Pervasive and Mobile Computing 90 (2023) 101756

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

MousePath: Lightweight phone-to-web information sharing via mouse interface

Yihui Yan [a,1,2,3], Zhiwei Wang [a,1], Qianyi Huang [b], Zhice Yang [a,*]

[a] School of Information Science and Technology, ShanghaiTech University, China
[b] School of Computer Science and Engineering, Sun Yat-sen University, China

ARTICLE INFO

Article history:
Received 29 August 2022
Received in revised form 18 January 2023
Accepted 1 February 2023
Available online 4 February 2023

Keywords:
Cross-device information sharing
Optical mouse
Visible light communication
Bluetooth

ABSTRACT

This paper proposes MousePath, a novel lightweight communication system between PC web pages and smartphones. MousePath works in two modalities, MousePath-OPT and MousePath-BL. MousePath-OPT works by putting the optical mouse on top of the smartphone's screen, then its transmission starts and instantly finishes without association and pairing fraction. It encodes data into the movement of the smartphone's display content and leverages the optical mouse of the computer to sense the movement for decoding the data. MousePath-BL works by emulating the smartphone as a Bluetooth wireless mouse. Then the smartphone can directly transmit information to the web page via generating mouse events. We prototype and evaluate the system with commercial computers and smartphones. A key benefit of MousePath is that it seamlessly bridges smartphones to co-located PC web applications. MousePath-OPT is more secure and convenient to use while MousePath-BL achieves higher throughput. Two representative web applications, *i.e.*, sensor sharing and message sharing, are developed to demonstrate MousePath's potential in enhancing PC web applications.

1. Introduction

The Web is progressively evolving. Traditional web pages primarily serve as an output interface for users to retrieve information from a remote server, while recent web pages also try to retrieve client-side information to enrich its functionalities. For instance, Media Capture and Streams API[4] enables web VoIP through accessing microphones and cameras.

Motivated by the popularity of smartphones, this paper considers a question following the above trend: can PC web pages, *i.e.*, web pages accessed via a desktop or a laptop, be further enhanced by taking information from co-located smartphones? A positive answer would lead to several interesting and beneficial web applications. For example, it will allow the PC web page to retrieve the account and password credentials stored in the smartphone for the auto-filling in of the login session. Another example is sensor sharing. Smartphones are rich in various sensing capabilities. Unique ones can be made use of by PC web pages. Below are a few examples. Accessing the localization sensors of the smartphone

* Corresponding author.
E-mail address: yangzhc@shanghaitech.edu.cn (Z. Yang).

1 Co-primary Authors.

2 Also with Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Science.

3 Also with University of Chinese Academy of Sciences.

4 https://www.w3.org/TR/mediacapture-streams/

https://doi.org/10.1016/j.pmcj.2023.101756
1574-1192/

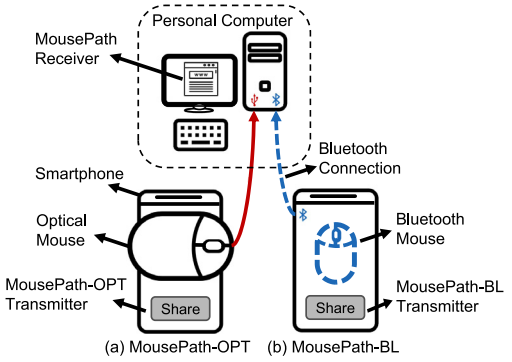


Fig. 1. Illustration of MousePath Usage. MousePath consists of two entities: the transmitter app on the smartphone and the receiver application on the web page. To transfer data from the smartphone to the web server, the user simply opens the app and either (a) puts the optical mouse on the screen of the smartphone, or (b) connects the smartphone to the computer through Bluetooth (if any). The data can then be decoded by the receiver scripts on the web page.

enables precise location-based services (LBS) on PC web pages. Through light sensors, the online photo editor can auto-tune its color space to fit the ambient light condition. Further, accessing the heart rate sensor is helpful for remote health diagnosis.

When putting the above idea into practice, a data connection between the smartphone and the PC web page must be established. However, this is surprisingly challenging. There are already mature solutions for sharing information between smartphones and PCs, such as the Your Phone App[5] in Windows 10 and the Universal Clipboard in Apple products[6] . They all target *phone-to-PC* sharing rather than *phone-to-web* sharing. In situations where the web page already provides suitable input interfaces, such as text box and image drop area, the two concepts are close. However, when the web page requires verbose input such as location altitude and raw sensor samples, there is a gap between the two concepts. The *phone-to-PC* sharing brings only information to the PC's operating system, thus additional effort, *i.e.*, installing a browser plugin, is needed to further convey the information to the web page. This, however, cancels out the major benefits of using a web application — the simplicity of no installation or management efforts being required.

In this paper, we propose MousePath, a lightweight and convenient way to realize *phone-to-web* sharing. Its key idea is based on the fact that web applications can directly take input from mouse events, such as cursor shifts. Meanwhile, MousePath incorporates techniques to allow the smartphone to seamlessly take over the mouse to generate certain events for conveying information. This renders a convenient *phone-to-web* sharing approach. Specifically, it works in two modalities.

As shown in Fig. 1(a), MousePath-OPT [1] works by putting an optical mouse on top of a smartphone screen, and then the information from the smartphone is directly transferred to the web application in the PC. MousePath-OPT transfers data through a novel channel lying between the smartphone screen and the PC's optical mouse. The key insight is to make use of the motion sensing ability of optical mice. We observe that, when putting the optical mouse on the screen of the smartphone, it can sense the movements of the display content. Based on this property, the MousePath-OPT transmitter in the smartphone encodes the data into the movements of the display content, which fools the optical mouse into treating the content movement as real physical movement. As a result, the MousePath receiver, *e.g.*, a piece of JavaScript in the web page, can infer the movements of the display content from the system's mouse trajectories, through which the data can be decoded.

MousePath-BL (Fig. 1(b)) is similar to MousePath-OPT in that they both make use of the mouse movements to convey information, but it works by directly emulating the smartphone as a Bluetooth mouse. The feasibility originated from the Bluetooth stack, which uses profile-based schemes to simplify the connection management of various peripheral devices. It allows the device to claim which profile to use. Based on this feature, we force the smartphone to choose the mouse profile for the Bluetooth connection to fool the host system. Then, the MousePath-BL transmitter app can report mouse movement events to the PC. Similarly, the data is encoded into the movements, allowing the MousePath receiver within the web page to directly decode it.

Although we have not noticed any other *phone-to-web* sharing systems, MousePath is not the only way to achieve this. For example, by using the latest Media Capture and Streams API and compatible browsers, web pages can access the camera to capture a QR code stream on the smartphone to achieve similar functionalities. MousePath is superior to the camera-based method in API compatibility, hardware availability, and visual privacy.

5 https://www.microsoft.com/store/productId/9NMPJ99VJBWV

6 https://support.apple.com/en-us/HT209460

Our contributions are:

- We propose a novel lightweight screen-to-mouse communication channel by leveraging screens for transmitting and optical mice for receiving. To our knowledge, there are no similar methods which utilize the original sensing ability of optical mice for communication.
- We propose the MousePath-OPT system based on the screen-to-mouse communication channel, which can serve as a ubiquitous and convenient data path between smartphone and PC web applications.
- We propose the MousePath-BL system to complement MousePath-OPT. MousePath-BL emulates the smartphone as a Bluetooth mouse by taking advantage of the Bluetooth profile. MousePath-BL is able to achieve higher throughput when a Bluetooth interface is available.
- We implement a prototype system of MousePath and evaluate it with various commodity optical mice and smartphones. The receiver is written in Javascript and can be embedded into web pages and directly run with almost any web browser.
- We demonstrate MousePath with two web applications. One enables precise location-based services for PC web pages and the other shows the feasibility of integrating with the phone's password manager.

2. Related work

MousePath is a special communication system based on visible light and Bluetooth. Thus, we compare it with related screen-to-camera channels and wireless channels. Besides, due to the importance of desktop computers in office tasks, the literature explored many methods to enhance their interaction. While we mainly focus on the communication issue, enhancing desktops with smartphones is relevant to our developed applications. For the same reason, we also discuss augmented mouse.

Screen-to-camera Channel. Since the optical mouse sensor is actually an image sensor, MousePath is related to the area of screen-to-camera communication. At a high level, the approaches encode information in the display content of the screen, and use a camera to capture the screen and decode the information. The basic example is scanning QR-codes. Recent efforts in screen-to-camera research improve the performance in various aspects, including the data rate [2,3], computation overhead [4], visibility [5,6], *etc.* Unlike the aforementioned, the screen-to-mouse channel leveraged in this paper has not been investigated before. Specifically, since the optical mouse by default only provides movement information, MousePath decodes messages from the mouse's moving trajectories instead of from images. Further, MousePath-OPT encodes messages by slightly shifting the same display pattern in consecutive display frames instead of modifying the display content.

Temporary Wireless Channel. Wi-Fi Direct and Bluetooth are radio technologies designed for ad-hoc wireless connections, but, in practice, the association process still costs considerable time (up to several seconds [7]), which cannot be ignored when performing one-off interactions. Other communication forms are also studied under the context of cross-device information sharing. Acoustic communication [8,9] and visible light communication [2,3,10] are two popular forms in a mobile situation, but people have a skeptical attitude toward exposing microphones and cameras to web applications, since they bring the risk of privacy leakage. Further, some sensors, such as ambient light sensors that are available on mobile devices [10], are not available on PCs, hence cannot be used. Moreover, unlike MousePath, the schemes based on them do not target the *phone-to-web* situation, and additional efforts are required to communicate to the web application.

Enhancing Desktop Interactions with Smartphones. There is a long history of researchers combining smartphones and desktops. The Pebbles project [11] makes use of a touch-screen personal digital assistant (PDA) to display the user interface of desktop applications, extending the desktop's display and control interface. Similar approaches are discussed under the context of data sharing [12], I/O sharing [13], math equation editing [14], and so on. Moreover, smartphones are explored as a trusted computing device to improve the security of desktop computers [15,16]. All of these works assume there is a network connection between the smartphone and the desktop, but ignore the practical overheads in establishing and maintaining the connection. MousePath provides a ubiquitous and convenient way to transfer data from a smartphone to a desktop.

