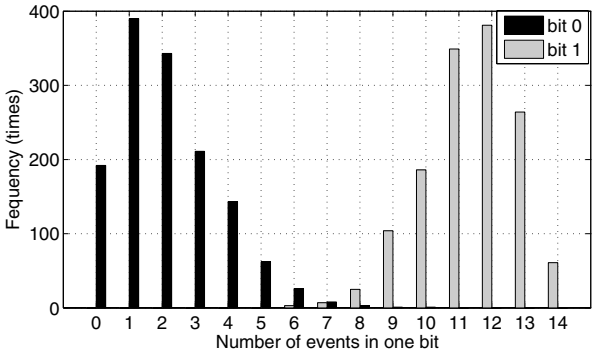


Figure 6: Number of events in bit one and bit $zero$ for transmissions at 4 bits/s

and $0e$. We devise the algorithm 1 to simultaneously demodulate the received event sequence and find the bit starting point. The intuition behind the algorithm is that the correct bit synchronization maximizes total number of events in all $ones$ and minimize number of event in all $zeros$ We define *1e0e* ratio as being the normalized ratio between the *total number of events in all ones* and *total number of events in all zeros* :

$$1e0e\ Ratio = \frac{\frac{\Sigma(Event\ Counters\ in\ Ones)}{Number\ of\ Ones}}{\frac{\Sigma(Event\ Counters\ in\ Zeros)}{Number\ of\ Zeros}}$$

This ratio is maximized when bit synchronization is correct. The ideal synchronization, for example, should have total number of events in all $zeros$ close to 0, and number of events in all $ones$ close to the total number of events in the whole event sequence, in which case *1e0e* ratio reaches its maximum.

Algorithm 2: Min Distance Demodulation Algorithm

input : E_t - Event sequence in time domain
$BitRate$ - Transmission bit rate (bps)
$MessageLength$ - Original message length
$PosMessageVec$ - Possible message vector
$1e$ and $0e$ - Expected number of events in ones and zeros
output: $RxBitSeq$ - Received bit sequence

```
bitPeriod ← 1000/BitRate
minDistance ← MAX-INT
for startPos =1 →bitPeriod do
    foreach message in PosMessageVec do
        rotatedMesgVec =getAllCyclicVersions(message)
        for j =1 → MessageLength do
            eCount[j ] = sum(E_t[startPos + (j − 1) ∗
            bitPeriod, startPos + j ∗ bitPeriod])
        foreach rotatedInstance in rotatedMesgVec do
            currentDist = 0;
            foreach i^th bit in rotatedInstance do
                if i^th bit == 1 then
                currentDist = currentDist + max(0, 1e - eCount[i])
                else
                currentDist = currentDist + max(0,eCount[i] - 0e)
            // Update Min distance
            if currentDistance < minDistance then
                minDistance = currentDist;
                RxCandidate = rotatedInstance;
return RxCandidate
```

Algorithm 1 first converts the discrete timing event information to a event/no-event time series data. That is, if the received sequence of event is $E_{discrete} = \{E_1, E_2, ..., E_m\}$ in which E_i is i^{th} event, it will be represented by a vector in the form: $Et = [Et_1, Et_2, ...Et_{t_{max}}]$ where $Et_i = 1$ if there exists an event E_k such that $E_k = i$, and 0 otherwise. In the second step, the algorithm tries all possible bit starting points within the first bit period, with each trial involving a counting of the number of events in all bit periods of the sequence. The starting point that leads to the highest ratio is considered the correct bit sync position, while the bit sequence corresponding to that starting point is the demodulated result of the event sequence. At the end of this process, since the total number of events in all $ones$ and total number of events in all $zeros$ is found, the expected number of events in $ones$ and $zeros$, $1e$ and $0e$ can be derived and stored in memory for future demodulation. Figure 6 shows the distribution of the number of touch events registered corresponding to the transmissions of $zero$ and one evaluated by using algorithm 1 for a 3000 bit sequence of alternating $zeros$ and $ones$. The variations due to the transmission bit rate is recorded in table 1, which shows that the event count threshold required for decoding varies from 7 events for 4 bits/s to 2 events for 15 bits/s.

4.2 Minimum Distance Demodulation

Using the counter thresholds from the previous section, algorithm 2 demodulates the timing event sequence to get the data sequence sent by the transmitter. Sharing the same synchronization challenge with the threshold detection algorithm, this algorithm has to detect the point in time at which the data is transmitted. At the same time, it demodulates the event sequence to get the information that has been transmitted. Note that simply relying on the first event to determine the starting point is not enough since there is a fair amount of timing uncertainty in the communication channel. Intuitively, the algorithm traverses the sequence to try all possible starting points. At each point, it gauges the "distance" between the event sequence and all messages. It then ranks the positions with "similarity" value and selects the one that has highest "similarity" index. The message corresponding to that index will be the decoded value of the event sequence.