Augmented Mouse. Several mouse prototypes are invented with additional functions to enrich both the input and output capabilities. Some of them are designed to capture additional dimensions of user actions, such as rotations [17] and holding pressure [18,19]. A body of work explores the use of mice as an output device, for example, to provide visual [20], haptic [21] and force [22] feedback. Mice in LivingDesktop [23] can automatically move to improve the ergonomics. Mice in these works are research concepts and prototypes, instead, MousePath-OPT is based on commercial optical mice and uses them for communication. MousePath-OPT enriches desktop interactions by combining mice with smartphones. At a high level, MagicDesk [24] uses a similar way to organize the screen and the optical mouse. Their mouse is put on a digital touchable screen. The underlying screen shows additional UI elements associated with the mouse, such as additional menu items. Users can choose to directly press the touch screen instead of moving the mouse to click the button, in this way to facilitate efficiency. But unlike MousePath-OPT, their screen does not transfer data to the mouse.

Near-Field Communication. Recent work has developed near-field communication by using various sensors. MagneComm [25] utilizes the Magnetic Induction (MI) signals from the CPU and the magnetometer to build a one-way channel from the laptops to mobile phones, which is in the opposite direction from ours. Ripple [26,27] develops a

Table 1
Summary of related work.

Name	Tx Interface	Rx Interface	Privacy Concern	Connection Setup	Rx Hardware Availability on PC		Usability	Throughput
					Laptop	Desktop		
PixNet [2]	Display	Camera	**Visual**	No	Yes	**Rare**	Phone-to-Web[1]	12 Mbps
Lightsync [3]	Display	Camera	**Visual**	No	Yes	**Rare**	Phone-to-Web[1]	11 Kbps
PIXEL [4]	Customized Light	Camera	**Visual**	No	Yes	**Rare**	Device-to-Phone	14 bps
ChromaCode [5]	Display	Camera	**Visual**	No	Yes	**Rare**	Phone-to-Web[1]	777 Kbps
TextureCode [6]	Display	Camera	**Visual**	No	Yes	**Rare**	Phone-to-Web[1]	22 Kbps
Dhwani [8]	Speaker	Microphone	**Audio**	No	Yes	**Rare**	Phone-to-Web[1]	2.4 Kbps
LightTouch [10]	Display and Bluetooth module	Ambient Light Sensor and Bluetooth module	-	**Yes**	**Rare**	**No**	Phone-to-PC	91-172 Kbps
Split-trust Browsing [15]	Network Interface Card		-	**Yes**	Yes	Yes	Phone-to-Web[2]	-
MagneComm [25]	CPU	Magnetometer	-	No	**No**	**No**	PC-to-Phone	110 bps
Ripple [26]	Vibration Motor	Accelerometer	-	No	**No**	**No**	Phone-to-Phone	20-80 bps
Ripple II [27]	Vibration Motor	Microphone	**Audio**	No	Yes	**Rare**	Phone-to-Web[1]	30 Kbps
Vinteraction [28]	Vibration Motor	Accelerometer	-	No	**No**	**No**	Phone-to-Phone	5-20 bps
MousePath-OPT	Display	Optical Mouse	-	No	Yes	Yes	Phone-to-Web	120 bps
MousePath-BL	Bluetooth Module	Bluetooth Module	-	**Yes**	Yes	**Rare**	Phone-to-Web	600 bps

[1] Browsers need to support specific Web APIs.
[2] Customized browser.

system to communicate small packets through physical vibration, which involves vibration motors, accelerometers and microphones. Yonezawa et al. [28] present Vinteraction to transmit data from smartphones to smart tablets through vibration. Dhwani [8] proposes an acoustics-based NFC system to peer-to-peer transmit data via speakers and microphones on mobile phones. Since smartphones are rich in sensors, they can natively receive various near-field signals without additional devices. However, most PCs have to use additional hardware components to receive. Although microphones are available in some computers, directly directing audio signals to web applications causes privacy concerns. In contrast, MousePath is privacy-preserving and compatible with general PCs without requiring additional hardware.

Table 1 compares MousePath with the above work in terms of hardware availability, privacy leakage, usability, and throughput. Most of the related work do not target the *phone-to-web* sharing problem. While some of them can be further improved to achieve the same goal, they have issues of privacy leakage, hardware availability, and connection setup overhead.

3. Motivation and technical background

This section describes the motivation and background knowledge of MousePath.

3.1. Motivation

It is a very common demand to share information on the smartphone to the web page opened on a co-located PC. However, it remains a cumbersome practice in many situations. An example is when the user tries to log in to web accounts on public computers, *e.g.*, desktops in the Internet lounge, school library, *etc.* He/she still has no convenient way to input the account and password information. One solution is to log into the browser to sync everything across computers, but it raises security and privacy concerns. Another choice is through smartphone password manager apps, but the companion software has to be first installed on the public computer to receive messages from the smartphone app.

We note that the above is not an issue of password managers or general network capabilities. Fundamentally, it is a usability issue as we lack convenient ways to transfer messages directly from smartphones to web applications. We consider it as the *phone-to-web* sharing problem. Apart from transferring web credentials, enabling convenient and ubiquitous *phone-to-web* sharing can facilitate many web applications. For example, having the smartphone's location information allows for auto-filling of shipping addresses on web pages. Through sharing the smartphone's light sensor data, online photo editors can auto-tune its color space to fit the local ambient light condition. Further, the smartphone can act as an agent for PCs to access sensors of other smart devices, *e.g.*, accessing the heart rate sensor of smartwatches is helpful for web-based remote health diagnosis.

Achieving *phone-to-web* sharing has no ready solutions so far. One reason is that the web browser is designed as a sandbox to isolate the web content from the host system for security reasons. As a result, web applications lack convenient mechanisms to access I/O interfaces of the host system, which largely limits their potential in the above application scenarios.

Recent web standards are pushing new features to meet the above demand. With the user's permission, a web script is able to access the video, audio, and Bluetooth interface[7] of the host system. We note that these features are much more

[7] https://www.w3.org/community/web-bluetooth/

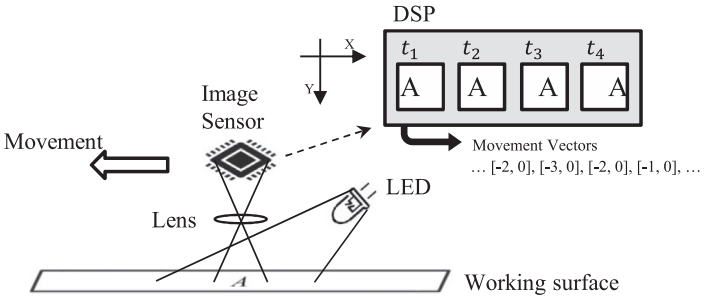


Fig. 2. Optical Mouse Uses Image Sensor to Detect Movement. For example, when the mouse is moved leftwards, the mouse DSP detects the movement according to the rightward movement of the texture "A" in the captured images. The mouse reports its observation via movement vectors showing negative X values, *i.e.*, the leftward direction in the mouse's local coordinate system .

convenient than browser plugins providing similar functionalities since they are natively supported by the browser and do not require user's installation and management actions. However, our investigations show that the new features bring back security and privacy concerns, and have capability issues, *i.e.*, not supported by every browser and browser versions.

In this paper, we seek an alternative approach to fulfill such a demand. We go back to investigate the most basic and in-common input interfaces supported by web pages to see if there are any opportunities to convey information through them. Our focus is attracted by the mouse interface. If the information can be carried in mouse trajectories, any foreground programs can access and decode it. In the following subsections, we look inside the mouse interface.

3.2. Optical mouse

An optical mouse is a kind of optical imaging system (Fig. 2). It uses a backlight (LED or laser) to illuminate the working surface. The image sensor, which is actually a low resolution but high frame rate CMOS sensor, continuously takes images of the working surface at a rate of thousands of frames per second. When the mouse is moved, the images change as the textures of the captured working surface also change (see the anchor character "A" in Fig. 2 for example). The sampling rate of the image sensor is so high that sequential images tend to partially overlap. Thus, the Digital Signal Processing (DSP) unit of the optical mouse can make use of the overlapped images to compute the direction and distance of the movement [29].

The DSP quantifies the movements to *movement vectors* $\vec{M}_i = [M_i^X, M_i^Y]$, which represent the accumulated displacement along the X-axis and Y-axis of a mouse's local coordinate system from time t_{i-1} to time t_i. Note that the integration of all $\vec{M}_i$ gives the trajectory of the movements, thus $\vec{M}_i$ are reported to the host system for GUI control. Although, in principle, the DSP can generate one $\vec{M}_i$ for every two sequential images, the default reporting/polling rate of $\vec{M}_i$ is fixed at 125 Hz in normal optical mice.

3.2.1. Idea of screen-to-mouse channel

The mechanism of the optical mouse imaging system leads to the following observation: *besides the reflected backlight, the image sensor of the optical mouse can also capture other light signals*. Specifically, when the optical mouse is put on a screen, it is possible to capture the display content.

Motivated by the above observation, we also find that the optical mouse reports movements if the underlying display content moves [1]. The observation and validation imply an untapped opportunity in transmitting data from the smartphone to the computer by fooling the optical mice with the movement of the smartphone's display content. Specifically, the data can be encoded in the shift directions of the texture. At the same time, the optical mouse can identify the shift directions, and report to its host system with movement vectors. The web applications in the host system can then decode the data encoded by the smartphone through analyzing the movement vectors. The above is just the idea of MousePath-OPT. We will elaborate it in Section 4.

3.3. Bluetooth wireless mouse

Bluetooth is a wireless communication technology widely used for short-distance data exchange. In order to simplify the connection management of heterogeneous peripheral devices, the Bluetooth standard uses the profile-based scheme. The profiles specify how the Bluetooth stack interprets packets, and where to direct their content in the host system to achieve their functions.

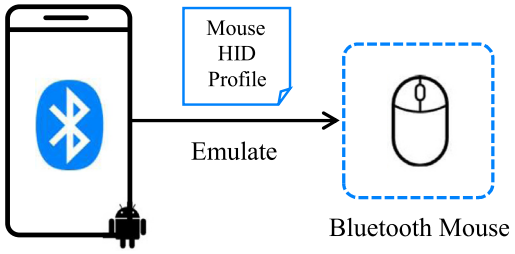


Fig. 3. Turning Smartphone into Bluetooth Mouse. By choosing the Bluetooth Mouse HID profile in the smartphone operating system, the smartphone can be recognized as a Bluetooth mouse. The Bluetooth packets between the smartphone and host will be decoded and forwarded to the mouse stack of the host system. The smartphone can thus generate arbitrary mouse events.

Bluetooth mouse Human Interface Device (HID) is one of the Bluetooth profiles.[8] It tunnels Bluetooth packets to the USB HID protocol stack, where they are further trapped by the system's mouse event handler like the USB mouse case. The profile specifies the protocol that a Bluetooth mouse device should follow when reporting mouse events via Bluetooth connection. Once a profile is chosen, the allowed packet format and the data path in the host system are determined. For the sake of consistency, Bluetooth mouse events mimic USB mouse events. Specifically, movement event is also represented by movement vector $\vec{M}$.

3.3.1. Idea of emulating a Bluetooth Mouse

Our target is to break the communication boundary between smartphones and web applications. Intuitively, if the smartphone is able to emulate a Bluetooth mouse and flexibly generate mouse events through software, then the movement vectors can be manipulated more efficiently to convey information.

The above idea becomes possible due to a new feature in Android. In order to meet the demand of controlling smart devices, *e.g.*, smart TVs, with smartphones, Android allows apps to select the Bluetooth HID profiles since version 9.[9] When the smartphone selects the mouse profile, it could emulate a Bluetooth mouse. As shown in Fig. 3, the HID profile of the phone announces that its Bluetooth connection follows the mouse profile, thus the host interprets the Bluetooth packets from the phone as mouse events.

MousePath-BL system takes advantage of this feature to establish a direct channel between smartphones and web applications. Unlike an ordinary Bluetooth mouse, the mouse event of MousePath-BL is not generated for reflecting physical movement and control but for communication purposes. It is described in Section 5.

4. MousePath-OPT

This section extends the idea in Section 3.2 to realize the screen-to-mouse channel. This special communication channel, which has not been studied before, contains unique challenges rooted from both the smartphone's display system and the optical mouse's imaging system. We address them with the transmitter and receiver design.

The *MousePath-OPT transmitter* is an app running on the user's smartphone (Section 4.1). The app displays a *tx-texture* on its GUI and shifts it to stimulate the optical mouse to generate movement vectors. The shift directions and distance are chosen according to the modulation method and the data bits. The app is also responsible for managing the data to be transmitted, for example, the login credentials, data from sensors, and the like.

The *MousePath-OPT receiver* is a web script offered by the webserver and running in the web browser on the PC. The core part of the *MousePath-OPT receiver* is the decoder (Section 4.2). The decoder routine obtains movement vectors from the system's mouse interface and decodes them to obtain the bitstream transmitted by the *MousePath-OPT transmitter* (see Fig. 4).

4.1. Transmitter design

4.1.1. Modulation

The movement of the texture in the *MousePath-OPT transmitter* app can stimulate the reaction of the optical mouse sensor. The displayed texture, called *tx-texture*, and its movements determine the intensity of the reaction, *i.e.*, the amplitude of the movement vector $\vec{M}(t_i)$, and thus affect the quality of the channel. Therefore, the design goal of *tx-texture* and its movement pattern is to maximize the amplitude of the mouse movement vectors while avoiding possible movement ambiguities. There are two properties complicating the problem.

[8] Keyboard and mouse are two common HIDs, their implementations are very similar in the Bluetooth and host HID stack. However, we found that keyboard HID has several subtle issues such as handling printable/special characters, ordering press and release, ignoring repeated keys, *etc.* We then chose to use mouse movements as the carrier for a simpler and also unified solution.

[9] https://www.techrepublic.com/article/how-android-p-plans-to-turn-your-phone-into-a-bluetooth-keyboard-or-mouse

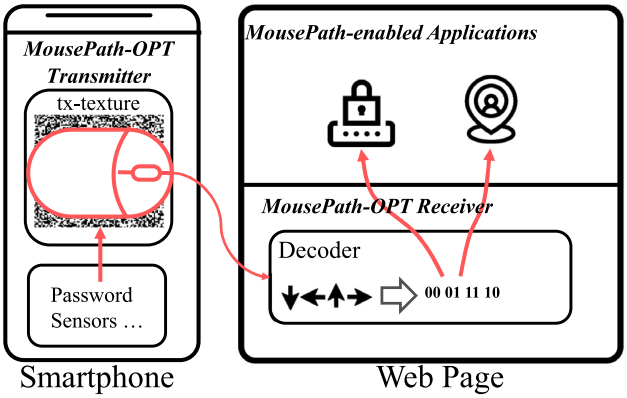


Fig. 4. Overview of MousePath-OPT. It consists of two major entities. The *MousePath-OPT transmitter* app in the smartphone and the *MousePath-OPT receiver* script embedded in the web page. The transmitter gathers information in the smartphone, such as account credentials and sensor data, and transfers them to the *MousePath-OPT receiver* to enable new web functionalities, *e.g.*, enabling Precise Location-based Service, and facilitating Password Managers.

Table 2
Modulation: Direction shift keying.

Bits	00	01	10	11	Null
Symbol	↓	←	→	↑	•

First, optical mice have a maximum detectable moving speed, which is related to the sampling rate of the image sensor. Recall that the mouse DSP measures movements by comparing consecutive images, hence the displacement between two images should not be too large, otherwise the DSP will fail to judge the correct displacement. The maximum moving speed of evaluated mice is limited to 30 inches per second.[10] As the image sensor samples 3000 images per second, the maximum displacement of the underlying texture between two sequential mouse images should not exceed $30/3000 = 0.01$ in, which is equal to the width of 4 pixels of the screen (400 pixels per inch).

At the same time, the movement emulated by the *tx-texture* is quite different from the real physical movement. As the screen refresh rate (60 Hz) is much slower than the mouse's sampling rate (3000 Hz), from the perspective of the optical mouse sensor, its images are identical most of the time. As a result, the maximum shift of the *tx-texture* is determined by the two images encountering the screen refresh rather than all images during the 1/60 s. Therefore, the shift distance of the *tx-texture* in two sequential display frames should be within 4 pixels.

Second, the mouse's imaging system can only cover an area of 0.04 in $\times$ 0.04 in of the working surface. When the mouse is placed on the phone's screen (the pixel density is about 400 pixels per inch). The optical mouse judges the movements according to an area of 16 pixels $\times$16 pixels of the *tx-texture*.

The above properties indicate that, in order to uniquely determine the shift direction, for any 16 pixels $\times$16 pixels area in the *tx-texture*, there should be no neighboring areas within 4 pixels that are identical or similar. We generate *tx-texture* based on this constraint. In addition, we use black and white textures to increase the contrast for the image sensor.

Similar to the modulation methods in other communication systems [30], both the shift direction and shift distance of the *tx-texture* can be used to represent bits. However, as the channel contains large noise, we fix the shift distance and only modulate the shift directions. We call the scheme *Direction Shift Keying* (DSK). Table 2 shows one possible mapping, which uses four shift directions as symbols ($\rightarrow, \downarrow, \leftarrow, \uparrow$) to represent bits. The symbol "•" represents no shift for the display frame, which is mainly used in the preamble.

4.1.2. Framing

Each packet has a preamble, which is used to identify the beginning of a packet. The preamble can also be used to find the rough boundary of symbols. We use synchronization in Section 4.2 to determine the fine boundary. Specifically, the preamble uses the following symbol sequences:

$$\leftarrow\leftarrow \bullet\bullet \rightarrow\rightarrow \bullet\bullet$$

Despite the adoption of conservative modulation schemes, in practice, the bit error rate is still much higher than the theoretical expectation. To increase the successful delivery rate of messages under such harsh conditions, we use Reed–Solomon (RS) codes [31,32] for forwarding error correction.

10 https://www.sparkfun.com/datasheets/Widgets/AV02-1278EN.pdf

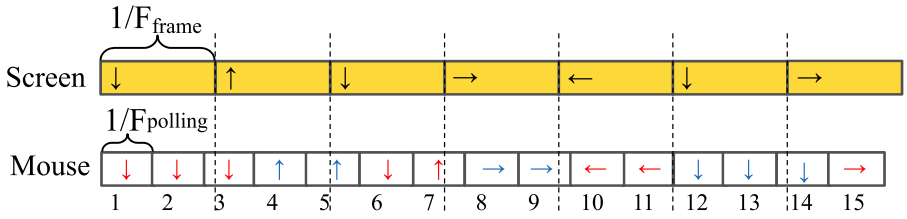


Fig. 5. Sampling Offset and Noise Exist in Received Movement Vectors. The screen refresh rate ($F_{\text{frame}} = 60$ Hz) and the mouse reporting rate ($F_{\text{polling}} = 125$ Hz) are different, bringing in a large sampling offset (see movement vectors reported at Slots 3 and 14). The movement vectors also contain noise or even wrong values (see Slot 7).

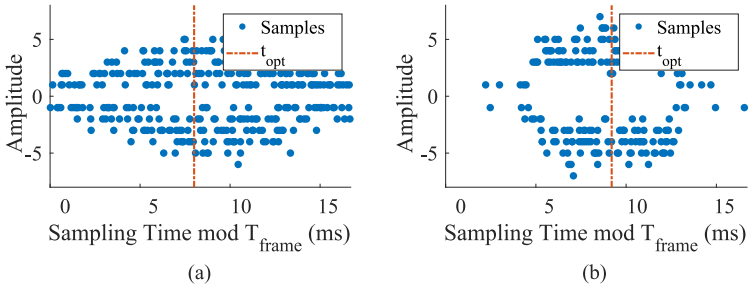


Fig. 6. Determine the Optimal Sampling Time. Dots in the figure are amplitudes of the X movement vectors folded in one refresh cycle T_{frame}, *i.e.*, [x,y] = [t_i mod T_{frame}, M_i^X]. As the mouse reports every 8 ms, samples in the range [$t_{\text{opt}} - 4$ ms, $t_{\text{opt}} + 4$ ms] mod T_{frame} are the most desirable, which are less noisy than the others. (a) Samples from mouse DELL MO56UOA. (b) Samples from mouse DELL MS111.