So the question remains as to how to measure the similarity between two sequences. We define a distance metric as following: let $D(i, j)$ be the distance between a event sequence that has a starting point at point i and the message, K_j, with $j = 1..number\ of\ messages$. Using the same notations as defined in the previous algorithm, in which $Et = [Et_1, Et_2, ...Et_{t_{max}}]$ is the event vector re-sampled along the time domain, an event counter, eC_p, for bit at p_{th} position from the starting point can be computed by:

$$eC_p = \sum_{q=(p-1)*bit\ period}^{p*bit\ period} Et_q$$

Then distance $D(i, j)$ can be derived as:

$$D(i,j) = \sum_{k=1}^{message\ length} d_k$$

with

$$d_k = \begin{cases} max(0, eC_p - 0e) & \text{if the } k^{th} \text{ bit on message } K_j \text{ is 0} \\ max(0, 1e - eC_p) & \text{if the } k^{th} \text{ bit on message } K_j \text{ is 1} \end{cases}$$

We note that since messages are cyclically transmitted, the algorithm does not only compute the distance of a sequence to a message but it does so for all unique *rotated* version of that message.

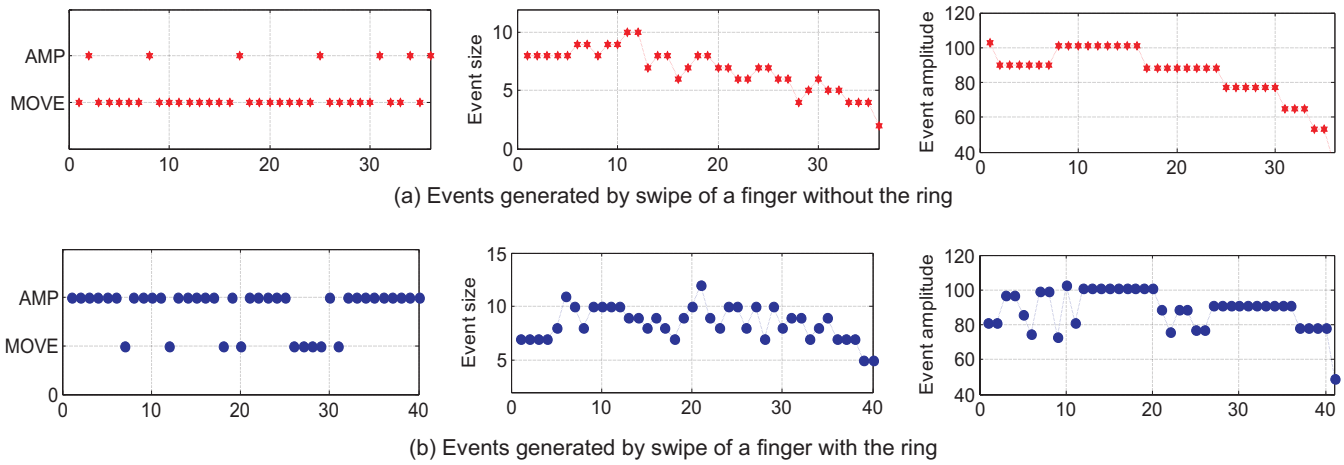


Figure 7: *Type, Size* and *Apmlitude* values generated from finger swipes with and without the ring. A ring bearing finger produces many *AMP* events while swipe without ring induces correlation between *Size* and *Amplitude*

The intuition behind this metric is that it rewards starting points that make the decoded sequence look similar to one of the messages in the message vector. The smaller the distance, the closer the decoded sequence to the message. Hence, smallest $D(i, j)$ will tell which position on the sequence is correct synchronization position and which message is the event sequence representing.

We note that when the number of possible messages is small (order of hundreds), it is feasible to apply Algorithm 2 to exhaustively search through the whole message space to demodulate. However, when the number of possible messages is large, the above exhaustive algorithm can become prohibitively expensive or impossible. In such cases, more efficient algorithm assuming no knowledge of the message becomes handy. That algorithm shares the same intuition with Algorithm 1, in that it tries all possible starting points. However, at each possible position, it directly converts the sequence to data bit sequence by counting number of events in each bit period and select the one that yields the highest $1e0e$ ratio.

Other demodulation schemes. In the process of finding the most suitable demodulation scheme, we experimented with several other demodulations schemes such as *Non-thresholding modulation*, $1e0e$ *ratio demodulation* and *maximum key correlation*. Non-thresholding modulation scheme does not require any training to learn expected number of events in zeros (*0e*) and ones (*1e*), instead it looks at all possible starting positions and compares them with all possible keys to find the best match. It simultaneously picks the synchronization point and decodes the sequence of events by selecting the starting point that gives the highest correlation with one of the possible keys. The maximum key correlation method takes an approach that is similar to minimum distance modulation but has a different evaluation function. For that we defined our own correlation coefficient function to take the noisy channel into account. The function gives one point to a bit that is equal to the bit at the same position on the correct key. It gives half a point to the bit that is not correctly decoded but has a number of events close to the $One - Zero$ threshold. Lastly, $1e0e$ ratio demodulation becomes useful when the possible message space is unknown or so large that it is prohibitively expensive to conduct an exhaustive search to find minimum distance or maximum correlation. The Min Distance Algorithm presented earlier outperformed these techniques in our experiments.

4.3 Ring Detection for Indirect Communication

As mentioned in Section 3, an indirect mode of communication is enabled when instead of the ring, a ring bearing finger is in direct contact with the touchscreen. In such cases, only the presence of a ring needs to be detected. However detecting the ring in the presence of finger movements (or finger swipes) is challenging since the events generated due to the movement of the finger and those by the ring cannot be easily distinguished.