4.2. Receiver design

4.2.1. Receiving properties of the Optical Mouse

Fig. 5 demonstrates an example of received movement vectors. According to Section 3, the mouse should report movement only when the display frame refreshes. However, as shown in Fig. 5, in Slots 2, 4, 6, even though the screen is static during the sampling period, the mouse still reports non-static movement vectors. This is probably due to the smoothing function of the mouse's DSP, *i.e.*, the mouse reports movement consistent with the previous sampling period. Due to this reason, the first samples reported by the mouse after the screen refreshes are usually more accurate.

4.2.2. Re-sampling at the optimal sampling time

Since for each screen refresh, the first reporting event is more accurate, we want to identify them for decoding. However, as the screen and the mouse are not synchronized, the receiver does not know the exact time the screen refreshes.

We infer the screen refresh time by observing the statistics of the reported amplitude. When the reporting event is the first one after the screen refresh, its amplitude is usually large, *e.g.*, above 4 or below -4, where positives and negatives indicate the direction; when the reporting event is not the first one, its amplitude is usually smaller, *e.g.*, between -2 and 2.

We take advantage of the eye diagram to show this effect. Fig. 6 is the scatter diagram of the X movement vectors, which collectively shows vectors reported by the mouse of a certain period by folding their sampling timestamps into $[0, T_{\text{frame}}]$. We can search within this range to find the value t_{opt} where samples in $[t_{opt} - 4\text{ms}, t_{opt} + 4\text{ms}]$ have a relatively larger amplitude. This is because t_{opt} reflects the time at which the screen refreshes. We choose samples close to t_{opt} mod T_{frame} for demodulation.

4.2.3. Demodulation

Demodulation is an inverse process of modulation. The movement vectors are mapped to bits accordingly. Note that the orientation of the phone and the mouse might not be accurately aligned, and we evaluate how the misalignment angle affects the decoding performance in Section 7.

5. MousePath-BL

5.1. Bluetooth mouse emulation

This subsection briefly introduces the initial workflow of emulating the Android phone as a Bluetooth mouse, as shown in Fig. 7. The emulation process mainly includes four parts: authorization, profile selection, pairing, and connection.

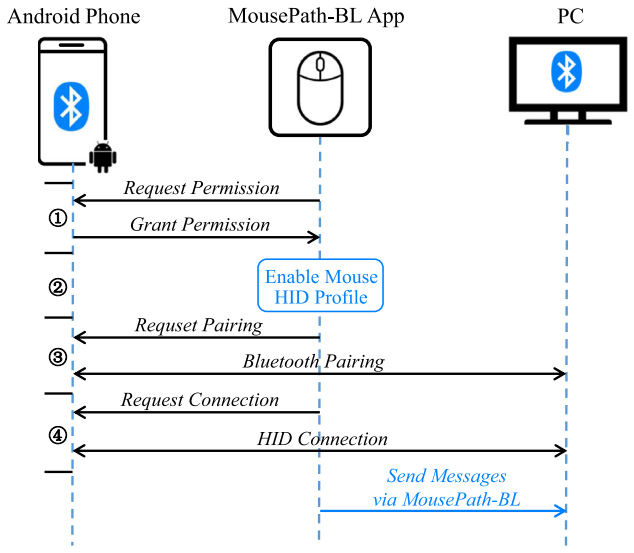


Fig. 7. Workflow of Bluetooth Mouse Emulation.

① **Authorization.** To connect to Bluetooth-enabled PCs and configure Bluetooth settings, Android apps must first declare the Bluetooth-related permissions. Besides, the permission for accessing location is also required, because Bluetooth discovery may leak location privacy.

② **Profile Selection.** Before pairing, the MousePath-BL app should select the mouse HID profile to emulate as a Bluetooth mouse. Interestingly, before a very recent update, this procedure could be done at any time in Win10 even after the pairing. Probably because it exposes a vulnerability that allows any paired Bluetooth device to change its profile [33], the latest Win10 disallows profile selection after paring.

③ **Pairing.** Pairing is the process that the Bluetooth peers use to exchange communication credentials and remember each other. It requires several user actions such as pin confirmation. Once the pairing is completed, the two devices are ready to establish the following Bluetooth connection. No pairing is required for the paired devices.

④ **Connection.** This procedure is used to confirm the connection between the paired devices. The MousePath-BL requests an HID connection with the PC's MAC address. If successful (within the Bluetooth range), it can start sending Bluetooth mouse reports to the PC, whose content could be directly received by web pages.

5.2. Transmitter design

In this subsection, we present the challenges and corresponding designs of MousePath-BL transmitter.

5.2.1. Basic modulation scheme

The modulation scheme of MousePath-BL is slightly different from MousePath-OPT. In MousePath-OPT, the movement vector is stimulated by the underlying content and contains large noise. It is also not possible to stably generate finer-grained movement vectors to increase the information capacity of each symbol. In MousePath-BL, the movement vectors are written into Bluetooth packets and the host receives them precisely. A more efficient modulation scheme is used in the MousePath-BL transmitter.

Specifically, in the Bluetooth mouse report,[11] the movement is represented by a movement vector consisting of two bytes, where each byte represents the displacement along the X-axis and Y-axis, respectively. Thus, there are up to 255×255 different movement vectors that can be used to represent information, *i.e.*, each symbol contains up to 16 bits. As a comparison, there are four movement vectors in the DSK scheme in MousePath-OPT, $(\rightarrow, \downarrow, \leftarrow, \uparrow)$, corresponding to 2 bits/symbol.

However, in practice, even though the movement vectors can be reliably received by the host system, there are two mechanisms affecting the actual performance. First, the value of the movement vector $\vec{M}$ is not the value captured by web and general GUI applications. We call the latter as the actual cursor shift and denote it as $\vec{P}$. There is a mapping between $\vec{M}$ and $\vec{P}$. This is because the host operating system introduces the mapping middle layer to facilitate GUI experience, *e.g.*, to allow the user to modify the mouse sensitivity. As a result, the mapping is different for different system settings.

[11] https://www.bluetooth.com/specifications/specs/human-interface-device-profile-1-1-1/

Second, the Bluetooth reports might be delayed for some reason. When the delay exceeds some threshold, the host operating system merges multiple delayed mouse reports into one mouse event, leading to several symbol losses (be merged to) and wrong cursor shifts (be merged by). We use the following schemes to handle these two issues.

5.2.2. Calibrating cursor shift

First of all, we describe the mapping between $\vec{M}$ and $\vec{P}$. Both $\vec{M^X}$ and $\vec{M^Y}$ are in the range of $[-127, 127]$, which is just a quantity without any unit. The cursor shift $\vec{P}$ is also a vector $[\vec{P^X}, \vec{P^Y}]$, representing the pixel distance that the cursor moves in the GUI system. The mapping between $\vec{M}$ and $\vec{P}$ is determined by the system settings. It is an unknown, maybe non-linear, but almost a deterministic function. For example, when the MousePath-BL app reports $\vec{M} = [10, 10]$, the browser receives $\vec{P} = [25, 25]$. When it reports $\vec{M} = [80, 80]$, the browser receives $\vec{P} = [204, 204]$. In order to infer bits from $\vec{P}$, the mapping function must be obtained.

We estimate the mapping function with the long-training (LT) preamble. Different from the preamble used for synchronization. The LT-preamble is used for estimating the mapping function. It covers all symbols used for encoding and remembers their relationship for decoding. It consists of all the symbols that are chosen for transmission, and the receiver remembers their values as the reference for demodulation. For instance, suppose 4 bits/symbol are used. The movement vectors are $\vec{M_i} = [M_i^X, M_i^Y] \in \{[x_i, y_j] \mid 0 \le i \le 3, 0 \le j \le 3\}$, the LT-preamble is designed as $\{[x_0, y_0], [x_1, y_1], [x_2, y_2], [x_3, y_3]\}$.

The more bits a symbol represent, the longer the LT-preamble is. It takes less than 1 min to estimate the mapping when 16 bits/symbol are used. While it can still be optimized through a better estimation algorithm (*e.g.*, interpolation) and storing in the cookie (since the estimation only needs to perform once for each web page), we adopt 6 bits/symbol in our applications in Section 6.1 to reduce the latency overhead in one-off transmission situations. Fundamentally, the choice is related to how the estimation overhead can be shared. If there are multiple transmissions, the overhead is amortized by each transmission. Otherwise, it is better to use a short training preamble, *e.g.*, in one-off situations, to reduce the latency. We adopt 6 bits/symbol in our applications (see Section 7.3) because only 20 bytes are contained in one message, and thus this choice achieves the most balanced latency performance in the use cases. However, 6 bits/symbol is not the ideal choice for all cases. For example, for long-term data transmission that needs multiple messages, *e.g.*, to transmit a file, dense symbols should be used to reduce the overall latency.

5.2.3. Combating delayed mouse reports

In practice, we observed some wrong cursor shifts as well as some missed cursor shift events. To figure out the root cause, we conduct the following experiment.

10,000 mouse reports are generated by the MousePath-BL transmitter app. They all convey the same movement vector $\vec{M_i} = [10, 10],\ i \in [0, 9999]$. The received cursor shifts $\vec{P}$ are recorded in Table 3. Most $\vec{P}$ (98.22%) equals to [25, 25]. 25 is caused by the mapping function. The rest cursor shifts are either wrong (0.48%), *i.e.*, [50, 50], [125, 125], [75,75], and others, or lost ($1.3\% = 1-98.7\%$). Coincidentally, most of the wrong values are multiples of the correct value. The reason behind this is that the operating system merges the delayed mouse reports into one mouse event. As the transmitted movement vectors are the same, the merged cursor shift is a multiple of 25. Thus, the merged cursor shift is a wrong value and the delayed movement vectors are erased, *i.e.*, the 1.3% loss. To combat this problem, we first filter the wrong values by choosing a special modulation symbol set, and then use channel coding to contain the lost symbols.

Modulation. To explain the scheme, we first show the results when transmitting 10,000 random symbols from a selected symbol set, *e.g.*, an 8×8 blue dot matrix shown in Fig. 8(a). The blue dots are viable cursor shifts, which are obtained through the LT-preamble. The red dots are wrong shifts. We note that some of the wrong shifts overlap with the viable ones, which causes confusion when judging the symbol.

Our modulation scheme is to select a special movement vector set to ensure that any merged cursor shifts can be differentiated from the viable ones. Fig. 8(b) shows the idea. We defined a viable region, within which the maximum value of the cursor shifts is smaller than the double of the minimum values. In other words, if the viable cursor shifts are from this region, no matter how the cursor shifts are merged, the merged value must exceed the region. We choose the movement vectors in this region for modulation. By doing so, no merged shifts will overlay with the viable ones, and the merged ones can be filtered to avoid confusing the demodulation.

Channel Coding. Since all the wrong values have been filtered out and lost symbols can be identified through timestamps, the channel between the MousePath-BL transmitter and MousePath receiver becomes an erasure channel, where symbols are either received or lost. It can also be handled by Reed–Solomon codes. Since the receiver knows which symbol is lost, compared with MousePath-OPT, its RS coding has better recovery performance. The packet structure is the same as MousePath-OPT's.

5.3. Receiver design

The procedure is similar to MousePath-OPT. The receiver detects the LT-preamble and then uses the estimated mapping to calibrate the cursor shifts. Then, it filters the infeasible symbols and remaps the viable ones to the data bits. After RS decoding, the correct message is obtained.

Table 3
Captured cursor shifts when transmitting $\vec{M} = [10, 10]$ in MousePath-BL App.

Cursor Shift $\vec{P} = [P^X, P^Y]$	[25,25]	[50,50]	[125,125]	[75,75]	others
Percentage (%)	98.22	0.31	0.09	0.05	0.03
Total (%)	98.70				

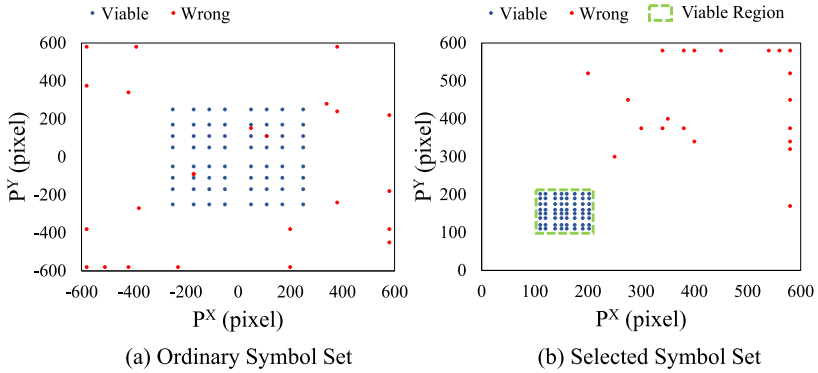


Fig. 8. Example of Cursor Shifts. Blue dots represent all the possible viable shifts. Red dots are the wrong (merged) values. (a) It is possible that the merged dots overlap with viable dots, leading to decoding errors. (b) By selecting a set of appropriate movement vectors, the wrong values are excluded from the viable region and hence can be differentiated.

6. Implementation

We first give two examples to show how MousePath can be used in practice, then describe the implementation details.

6.1. Mousepath enabled applications

To demonstrate the potential of applying MousePath to enhance PC web page functionalities, we implement two web applications shown in Fig. 9 and described below.

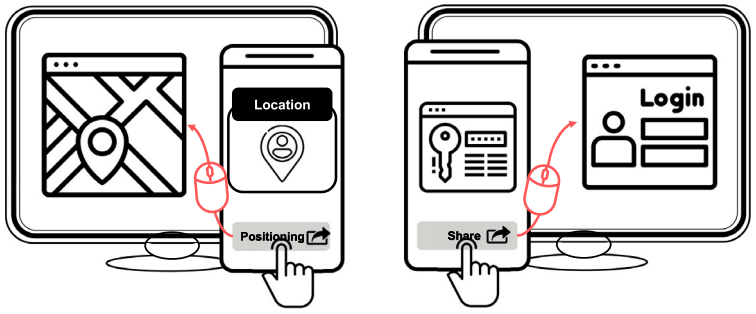
6.1.1. Precise location-based service (LBS)

When using web maps, online shopping, *etc.*, a precise location allows for a better user experience, *e.g.*, auto-filling the shipping address and recommendation of nearby coupon events. However, common indoor PCs lack dedicated localization sensors. They mainly rely on IP addresses to determine coarse-grained locations at the building or street-block level (GeoIP). In regions relying on NATs for Internet access, location errors can span hundreds of meters. On the contrary, the localization approaches of today's smartphones are more diverse and accurate. When GPS is not available indoors, many of its attributes like the locations of cellular base stations and Wi-Fi access points can be taken advantage of. Smartphones supporting Wi-Fi RTT can even be localized at the room level. MousePath enables precise LBS for PC web pages by retrieving the location information from co-located smartphones.

6.1.2. Password manager companion

Today's web passwords are complicated and are a burden for people to remember. One popular mitigating way is to store credentials in one place, such as password managers, which lock passwords in a USB dongle or cloud and do auto-filling via a simple click or copy-pasting action. However, it is not always convenient to use password managers across devices. For example, in public PCs, such as consoles in libraries, the USB interface is usually blocked and the login process to the cloud is subject to the risk of leaking the master key of the manager. MousePath allows the web page to retrieve credentials stored in the smartphone without the above limitations.[12] We note that scanning QR-code is a popular way for authentication, but the approach either requires an accompanying app of that website to be installed on the smartphone, which might not be preferred by many users, or the web servers needs to respond to the QR-code and relate it to the web application, which is possible but technically heavy for websites running a small business. MousePath allows the user to use one app to input for different websites, whilst the server only needs to embed a piece of Javascript code for receiving, which is convenient for the users as well as the website maintainers.

[12] Note that the path to the PC can be further secured by encrypting the MousePath channel with a one-time key, *e.g.*, the web server send it to the smartphone via Short Message Service (SMS).



(a) Precise Location-based Service (b) Password Manager Companion

Fig. 9. MousePath-enabled Web Applications. We implement two applications to demonstrate the potential of applying MousePath to enhance web interactions on PCs. (a) PCs lack precise localization sensors, whereas smartphones do not. MousePath enables precise LBS on PC web pages through retrieving the location information from a co-located smartphone. (b) The login credentials stored in the smartphone are shared to the web page for auto-filling.

6.2. Mousepath-OPT transmitter

The MousePath-OPT transmitter app calls the C++ Reed–Solomon library through Java Native Interface (JNI) for data encoding. The Galois field $GF(2^6)$ is used in Reed–Solomon coding. In order to modulate the movement vectors, the MousePath-OPT transmitter app displays a 500 × 500 pixels *tx-texture* on the screen. Due to the limitation of screen refresh rate, the *tx-texture* is updated 60 times per second, *i.e.*, 60 symbols are sent every second. Each symbol contains 2 bits, so the throughput of MousePath-OPT is 120 bps. An example of the MousePath-OPT transmitter app's UI is shown in Fig. 10(b). The location data or the password are encoded in the *tx-texture*. Then the user can put a mouse on top of the *tx-texture* to start the data transmission.

In Android, the display pipeline is synchronized by the VSync signal. When the VSync comes, the UI will display the previous frame if the rendering of the current frame has not been finished. This phenomenon is called Jank [34]. To avoid Jank, the transmitter app uses the GPU instead of the CPU for rendering, and binds the *tx-texture* to the GPU texture cache. This ensures that the rendering latency is within 16 ms.

6.3. MousePath-BL transmitter

The MousePath-BL transmitter encodes data in the same way as MousePath-OPT, but it directly transmits the mouse reports through Bluetooth. MousePath-BL uses Bluetooth APIs of Android, *e.g.*, `BluetoothAdapter` and `BluetoothHidDevice`, to emulate a Bluetooth mouse. MousePath-BL is configured to send a mouse report to the PC every 10 ms, *i.e.*, 100 symbols are sent every second. Each symbol contains 6 bits, so the throughput of MousePath-BL is 600 bps.

6.4. MousePath receiver

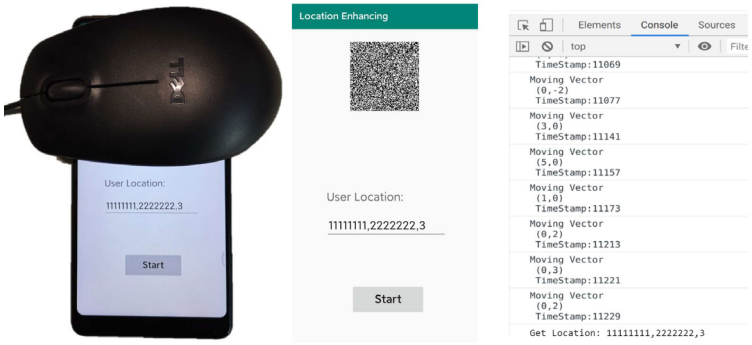
We implement the MousePath receiver as a Javascript application so that it can be embedded into web pages. The MousePath receiver uses the `Mouse Lock API` to constrain the target mouse cursor within an HTML widget to prevent the cursor from going past the boundary of the window. It captures and demodulates the cursor shifts from the `mousemove` events, and then uses the JavaScript Reed–Solomon library for error correction. As shown in Fig. 10(c), the script can successfully decode the information. There is no big difference between MousePath-OPT and MousePath-BL in the receiver implementation.

7. Evaluation

In this section, We evaluate the performance of MousePath-OPT and MousePath-BL from the communication aspects. Then, we discuss their effectiveness in the two web applications.

7.1. Evaluation of MousePath-OPT

In this part, we use the symbol error rate (SER) as the metric to evaluate the texture-related and device-related factors for MousePath-OPT. By default, we use a Xiaomi Mix 2S smartphone as the transmitter and a desktop equipped with a Logitech M100 optical mouse as the receiver. The MousePath-OPT transmitter sends 60 symbols per second. Each SER value is calculated over 10,000 symbols.



(a) (b) (c)

Fig. 10. Implementation of MousePath-OPT. (a) Putting the mouse on the smartphone to use it. (b) UI for location sharing. (c) The example output from the browser's console shows captured cursor shifts and timestamps.