Figure 7 shows three fields of the touch event outputs: *Type*, *Size* and *Amplitude*, generated when a user swipes a finger across the screen with and without the ring. We leverage two key observations from the patterns observed for designing a robust detection algorithm: (i) events generated by the finger movement without the ring are mostly of type *MOVE* while those generated by the ring are mostly of type *AMP*, however due to the excess pressure exerted from the drag force of the finger on the touchscreen, a few *AMP* events can also be generated during finger swipe movements without the ring; (ii) in the absence of the ring, the sequence of *Size* and *Amplitude* values are correlated since increasing the pressure brings more surface area of the finger in contact with the screen. We confirm these observations by collecting data from a large number of swipe movements, both with and without the ring from 5 different users.

Since both the presence of a large number of *AMP* events and the absence of correlation between *Size* and *Amplitude* indicate the presence of a ring, we define a metric p_{ring}, which relates to the normalized number of *AMP* events registered (n_{amp}) and the correlation coefficient between the *Size* and *Amplitude* values (c_{SA}) as:

$$p_{ring} = \alpha \times n_{amp} + (1 - \alpha) \times (1 - c_{SA})$$

where $\alpha \in [0, 1]$ is parameter which signifies the relative contributions of n_{amp} and c_{SA} in determining the p_{ring} value. Given a set of generated events, a detection threshold λ_{th} is then used on the p_{ring} value to classify the presence or absence of the ring. We determined the values of the two parameters α and λ_{th} through a training set consisting of 1000 swipes from 3 different users, using traditional least square minimization. After the training, α and λ_{th} were determined to be 0.83 and 0.5 respectively.

5. EXPLORING THE PARAMETER SPACE

Data transmission using capacitive touchscreen communication is an unexplored mode of communication. In this section, we explore the dynamic ranges of frequency, voltage and signal types that can be used for triggering usable events through the screen driver. Having picked the most suitable set of parameters, we then study the performance of the communication system for different use cases.

In order to conveniently vary the input signal parameters, as required in this analysis, we placed a flat, rounded copper piece on the screen surface of a Samsung Galaxy Tab 10.1, which uses a Atmel maxTouch touch panel [7], and attached it to the output of a AFG 3000 Series function generator [4]. This setup simulates the touch of the ring on the screen surface while offering two main benefits over the battery-powered prototype described in Section 6: (i) it alleviates the microcontroller's limitation in generating arbitrary waveforms and (ii) it greatly expands the scale of repeated experiments (order of tens of thousands of logging runs) which would be otherwise limited by time and human effort.

5.1 Triggering Touch Events

The inner-workings of the touch screen are proprietary and not available for use in designing either our hardware or software. A main task is to determine what type of electrical signal will be interpreted as a touch event when it is injected into the touchscreen. To answer this, we inject different signals from a function generator through an attached electrode approximately the size of a finger to the surface of the touchscreen.

The touch events retrieved by the tablet's operating system, Android 3.2, are represented in a 6-tuple structure depicted in Figure 4. Indicated through *Event Type* field, touches are classified into one of the following types: *MOVE*, *AMP*, *MOVEAMP*, *PRESS*, *RELEASE* and *SUPPRESS*. For example, a *MOVEAMP* event is registered when both touch pressure and X,Y-coordinates change at the same time; and a *SUPPRESS* event happens when the touch pressure exceeds a predefined threshold. Note that such touch events are triggered when a finger first touches the panel, when the position of the finger on the screen changes, when the pressure changes, and when the finger leaves the screen. *Touch Size* and *Touch Amplitude* specify the size and amplitude of the touch respectively. *Pointer ID* is used to differentiate the presence of two or more points of contact at the same time, or multi-touch. A physical touch causes voltage changes at many different electrodes, but the firmware and driver aggregate them to output a single touch event to the operating system. Since the aggregation algorithm is proprietary, the conversion from electrical signals of our interest to such touch events can only be empirically learnt.

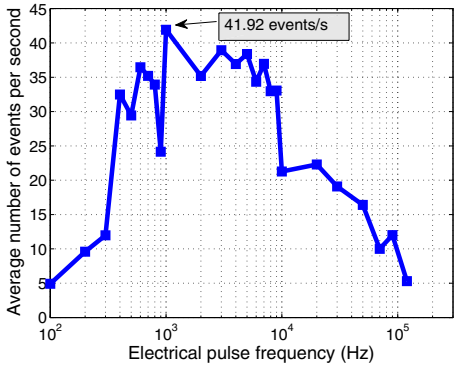


Figure 8: Touchsreen responses to 10 Volt peak-to-peak square wave signals with different frequencies ranging from 100Hz to 120KHz (log scale)

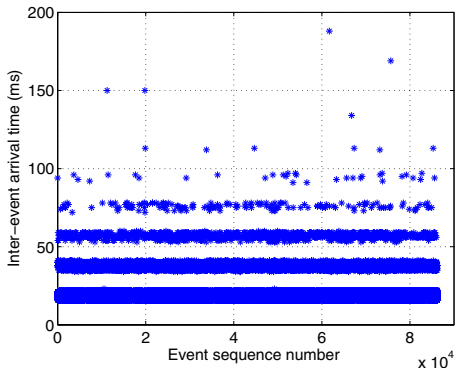


Figure 9: Distribution of inter-event arrival times of events generated by a 10 Volt peak-to-peak 1KHz square wave signal captured in kernel level log files