7.1.1. Texture-related factors

We first evaluate how the texture-related parameters affect the performance of MousePath-OPT. We focus on the following four parameters:

(a) The moving distance of the *tx-texture* in consecutive frames, which is called **shift step** s.
(b) The size of the minimum pixel block in the *tx-texture*, which is called **texture granularity** g.
(c) The content of the *tx-texture*, *i.e.*, **pattern**.
(d) The **misaligned angle** between the mouse and the *tx-texture*.

First, we evaluate MousePath-OPT with different combinations of shift step s and texture granularity g with the default *tx-texture*. Fig. 11(a) shows the average SER for s from 1 pixel to 4 pixels and g from 2×2 pixels to 4×4 pixels. In all these cases, the SERs first decrease and then increase. As mentioned in Section 4.1, the maximum value of s should be less than 4 pixels. From Fig. 11(a), we can see that 2 and 3 pixels are the most suitable values for s as 1 pixel is too small to trigger the movement. Similar to s, a large or small g is not good for the performance. Therefore, we set s to be 2 pixels and g to be 3×3 pixels as the default parameters in MousePath-OPT.

Then, we evaluate the impact of the pattern of the *tx-texture*. We test three patterns: the hand-crafted symmetric and asymmetric patterns with simple geometric shapes, and the random pattern with random black and white pixel blocks. Fig. 12 shows examples of the *tx-textures*. Fig. 11(c) plots the CDF of SER when using different *tx-textures*. The performance of the symmetric pattern is the worst, this is because its shifts have self-similarity, which leads to ambiguities in the DSP chip when determining the shift direction and distance. The performance of the random and asymmetric patterns is close. By default, we use the random pattern.

Next, we evaluate the impact of device alignment. We define the correct usage of the MousePath-OPT as being to put the mouse on top of the screen and parallel to the short edge of the screen, as shown in Fig. 10(a). However, in practice, there may be a misaligned angle between the *tx-texture* and the mouse sensor. Fig. 11(b) plots the CDF of SER versus the angle. When the angle increases, the performance becomes worse. When the angle is 20° or above, moving upward can easily be interpreted as moving leftward or rightward, and thus the SER is high; when the angle is 10° or less, with a high probability the SER is less than 10%. According to our experience in placing the mouse, the angle is usually within 10°. Besides the angle, the location of the mouse might also affect the performance. As long as the mouse sensor is covered by the *tx-texture* (this can be ensured by showing a sufficiently large *tx-texture* on the transmitter UI), the relative location of the mouse and the *tx-texture* has no impact on the performance. This is because the movement is tracked by the difference of the two shifted *tx-texture* frames rather than the content of the *tx-texture*. However, if the mouse is moved during data transmission, the movement induced by the display content will be affected and cause decoding error. In this case, MousePath-OPT will not work unless the movement is stopped.

7.1.2. Device-related factors

In this part, we evaluate the performance of MousePath-OPT on different smartphones and mouse models. We choose Xiaomi Mix 2S (with an LCD screen), Xiaomi 8 (with an OLED screen) and Huawei Mate 20 (with an LCD screen). For the optical mouse, we choose DELL MS111, DELL MO56UOA and Logitech M100 as the receiver. Fig. 11(d) plots the CDF of SER of different smartphone–mouse combinations. Different mice have different performances, as their movement detection algorithms are different. The mouse model with the best performance on Xiaomi Mix 2S (Curve DELL MS111, Logitech M100, and DELL MO65UOA) is DELL MS111. We use it as the default receiver in the following experiments. Fig. 11(d) also shows the performance of different types of screens when using DELL MS111 as the receiver. We note that in general, the performance of LCD and OLED is close. Because of Huawei's unstable screen refresh rate, Xiaomi's performance is slightly better.

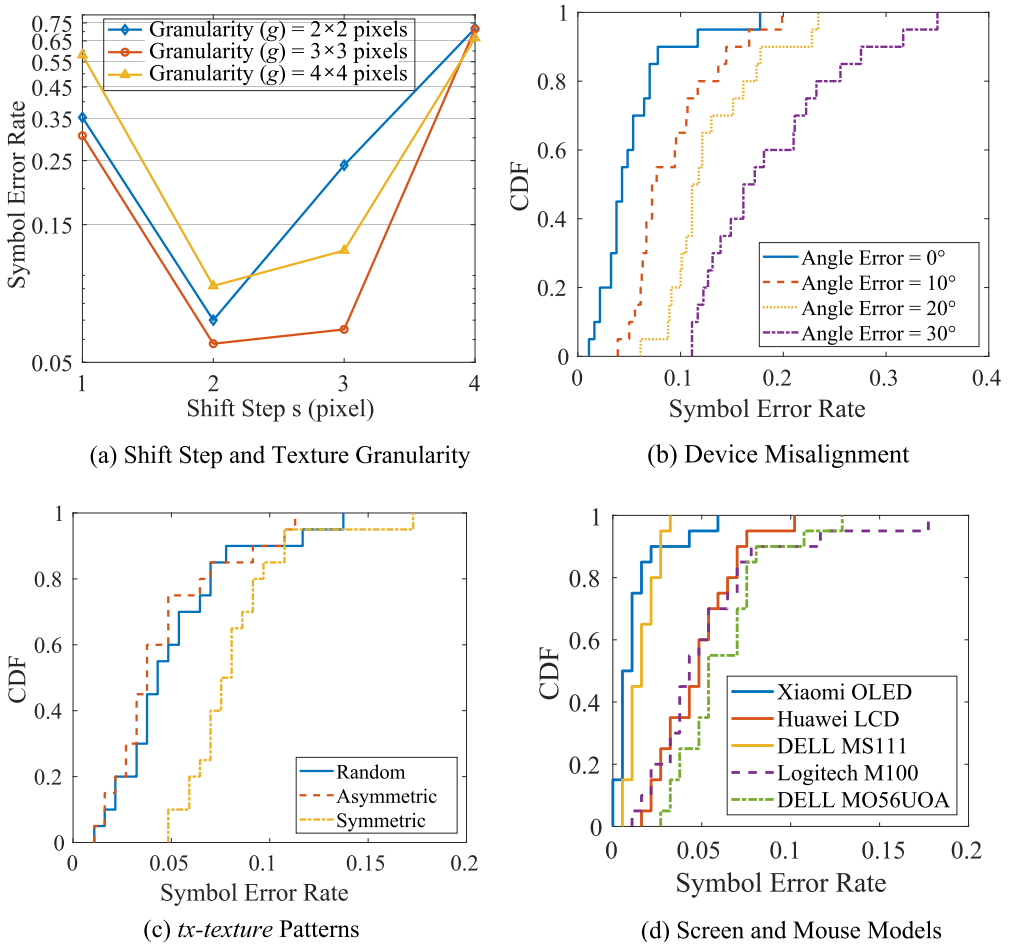


Fig. 11. MousePath-OPT Error Rate v.s. Different Configurations.

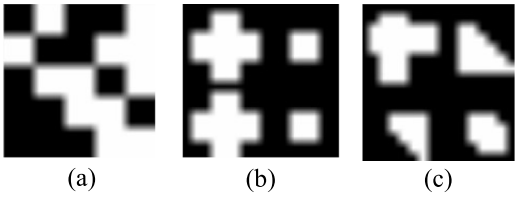


Fig. 12. Examples of *tx-texture* Pattern. (a) Random pattern. (b) Symmetric pattern. (c) Asymmetric pattern. A pattern consists of 5 × 5 black and white 3 × 3 pixel blocks. The full *tx-texture* consists of repeating patterns.

7.2. Evaluation of MousePath-BL

In this part, we evaluate the performance of MousePath-BL. We use a Xiaomi Mix 2S smartphone as the default transmitter and a laptop with a Bluetooth interface as the receiver. We mainly focus on the following two parameters: (a) The distance between the smartphone and the collocated PC. (b) The smartphone model. We use the symbol loss rate (SLR) to measure the performance. The MousePath-BL transmitter sends 100 symbols per second. Each SLR value is calculated over 10,000 mouse reports. Each test is repeated 10 times to obtain the standard deviation. The SLR counts the ratio of lost cursor shifts after the filtering. The loss is mainly caused by the merged reports. Incorrect filtering and Bluetooth packet loss also contribute to the SLR.

In daily use, the distance between the smartphone and the PC is likely within 100 cm. We measure the average SLR under different distances from 1 cm to 100 cm. As shown in Fig. 13(a). The SLRs are stable and around 2%. Additionally,

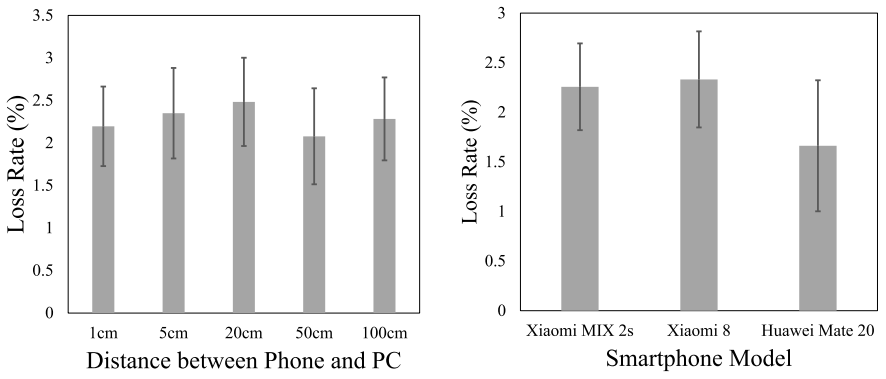


Fig. 13. MousePath-BL Symbol Loss Rate v.s. Different Configurations.

Table 4
Localization error w/ and w/o MousePath.

Localization method	Localization error (m)			
	Dormitory	*Canteen*	*Library*	*Lab*
IP Based	291	365	433	380
Smartphone's localization API	42	19	5	11

Xiaomi 8 and Huawei Mate 20 are used as the receiver to understand the impact of different smartphone models. Fig. 13(b) shows that the performance of the three models is very close. The above results suggest that both distance and smartphone model have little impact on the MousePath-BL performance. This is because they are based on certified Bluetooth chips, which offer uniform performance.

7.3. Evaluation of MousePath enabled applications

This subsection evaluates the effectiveness of applying MousePath to enhance two web applications.

7.3.1. Application I: Precise location-based service.

In the transmitter app, we extract the location estimation of the smartphone through the Android localization API and share it to the PC web page through MousePath. By default, the PC web page uses the Geo-IP database for getting the estimated location. We sample locations in different places and compare the accuracy of the two approaches in Table 4. In all the locations, IP-based localization is not accurate. This is because the resolution of the Geo-IP database in our area is blurred by NAT. On the contrary, the location obtained from the smartphone is more accurate since it uses information from, *e.g.*, the locations of observable base stations.

MousePath can also be used to share other sensors of the co-located smartphone. Specifically, MousePath can be used to convey voice commands to allow users to control the PC web application with the voice. For example, when watching online courses, it is common to roll back and forth of the video, but it is not convenient to take notes at the same time. The smartphone can act as an agent to first recognize the speech and then transmit the commands to the web application via MousePath. Similarly, MousePath can be used to share the ambient light sensor, so that the web page can adjust the brightness of its background automatically. This is helpful in improving the user experience of online media editing and moving watching.

7.3.2. Application II: Password manager companion.

With MousePath, a smartphone can share passwords to the PC applications by coding the information in cursor shifts. We measure the delay of transmitting a 20-byte password and compare MousePath with other network technologies.

In the following, we compare the delay of MousePath with two password-sharing solutions that use built-in Bluetooth (original Bluetooth) and Wireless LAN (WLAN) sharing respectively. We use a Xiaomi Mix 2S smartphone as the transmitter and a PC as the receiver. For WLAN sharing, we use `KDE Connect` and assume the two devices have already been in the same LAN, *i.e.*, the user does not need to connect them to the same Wi-Fi network. 10 participants (5 males and 5 females) are asked to use the above solutions to transmit the password in the smartphone to the web application opened on the PC. They are aged from 22 to 26 (mean = 24.1, SD = 1.22). Among them, 5 participants are postgraduates of Computer Science and Technology and others are engaged in different occupations, including one junior high school teacher, one kindergarten teacher, one translator, one accountant and one software developer. All of them are experienced

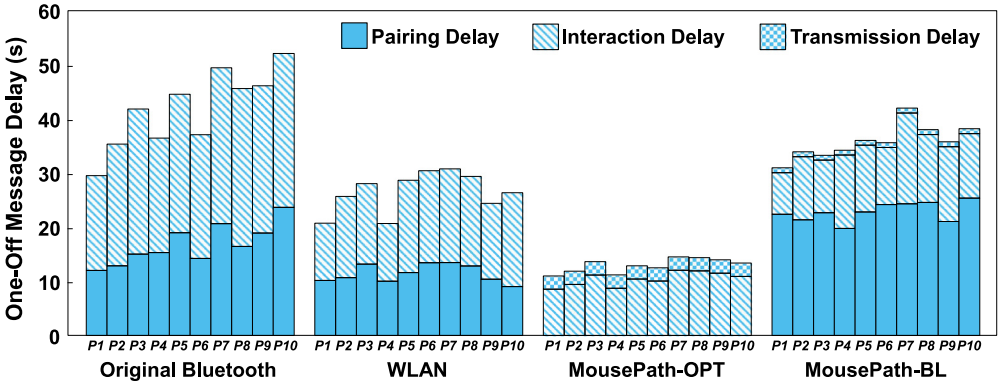


Fig. 14. Delay Decomposition of Different Sharing Solutions. The delay is measured when sharing a 20-byte password to a PC with different sharing solutions. The experiments are conducted by ten participants ($P1$, $P2$, . . ., $P10$).

Table 5
User's subjective satisfaction of the four sharing solutions with a 1-10 rating scale.

	Original bluetooth	WLAN	MousePath-OPT	MousePath-BL
Score (1-10)	7.00	7.56	8.67	6.78

PC users. Before the experiments, we demonstrated the operations to the participants. The experiments did not begin until they claimed they were familiar with the operation. The delays of the test are logged and each test is repeated 3 times. It takes every participant about half an hour to learn and complete the tests.

The overall delay consists of three parts. Device pairing delay is the time used to set up the connection between the smartphone and the PC, *e.g.*, selecting the network, typing in the password, *etc.* Transmission delay is the time that is actually used for data transmission. The remaining part of the delay is interaction delay. It is the time that the user spends interacting with the device and PC after pairing. *e.g.*, putting the mouse on the smartphone, copying and pasting the password to the web browser, *etc.*

Fig. 14 shows the results. We note that the transmission delay of the original Bluetooth and WLAN is too short to be shown in the bars. WLAN is faster than the original Bluetooth sharing since it automatically pops up a file showing the transmitted password and does not need user actions to open the file. However, both of them are not convenient since they are only *phone-to-PC* solutions. Users still need to manually copy and paste the password to the web page from the Bluetooth File Transfer window or the Notepad, which is very cumbersome. Both MousePath-OPT and MousePath-BL are *phone-to-web* solutions. Although MousePath-OPT has a large transmission delay, its total delay is the smallest since it does not need pairing. MousePath-BL has a higher throughput but larger pairing delay than MousePath-OPT, hence MousePath-BL is more efficient for long-term transmission while MousePath-OPT is more suitable for transferring short messages in one-off sharing situations.

After completing all the tests, the participants are asked to rate the four sharing solutions with a 1–10 score. The score represents the satisfaction of using each solution, *i.e.*, the will to adopt the solution in the future to achieve similar aims. A higher score means the participants prefer the corresponding solution. Table 5 lists the average scores of the four solutions. MousePath-OPT gets the highest score because it is really fast and convenient. The scores of original Bluetooth sharing, WLAN, and MousePath-BL are close. Due to the smaller paring delay, WLAN is more popular than MousePath-BL. However, it needs an accompanying software, *i.e.*, `KDE Connect`, to be installed on the PC, which is not counted in the test. We note that even though the original Bluetooth sharing's overall delay is larger, it is slightly more popular than MousePath-BL. This is mainly because they both need similar Bluetooth pairing and the participants are more familiar with the conventional way. We believe as long as they get familiar with MousePath-BL, they will notice the delay difference and prefer the more efficient one.

8. Discussion

Potential Extensions. MousePath implies interesting security properties worth further exploration. For example, MousePath can be used to establish a secure user input system for public computers. The user could use the keyboard of his/her personal smartphone to do the input for the application via MousePath. This is without worrying about the risk of input leakage even when the underlying computer system is fully compromised. The key challenge is how to establish a trusted data path between the smartphone and the application. We plan to combine trusted computing technologies with MousePath to achieve this goal.

Mouse Sensitivity. In practice, we found that there are optical mice that cannot respond to texture shifts in the smartphone's screen, and thus cannot be used in MousePath. This is because of many aspects, including the intensity of the backlight, the focal length of the mouse lens, the movement detection algorithms in DSP, *etc.* To use these mice, the user might need smartphones with a thinner and brighter screen. Besides, on the mouse side, if the optical mouse can actually report sampled raw images [35], the texture can be directly used for conveying data like a QR-code. We plan to explore this property to improve the compatibility of MousePath in future work.

Reducing Latency. For MousePath-OPT, the main latency is limited by the data rate, which is determined by two factors, the screen refresh rate and the mouse polling rate. The mouse polling rate can be manually configured[13] to 1000 Hz and smartphones with a high refresh rate (*e.g.*, 120 HZ), which are common in the latest smartphones, can be used to reduce the delay. For MousePath-BL, the pairing delay is limited by the Bluetooth stack, which is hard to reduce but is a one-time overhead.

JavaScript Security. JavaScript (JS) is widely used in web applications. MousePath is implemented with the commonly-used JS *PointerEvents* API and basic JS operations. The security issues of JS has been extensively discussed in the literature. Most JS vulnerabilities are caused by untrusted scripts. Existing work proposed approaches to detect, filter, and thwart malicious scripts [36,37]. Also, under the context of advanced attacks, the JS execution engine, *e.g.*, the browser, can be secured with trusted computing hardware [38].

9. Conclusion

This paper presents MousePath, a direct and lightweight communication channel between smartphones and web applications based on the mouse interface. We propose two modalities of MousePath. MousePath-OPT makes use of a smartphone's screen as the transmitter and the optical mouse as the receiver. MousePath-BL emulates the smartphone as a Bluetooth mouse to transmit data via mouse events. MousePath fulfills the demand for *device-to-web* information sharing. We envision that it can enable a lot of interesting web applications.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Zhice Yang reports financial support was provided by National Natural Science Foundation of China. Qianyi Huang reports financial support was provided by National Natural Science Foundation of China.

Data availability

Data will be made available on request.

Acknowledgments

We thank Stephan Sigg for the valuable discussion and suggestions. This work is supported (in part) by the ShanghaiTech, China Startup Fund, and NSFC, China 62002224 and 62002150.

References

[1] Z. Wang, Q. Huang, Y. Yan, H. Ren, Y. Zhang, Z. Yang, Mousepath: Enhancing PC Web Pages through Smartphone and Optical Mouse, in: 19th IEEE International Conference on Pervasive Computing and Communications, PerCom 2021, Kassel, Germany, March 22-26, 2021, IEEE, pp. 1–7, http://dx.doi.org/10.1109/PERCOM50583.2021.9439139.

[2] S.D. Perli, N. Ahmed, D. Katabi, Pixnet: interference-free wireless links using lcd-camera pairs, in: Proceedings of the 16th Annual International Conference on Mobile Computing and Networking, MOBICOM 2010, Chicago, Illinois, USA, September 20-24, 2010, ACM, pp. 137–148, http://dx.doi.org/10.1145/1859995.1860012.

[3] W. Hu, H. Gu, Q. Pu, Lightsync: unsynchronized visual communication over screen-camera links, in: The 19th Annual International Conference on Mobile Computing and Networking, MobiCom'13, Miami, FL, USA, September 30 - October 04, 2013, ACM, pp. 15–26, http://dx.doi.org/10.1145/2500423.2500437.

[4] Z. Yang, Z. Wang, J. Zhang, C. Huang, Q. Zhang, Wearables Can Afford: Light-weight Indoor Positioning with Visible Light, in: Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys 2015, Florence, Italy, May 19-22, 2015, ACM, pp. 317–330, http://dx.doi.org/10.1145/2742647.2742648.