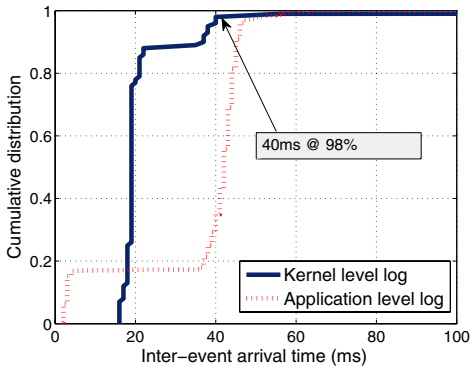
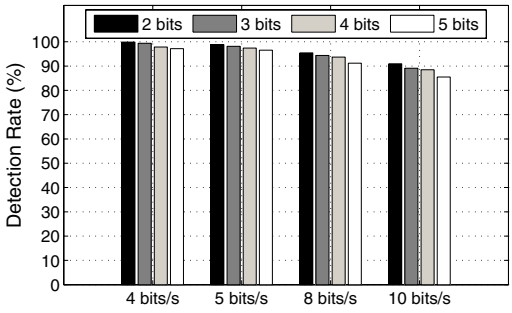
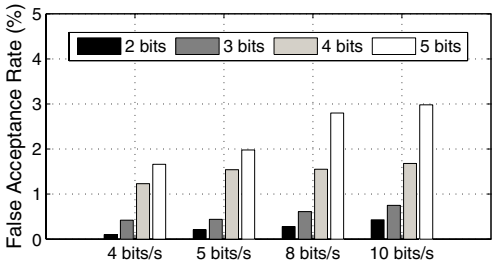


Figure 10: CDF of inter-event arrival times of events captured at application level and kernel level log files with 10 Volt peak-to-peak 1KHz square wave input signal

An important aspect of the system is the maximum possible data rate through the screen, which depends on two key characteristics of the screen: (i) the highest rate at which the driver and firmware allows touch events to be registered and (ii) the internal switching frequency of the sensing hardware. Atmel mXT1386's datasheet specifies a maximum of 150 *raw touch events* per second [7]. However, due to the driver in Android's software stack, the maximum rate is significantly reduced. We conducted many experiments to gauge the actual maximum event detection rate. We transmit signals with different waveforms, at different frequencies and voltage levels to a screen. With frequencies ranging from 100 Hz to 120 KHz, we observed that a 10 Volt peak-to-peak square wave signal at a frequency of 1 KHz can register the maximum rate of 41 events/second (i.e. average inter event arrival time of $\frac{1}{41} = 24$ ms). In particular, we began with finding the frequency that the touch-screen was most responsive. To do so, we set the Tektronix digital function generator to generate square wave of different frequencies at 10 Volt peak-to-peak amplitude. The frequency varies from 100 Hz to 1 KHz with 100 Hz difference, from 1 KHz to 10 KHz with 1KHz difference, and from 10 KHz to 120 KHz with 10s



(a) Detection rate for different message lengths and bit rates



(b) False acceptance rate for different key lengths and bit rates

Figure 11: Detection rate of 99.4% for 3-bit message transmitted at 4 bits/s

KHz difference. To collect the signals, we wired the output from the function generator to a flat solder electrode, then put the electrode on the surface of the Samsung Galaxy Tab 10.1 touchscreen. To make the electrode stable on the surface, we taped the it tightly to the touchscreen to avoid unintended movement. For each frequency, we collected the data for 200 seconds. Then record the number of event collected from the kernel. The average number of events is shown in Figure 8, which suggests that the screen best responses to signal at 1 KHz.

Figure 10 shows CDF of inter-event arrival times at kernel and application level log files. While almost 90% of the times, 2 consecutive events captured by the kernel log happen within 20 ms with very little variation, that number widely varies from 3 ms to 48 ms in the case of application level log. That observation indicates that using the timing information from kernel level log would could improve the demodulation results which mainly relies on event timestamps. In additional, we also observed that sinusoidal or triangular signals do not register any events. We further test with signals with different amplitudes and noticed that if the voltage amplitude of the signal is too high, the screen blocks all subsequent touch events for a short period of time and sends an error event to the operating system, which is the SUPPRESS event mentioned above. If the voltage is too low, the signal is either not detected or detected at a very low rate by the touch screen. We do not present the results of those variations in this paper due to the space limit.

A scatter plot of 86,200 events collected over 1850 seconds, Figure 9, illustrates the distribution of inter-arrival times, i.e. the time difference between two consecutive events, captured in kernel level log. An interesting pattern can be observed in Figure 11 is: most of the inter arrival times fall into specific narrow bands which we believe to be due to firmware throttling. Its cumulative distribution shows that 98% of the time, the inter-event arrival time is less

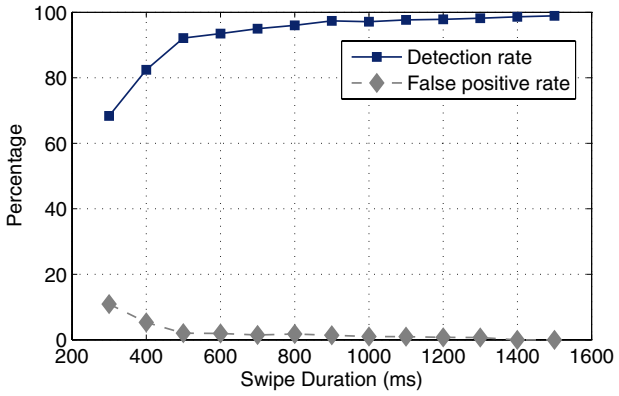


Figure 12: Multi-user games: Swipe detection rate

than 40 ms. Note that this event detection rate is more than 7 times lower than the rate of 150 raw touch events per second specified by the manufacturer [7]. Without access to the physical layer and the proprietary driver, we cannot determine the origin of this discrepancy. However, data rate could be at least 7 times faster than what we currently achieve if we have access to the driver; and an even higher data rate might be possible with direct access to the lower physical layer.

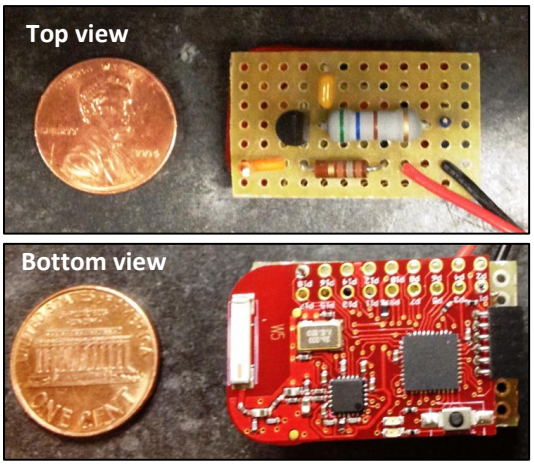
5.2 Bitrate vs. Detection Rate Trade-off

The main performance metrics here are the detection rate and the false acceptance rate. The detection rate signifies the probability of correct decoding of a message while the false acceptance rate characterizes the probability of a wrong message being incorrectly decoded as the original message. As explained in section 4 and shown in table 1, there exists a trade-off between the detection rate and the bit rate at which messages can be decoded from the touchscreen event logs. Correspondingly, since there are higher chances of incorrect decoding at higher bit rates, the number of false positives increase as the bit rate increases. In order to quantify this phenomenon, we use the setup described above to repeatedly transmit messages of different length at different bit rates. Figure 11 shows the detection and false acceptance rates observed. To derive the detection rate for each message length and bit rate, we transmit each message of that length 5000 times and present the average percentage of messages that are correctly decoded. Similarly, the false acceptance rate is derived by sequentially fixing each message as the correct message and transmitting all other messages of the same length 5000 times.

The trends in Figure 11 indicate a gradual decline in the detection rate with the increase in either the transmission bit rate or message length. We note that for simple parental control applications, a 99% detection rate can be achieved by using 2 or 3 bit messages at 4 bits/s. For applications that have a less stringent detection rate requirement, a much higher bit rate can be used to speed up the required data transmission time.

5.3 Indirect Communication Results

The next set of results are targeted towards detection of individual users in an indirect communication scenario. In this scenario, while the bit rate required is not very high, touching the ring to the touchscreen would hinder in the game-playing process. As such we leverage the fact that even if the finger-tip of the ring bearing finger touches the screen, the patterns in the registered event logs can be



(a) Prototype circuit board

(b) Usage of the prototype ring

Figure 13: The prototype ring and its usage for transmitting short messages from the ring to a touchpad

used to differentiate between a user with a SigRing and one without it.

In order to quantify the performance of this algorithm, we collected a total of 6,000 swipes from 3 different users with half the swipes with a ring on. We asked the users to vary the swipe duration between 300ms to 1.5 seconds but since making a swipe last for precisely a given time is hard, we bucket the collected swipes into 100ms durations starting from 250ms to 1550ms and discard swipes outside of this range. The swipe duration of all swipes within a bucket are approximated by the mean value of the bucket. Using the move events registered in this dataset, we calculated the detection rate of ring bearing users and the percentage of swipes without rings which were wrongly classified as one with rings, i.e., the false acceptance rate.

The resulting values shown in Figure 12 shows that the detection rate increases with the duration of the swipe, at first sharply from $\sim$68% for 300ms swipes to $\sim$92% for 500ms swipes and then gradually after that. Thus if the swipes used in a multi-player game is more than 700ms, the users can be classified correctly with a 95% confidence level.

6. RING PROTOTYPE AND EVALUATION

The use of this communication channel with touchscreen device requires a hardware token to generate the appropriate electrical signal and inject it into the touch sensing circuitry. In this section, we describe our prototype of the ring which uses off-the-shelf components.