[5] K. Zhang, C. Wu, C. Yang, Y. Zhao, K. Huang, C. Peng, Y. Liu, Z. Yang, Chromacode: A Fully Imperceptible Screen-Camera Communication System, in: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, MobiCom 2018, New Delhi, India, October 29 - November 02, 2018, ACM, pp. 575–590, http://dx.doi.org/10.1145/3241539.3241543.

[6] V. Nguyen, Y. Tang, A. Ashok, M. Gruteser, K.J. Dana, W. Hu, E. Wengrowski, N.B. Mandayam, High-rate flicker-free screen-camera communication with spatially adaptive embedding, in: 35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016, IEEE, pp. 1–9, http://dx.doi.org/10.1109/INFOCOM.2016.7524512.

[7] P.H. Kindt, M. Saur, M. Balszun, S. Chakraborty, Neighbor Discovery Latency in BLE-Like Protocols, IEEE Trans. Mob. Comput. 17 (3) (2018) 617–631, http://dx.doi.org/10.1109/TMC.2017.2737008.

13 https://wiki.archlinux.org/title/Mouse_polling_rate

[8] R. Nandakumar, K.K. Chintalapudi, V.N. Padmanabhan, R. Venkatesan, Dhwani: secure peer-to-peer acoustic NFC, in: ACM SIGCOMM 2013 Conference, SIGCOMM 2013, Hong Kong, August 12-16, 2013, ACM, pp. 63–74, http://dx.doi.org/10.1145/2486001.2486037.
[9] A. Mehrabi, A. Mazzoni, D. Jones, A. Steed, Evaluating the user experience of acoustic data transmission, Pers. Ubiquitous Comput. 24 (5) (2020) 655–668, http://dx.doi.org/10.1007/s00779-019-01345-7.
[10] X. Liang, T. Yun, R.A. Peterson, D. Kotz, Lighttouch: securely connecting wearables to ambient displays with user intent, in: 2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, May 1-4, 2017, IEEE, pp. 1–9, http://dx.doi.org/10.1109/INFOCOM.2017.8057210.
[11] B.A. Myers, Using handhelds and PCs together, Commun. ACM 44 (11) (2001) 34–41, http://dx.doi.org/10.1145/384150.384159.
[12] A.L. Simeone, J. Seifert, D. Schmidt, P. Holleis, E. Rukzio, H. Gellersen, A cross-device drag-and-drop technique, in: 12th International Conference on Mobile and Ubiquitous Multimedia, MUM '13, Luleå, Sweden, December 02-05, 2013, ACM, pp. 10:1–10:4, http://dx.doi.org/10.1145/2541831.2541848.
[13] S. Oh, H. Yoo, D.R. Jeong, D.H. Bui, I. Shin, Mobile Plus: Multi-device Mobile Platform for Cross-device Functionality Sharing, in: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys'17, Niagara Falls, NY, USA, June 19-23, 2017, ACM, pp. 332–344, http://dx.doi.org/10.1145/3081333.3081348.
[14] H. Xia, J. Zhang, Y. Zhu, C. Yu, Y. Shi, Mobile assistant: enhancing desktop interaction using mobile phone, in: Interactive Tabletops and Surfaces, ITS'12, Cambridge/Boston, MA, USA, November 11-14, 2012, ACM, pp. 379–382, http://dx.doi.org/10.1145/2396636.2396705.
[15] R. Sharp, A. Madhavapeddy, R. Want, T. Pering, Enhancing web browsing security on public terminals using mobile composition, in: Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys 2008), Breckenridge, CO, USA, June 17-20, 2008, ACM, pp. 94–105, http://dx.doi.org/10.1145/1378600.1378612.
[16] A. Perrig, M.K. Reiter, J.M. Mccune, Reducing the trusted computing base for applications on commodity systems, 2009.
[17] G. Perelman, M. Serrano, M. Raynal, C. Picard, M. Derras, E. Dubois, The Roly-Poly Mouse: Designing a Rolling Input Device Unifying 2D and 3D Interaction, in: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015, ACM, pp. 327–336, http://dx.doi.org/10.1145/2702123.2702244.
[18] S. Kim, H. Kim, B. Lee, T. Nam, W. Lee, Inflatable mouse: volume-adjustable mouse with air-pressure-sensitive input and haptic feedback, in: Proceedings of the 2008 Conference on Human Factors in Computing Systems, CHI 2008, 2008, Florence, Italy, April 5-10, 2008, ACM, pp. 211–224, http://dx.doi.org/10.1145/1357054.1357090.
[19] J. Cechanowicz, P. Irani, S. Subramanian, Augmenting the mouse with pressure sensitive input, in: Proceedings of the 2007 Conference on Human Factors in Computing Systems, CHI 2007, San Jose, California, USA, April 28 - May 3, 2007, ACM, pp. 1385–1394, http://dx.doi.org/10.1145/1240624.1240835.
[20] X. Yang, E. Mak, D.C. McCallum, P. Irani, X. Cao, S. Izadi, Lensmouse: augmenting the mouse with an interactive touch display, in: Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, Atlanta, Georgia, USA, April 10-15, 2010, ACM, pp. 2431–2440, http://dx.doi.org/10.1145/1753326.1753695.
[21] W. Park, S.H. Park, L. Kim, S. Shin, Haptic Mouse Interface Actuated by an Electromagnet, in: International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2011, June 30 - July 2, 2011, Korean Bible University, Seoul, Korea, IEEE Computer Society, pp. 643–646, http://dx.doi.org/10.1109/CISIS.2011.107.
[22] F. Kiss, V. Schwind, S. Schneegass, N. Henze, Design and evaluation of a computer-actuated mouse, in: Proceedings of the 16th International Conference on Mobile and Ubiquitous Multimedia, MUM 2017, Stuttgart, Germany, November 26 - 29, 2017, ACM, pp. 261–271, http://dx.doi.org/10.1145/3152832.3152862.
[23] G. Bailly, S. Sahdev, S. Malacria, T. Pietrzak, Livingdesktop: Augmenting Desktop Workstation with Actuated Devices, in: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, May 7-12, 2016, ACM, pp. 5298–5310, http://dx.doi.org/10.1145/2858036.2858208.
[24] X. Bi, T. Grossman, J. Matejka, G.W. Fitzmaurice, Magic desk: bringing multi-touch surfaces into desktop work, in: Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011, ACM, pp. 2511–2520, http://dx.doi.org/10.1145/1978942.1979309.
[25] H. Pan, Y. Chen, G. Xue, X. Ji, Magnecomm: Magnetometer-based Near-Field Communication, in: Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, MobiCom 2017, Snowbird, UT, USA, October 16-20, 2017, ACM, pp. 167–179, http://dx.doi.org/10.1145/3117811.3117824.
[26] N. Roy, M. Gowda, R.R. Choudhury, Ripple: Communicating through Physical Vibration, in: 12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 15, Oakland, CA, USA, May 4-6, 2015, USENIX Association, pp. 265–278, https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/roy.
[27] N. Roy, R.R. Choudhury, Ripple II: Faster Communication through Physical Vibration, in: 13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016, USENIX Association, pp. 671–684, https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/roy.
[28] T. Yonezawa, J. Nakazawa, H. Tokuda, Vinteraction: Vibration-based information transfer for smart devices, in: Eighth International Conference on Mobile Computing and Ubiquitous Networking, ICMU 2015, Hakodate, Japan, January 20-22, 2015, IEEE Computer Society, pp. 155–160, http://dx.doi.org/10.1109/ICMU.2015.7061059.
[29] M. Bachratý, M. Žalman, 2D Position Measurement with optical laser mouse sensor, 2010.
[30] D. Tse, P. Viswanath, Fundamentals of wireless communication, Cambridge University Press, ISBN: 9780511807213, 2005, http://dx.doi.org/10.1017/CBO9780511807213.
[31] J. Li, The efficient implementation of Reed-Solomon high rate erasure resilient codes, in: 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '05, Philadelphia, Pennsylvania, USA, March 18-23, 2005, IEEE, pp. 1097–1100, http://dx.doi.org/10.1109/ICASSP.2005.1415905.
[32] J. Chen, P. Owsley, A burst-error-correcting algorithm for Reed-Solomon codes, IEEE Trans. Inf. Theory 38 (6) (1992) 1807–1812, http://dx.doi.org/10.1109/18.165456.
[33] F. Xu, W. Diao, Z. Li, J. Chen, K. Zhang, Badbluetooth: Breaking Android Security Mechanisms via Malicious Bluetooth Peripherals, in: 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019, The Internet Society, https://www.ndss-symposium.org/ndss-paper/badbluetooth-breaking-android-security-mechanisms-via-malicious-bluetooth-peripherals/.
[34] Y.-D. Lin, E.T.-H. Chu, E. Chang, Y.-C. Lai, Smoothed graphic user interaction on smartphones with motion prediction, IEEE Transactions on Systems, Man, and Cybernetics: Systems 50 (4) (2017) 1429–1441, http://dx.doi.org/10.1109/TSMC.2017.2685243.
[35] M.M. Da Silva, J.R. de Almeida Nozela, M.J. Chaves, R.A.B. Junior, H.J. Rabal, Optical mouse acting as biospeckle sensor, Optics Communications 284 (7) (2011) 1798–1802, http://dx.doi.org/10.1016/j.optcom.2010.12.037.
[36] M. Musch, M. Steffens, S. Roth, B. Stock, M. Johns, Scriptprotect: Mitigating Unsafe Third-Party JavaScript Practices, in: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019, ACM, pp. 391–402, http://dx.doi.org/10.1145/3321705.3329841.

[37] A. Taly, Ú. Erlingsson, J.C. Mitchell, M.S. Miller, J. Nagra, Automated Analysis of Security-Critical JavaScript APIs, in: 32nd IEEE Symposium on Security and Privacy, S&P 2011, Berkeley, California, USA, May 22-25, 2011, IEEE Computer Society, pp. 363–378, http://dx.doi.org/10.1109/SP.2011.39.

[38] S. Eskandarian, J. Cogan, S. Birnbaum, P.C.W. Brandon, D. Franke, F. Fraser, G.G. Jr., E. Gong, H.T. Nguyen, T.K. Sethi, V. Subbiah, M. Backes, G. Pellegrino, D. Boneh, Fidelius: Protecting User Secrets from Compromised Browsers, in: 2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019, IEEE, pp. 264–280, http://dx.doi.org/10.1109/SP.2019.00036.