6.1 Hardware Prototype

The core of the token is the low cost, low power microprocessor, TI-MSP430F2722 [35] that was programmed to generate modulated 3 Volt square waves at a frequency of 1KHz. Figure 14 shows the schematic of the custom-built ring. This square wave is modulated with On-Off keying to trigger artificial touch events in the screen's firmware. The microprocessor is mounted on a 18 mm x 30 mm off-the-shelf board, part of TI-MSP430 eZ430 development kit, as shown on Figure 13(a)-bottom view. We specify the transmission data rate and data sequence by programming the microprocessor through the USB interface that comes with the kit. The square wave and its parameters were selected through experiments with a function generator, as described in section 5. Since

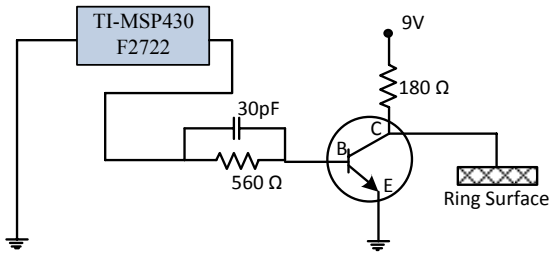
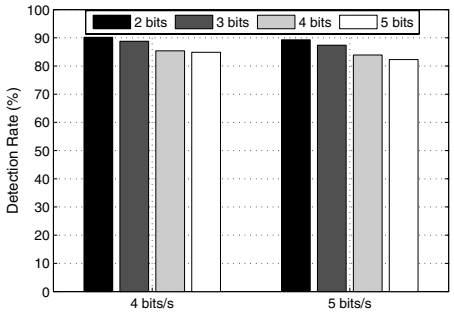


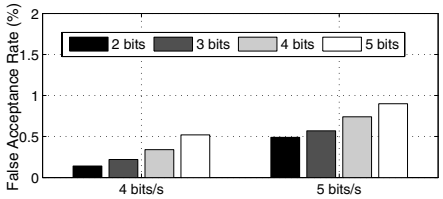
Figure 14: Schematic of the custom-built ring

we found that 3 Volt was not adequate for generating touch events, we amplify the output of the microprocessor using a single bipolar transistor, BC548B [5], with the supply voltage of 9 Volt (Figure 13(a)-top view). One of the most challenging parts of the prototype was to design the electrode configuration that would allow the signal to be injected in series with the touchscreen and the body capacitance of the user. The best point in the circuit to inject the signal, V'_{sig} in figure 1, would be in series with the finger and the rest of the body at a point close to the screen. This has obvious anatomical difficulties and the low internal resistance of the body makes injection between two closely spaced electrodes, as on the inside surface of a ring, impractical. We opted for a system where the user would wear an insulating ring where V'_{sig} was injected between electrodes on the inside and outside of the dielectric band. The inner electrode was connected with the finger and, through the body capacitance C_B and case capacitance C_c (as described in section 2), to the internal circuitry in the tablet. The outer electrode on the ring was directly pressed on the screen, forming C_s to complete the circuit.

Because a uniform and reproducible contact between the touchscreen and the ring is essential to minimize the error rate, we choose to use a flexible conductive material to make the electrode and design the face of the ring to control the compression of that material. If the pressure is too high, the screen bends and its capacitance, C_s, increases which in turns can introduce errors. We control this pressure by surrounding the electrode with an insulating spacer of the



(a) Detection rate



(b) False acceptance rate

Figure 15: Detection rate and False acceptance rate using ring prototype for different message lengths and bit rates

correct thickness to properly control the compression of the flexible electrode.

6.2 Preliminary Prototype Performance

Using the prototype ring, we experimented with injecting messages through the Samsung Galaxy Tablet 10.1 touchscreen. While conducting the experiment, we noticed that at times no events were triggered during the transmission of a 1 bit or vice versa. This lead to unreliable decoding of messages but we were still able to distinguish two codes with a larger hamming distance. We used one ring with the code '1110' and one ring with the code '1000' and touched each of the 50 times on the tablet display. A simple threshold algorithm that uses as input the number of touchscreen events generated was able to identify the first ring correctly 44 times and the second ring 43 times, leading to an overall detection rate of 87%. We suspect that the quality of the contact between the ring and the touch panel plays a critical role in these experiments.

To eliminate this variance due to contact differences from touch to touch, we experimented with transmitting multiple messages while the ring was held steady on the display. Here, we used message lengths between 2 and 5 bits transmitted at the rates of 4 bits/s and 5 bits/s from which the detection rate (DR) and false acceptance rate (FAR) are evaluated. For each message at each data rate, we put the ring down onto the screen 3 times and keep it there long enough so that 200 repetitions of the message are transmitted from the ring to the screen. We show in Figure 15 the DR and FAR results over the 200 repetitions from best case (presumably best contact) of these three trials. Each bar represents the average rates over different data rates and message lengths. We observed that the detection rate decreases with the increase of both the message length and bit rate. Note, however, that the overall detection rate could be improved through retransmissions of the message. Therefore, even the lower detection rate of 82% may still be adequate for some of our targeted applications. For the user identification application, for example, up to 3 seconds of continuous repeated message transmission would results in less than 6 errors per 1000 uses. These results illustrate what can be achieved with this transmitter if the reliability issues are worked out.

We believe that another source of error in this prototype stems from the relatively long rise time of the square wave since the touchscreen events appear to be triggered by the edges in the input signal. It is also important to note that both the electronics and the firmware of the screen, which we do not have access to, are optimized for the relatively slow movement of a human finger. Thus, the screen driver deliberately throttles the maximum rate of touch events, which reduces touch error in normal use but limits our system to very low bit rate transmission. We believe that the transmission rate could be improved substantially with access to the touchscreen controller firmware, which should allow processing internal touchscreen measurements, e.g. physical voltage differences.

7. DISCUSSION

Let us briefly consider several remaining issues related to the energy consumption and security applications of this technique.

Energy Consumption. The current prototype implementation is based on a 3 Volt microprocessor driving a 9 Volt high speed bipolar transistor amplifier to generate a continuous signal. Energy consumption and some of the synchronization issues in our prototype could be significantly reduced by incorporating a switch under the contact surface that powers up the ring when pressed against the touch screen. To estimate the cost and battery life of such a ring version, we use the smallest readily available lithium primary battery, the CR2025 which is 10 mm diameter and supplies 3.0 volts with a 30 mA-h capacity. The typical current drain in standby with RAM-retention of a modern microprocessor (e.g. the TI MSP430 family) is about 0.1 microamps. Even with this small battery, this would provide over 3 decades of standby lifetime for the ring electronics. Once awake, the processor will use significantly more current, but the minimal computing requirements result in this being low, also. The smaller MSP-430 processors typically use about 220 microamp at 1 MHz, so even if shifting out the short code takes 100 cycles of the CPU, this battery will still provide enough energy for over 5000 uses.

Since the capacitances are very small, the current also will be low and a simple buck-boost dc-dc converter with one miniature inductor will be quite adequate to supply the 9 Volt[6]. Assuming only a 10 % charge conversion efficiency for the converter, this circuit still uses only about 2 nanocouloumbs/charge-discharge cycle. Modulating at 1 KHz and sending 10 bits/second, this allows the battery to supply over 50 million bits, far in excess of any of the other limits in the system. The cost of such a system will be dominated by the processor, several tens of cents, but in high volume that can be replaces by a simple sequence generator, either read-only or flash, for only a few cents.

Security considerations. The current limits on datarate only allow transmission of very short codes and thus allow only weak authentication at best. Improvements in datarate through modifications in the touchscreen firmware could alleviate these limits, however. The low carrier frequency of our system, between 5-10 kHz, would then also offers additional protection against eavesdropping. Since antenna size should be proportional to the wavelength of the signal, transmission of this signal into the RF domain would require an antenna much larger than the size of the human body. While we cannot rule out that some signal can be received with customized resonant antennas, however, the level of effort required would be much higher than for picking up a e.g. 2.4 GHz signal used in WiFi and Bluetooth. If such eavesdropping ever were an issue, it could

also be addressed by transmitting a noise signal from the receiving device.

Finally, the design could be enhance with a feedback channel using a photodetector. The ring could receive information from the mobile device through this visual channel, where the device encodes the information in the pixel intensities. This would enable a challenge response protocol, which could greatly enhance the security of an authentication system.

8. CONCLUSION

We have presented the design and implementation of a technique to transmit information through a capacitive touchscreen. Our method triggers touch events in the touchscreen device by injecting an electric signal that affects the capacitance measurements of the screen. Our experiments show that this is feasible even with an off-the-shelf touchscreen system, albeit at very low bitrates. Controlled experiments with a signal generator demonstrates data rates of 5-10 bps. While some reliability challenges remains, we also achieved up to 4-5 bps with a wearable transmitter token in the form of a small signet ring and demonstrated that some signals can be transmitted through the human skin. Transmission of information via small physical tokens can be used to distinguish who is interacting with a mobile device, and can be useful for parental control, multiuser games (particularly when played on a single device), and possibly play a role in authentication solutions. It differs from other short-range communication systems in that it requires physical touch for communication, which can be an advantage if multiple potential users are so close that they cannot be differentiated with the other short-range systems. The technique could also be used to distinguish different devices touching the screen such as styluses or boardgame tokens. We believe that significantly higher data rates could be achieved by designing receiver capabilities into touch screens and few this work as a first step towards exploring how this touch sensor can participate in the exchange of information between mobile devices.

Acknowledgement

We are thankful to Nilanjan Paul, Joshua Devasagayaraj, and Ivan Seskar for their help with developing prototypes of the signet ring. This material is based in part upon work supported by the National Science Foundation under Grant Nos. 0845896 and 1040735. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

9. REFERENCES

[1] Face recognition on galaxy nexus is very neat, but easily compromised with the picture? http://tiny.cc/gj50gw, 2011.
[2] Nearfield communications forum. http://www.nfc-forum.org/, 2011.
[3] A. Adler. Vulnerabilities in biometric encryption systems. In *Proc. of AVBPA*, 2005.
[4] Afg3000 series arbitrary function generators. http://www.tek.com/AFG3000.
[5] Amplifier transistors bc548b. http://onsemi.com.
[6] An mcu-based buck-boost converter for battery chargers. http://www.st.com/.
[7] Atmel maxtouch mxt1386 specifications. http://www.atmel.com/devices/MXT1386.aspx.
[8] G. Bailador, C. Sanchez-Avila, J. Guerra-Casanova, and A. de Santos Sierra. Analysis of pattern recognition techniques for in-air signature biometrics. *Pattern Recognition*, 44(10-11):2468 – 2478, 2011.
[9] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proc. of NDSS*, Feb. 2002.
[10] Belkin. Wemo home automation system.
[11] N. Ben-Asher, N. Kirschnick, H. Sieger, J. Meyer, A. Ben-Oved, and S. Möller. On the need for different security methods on mobile phones. In *Proc. of MobileHCI*, Aug. 2011.
[12] H. Bojinov and D. Boneh. Mobile token-based authentication on a budget. In *Proc. of HotMobile*, Feb. 2011.
[13] Craftsman. Assurelink garage doors openers.
[14] P. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *Proc. of UIST*, Nov. 2001.
[15] Digital Modulation Schemes. http://robotics.eecs.berkeley.edu/ sastry/ee20/modulation/node5.html.
[16] P. Dunphy, A. P. Heiner, and N. Asokan. A closer look at recognition-based graphical passwords on mobile devices. In *Proc. of SOUPS*, July 2010.
[17] M. Faundez-Zanuy. On the vulnerability of biometric security systems. *Aerospace and Electronic Systems Magazine, IEEE*, 19(6):3 – 8, June 2004.
[18] C. Foresman. Apple facing class-action lawsuit over kids' in-app purchases. *Ars Technica*, Apr. 2011.
[19] Fundamentals of electrostatic discharge. http://www.esda.org/fundamentalsp5.html, 2010.
[20] J. Galbally-Herrero, J. Fierrez-Aguilar, J. D. Rodriguez-Gonzalez, F. Alonso-Fernandez, J. Ortega-Garcia, and M. Tapiador. On the vulnerability of fingerprint verification systems to fake fingerprints attacks. In *Proc. of Carnahan Conferences on Security Technology*, Oct. 2006.
[21] C. Gehrmann, C. J. Mitchell, and K. Nyberg. Manual authentication for wireless devices. *CryptoBytes*, Spring 2004.
[22] E. Haselsteiner and K. Breitfuss. Security in Near Field Communication (NFC). In *Proc. of the Workshop on RFID Security*, 2006.
[23] K. Hawkey and K. M. Inkpen. Examining the content and privacy of web browsing incidental information. In *Proc. of WWW*, May 2006.
[24] A. K. Jain, K. Nandakumar, and A. Nagar. Biometric template security. *EURASIP J. Adv. Signal Process*, 2008:113:1–113:17, Jan. 2008.
[25] E. A. Johnson. Touch Displays: A Programmed Man-Machine Interface. *Ergonomics*, 10(2):271–277, 1967.
[26] A. K. Karlson, A. B. Brush, and S. Schechter. Can i borrow your phone?: understanding concerns when sharing mobile phones. In *Proc. of CHI*, Apr. 2009.
[27] S. Kurkovsky, T. Carpenter, and C. MacDonald. Experiments with simple iris recognition for mobile phones. In *Proc. of Information Technology: New Generations*, pages 1293 –1294, april 2010.
[28] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. User evaluation of lightweight user authentication with a single tri-axis accelerometer. In *Proc. of MobileHCI*, Sept. 2009.
[29] S. Mathur, R. Miller, A. Varshavsky, W. Trappe, and N. Mandayam. Proximate: proximity-based secure pairing using ambient wireless signals. In *Proc. of MobiSys*, June 2011.
[30] T. Matsumoto, H. Matsumoto, K. Yamada, and S. Hoshino. Impact of Artificial "Gummy" Fingers on Fingerprint Systems. In *Proc. of SPIE Vol. 4677*, 2002.
[31] R. Mayrhofer and H. Gellersen. Shake well before use: Intuitive and secure pairing of mobile devices. *IEEE TMC*, 8(6):792 –806, Jun. 2009.
[32] M. L. Mazurek, J. P. Arsenault, J. Bresee, N. Gupta, I. Ion, C. Johns, D. Lee, Y. Liang, J. Olsen, B. Salmon, R. Shay, K. Vaniea, L. Bauer, L. F. Cranor, G. R. Ganger, and M. K. Reiter. Access control for home data sharing: Attitudes, needs and practices. In *Proc. of CHI*, May 2010.
[33] Mobile biometry: European funded project (fp7-2007-ict-1). http://www.mobioproject.org/, 2011.
[34] Motorola atrix: Technical specifications. http://bit.ly/f7nfEX, 2011.
[35] Msp430 ultra-low power 16-bit microcontrollers. http://msp430.com.
[36] A. Nicholson, M. D. Corner, and B. D. Noble. Mobile Device Security using Transient Authentication. *IEEE TMC*, 5(11):1489–1502, November 2006.
[37] Ringbow. http://ringbow.com/ringbow/.
[38] V. Roth, P. Schmidt, and B. Güldenring. The IR ring: authenticating users' touches on a multi-touch display. In *Proc. of UIST*, Oct. 2010.
[39] M. Sato, I. Poupyrev, and C. Harrison. Touche: enhancing touch interaction on humans, screens, liquids, and everyday objects. In *Proc. of CHI*, pages 483–492, New York, NY, USA, 2012.
[40] Y. Shaked and A. Wool. Cracking the bluetooth pin. In *Proc. of MobiSys*, pages 39–50, 2005.
[41] E. Sofge. Can a smartphone make your car smarter? *MSN Autos*, 2010.
[42] Sony - mofiria finger-vien pattern recognition. http://bit.ly/cIHGuj, 2009.
[43] F. Stajano and R. J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proc. of Security Protocols*, Apr. 2000.
[44] Technology review article: Pushing the limits of the touch screen, 2011.
[45] L. Thalheim and J. Krissler. Body check: Biometric access protection devices and their programs put to the test. *ct Magazine*, Nov. 2002.
[46] M. Weir, S. Aggarwal, M. Collins, and H. Stern. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proc. of CCS*, Oct. 2010.
[47] W. Westerman and J. G. Elias. US 0232567 Patent - Capacitive Sensing Arrangment, 2006.