Listen to Your Key: Towards Acoustics-based Physical Key Inference

Soundarya Ramesh
sramesh@comp.nus.edu.sg
Department of Computer Science
National University of Singapore

Harini Ramprasad
harinir@comp.nus.edu.sg
Department of Computer Science
National University of Singapore

Jun Han
junhan@comp.nus.edu.sg
Department of Computer Science
National University of Singapore

ABSTRACT

Physical locks are one of the most prevalent mechanisms for securing objects such as doors. While many of these locks are vulnerable to lock-picking, they are still widely used as lock-picking requires specific training with tailored instruments, and easily raises suspicion. In this paper, we propose *SpiKey*, a novel attack that significantly lowers the bar for an attacker as opposed to the lock-picking attack, by requiring only the use of a smartphone microphone to infer the shape of victim's key, namely *bittings* (or cut depths) which form the secret of a key. When a victim inserts his/her key into the lock, the emitted sound is captured by the attacker's microphone. *SpiKey* leverages the time difference between audible clicks to ultimately infer the bitting information, i.e., shape of the physical key. As a proof-of-concept, we provide a simulation, based on real-world recordings, and demonstrate a significant reduction in search space from a pool of more than 330 thousand keys to *three* candidate keys for the most frequent case.

CCS CONCEPTS

• Security and privacy → Side-channel analysis and countermeasures; **• Hardware → Sound-based input / output**.

KEYWORDS

Side-channel Attacks; Acoustic Inference; Physical Key Security

ACM Reference Format:

Soundarya Ramesh, Harini Ramprasad, and Jun Han. 2020. Listen to Your Key: Towards Acoustics-based Physical Key Inference. In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications (HotMobile '20), March 3–4, 2020, Austin, TX, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3376897.3377853

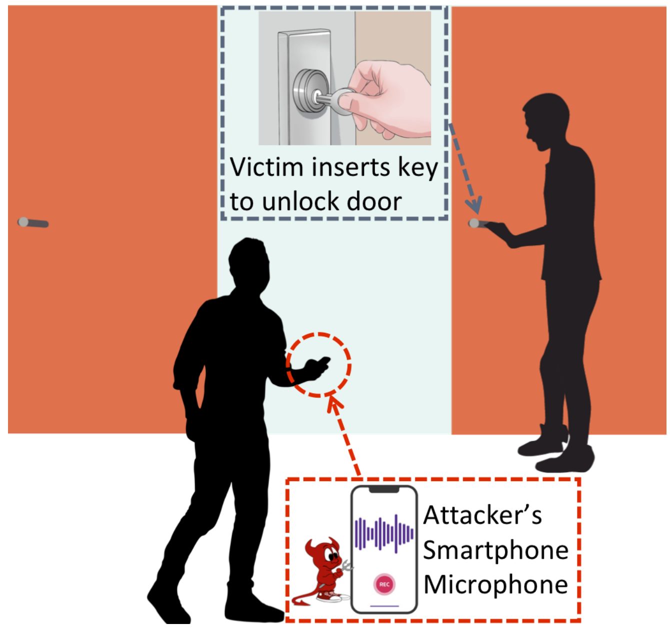

Figure 1: Figure depicts *SpiKey* attack scenario. Attacker records the sound of victim's key insertion to infer the shape, or "secret", of the key.

1 INTRODUCTION

Physical locks are the most prevalent means of securing objects including doors and mailboxes. Among many types of locks, pin tumbler locks are the most commonly used, with lock manufacturers Schlage and Yale dominating the market [6, 9, 16]. Despite the rise in digital locks, conventional pin tumblers continue to be widely deployed to secure homes and office spaces around the world [24].

However, there are several known attacks on the pin tumbler locks, with *lock picking* being one of the most widely known techniques [17, 18]. This requires an attacker to insert tailored instruments into the lock and manipulate the internal components (known as *pins*) of the locks to unlock without possession of a key. Nonetheless, lock picking has significant limitations, which is part of the reason why pin tumbler locks are still widely used. For instance, lock picking requires specific training and practice, and easily raises suspicion because it requires the attacker to insert into the lock a pair of specialized tools which is inevitably noticeable [2, 3]. In addition, lock picking inherently grants a single entry upon successful picking and also leaves traces because the picking scratches the surface of the pins [5, 22].

In light of these limitations, we pose the question – can we design an attack that is robust against the aforementioned challenges? To answer this question, we present *SpiKey*, a novel attack that utilizes a smartphone microphone to capture the sound of key insertion/withdrawal to infer the shape of the key, i.e., cut depths

HotMobile '20, March 3–4, 2020, Austin, TX, USA

ACM ISBN 978-1-4503-7116-2/20/03…$15.00
https://doi.org/10.1145/3376897.3377853

(referred to as *bittings*) that form the "secret" of the key, solely by the captured acoustic signal. For example, as illustrated in Figure 1, as a victim inserts the key into the lock, an attacker walking by the victim uses his/her smartphone microphone to capture the sound. However, it is extremely challenging to extract information from the sound to infer fine-grained bitting depths which differ by 15 milli-inch (0.381 mm). To solve this challenge, *SpiKey* captures and utilizes the time difference of audible *clicks* – that occur when *ridges* of a key (that form due to cuts of key bittings) come in contact with the pins inside the lock – to infer distances between the ridges given a constant speed of key insertion. Subsequently, *SpiKey* leverages a sequence of these inferred inter-ridge distances to ultimately infer the bittings, or secret, of the key.

Because only requiring a smartphone microphone, *SpiKey* yields many advantages such as enabling a layperson to launch the attack, in addition to significantly reducing suspicion. Moreover, as *SpiKey* infers the shape of the key, it is inherently robust against anti-picking features in modern locks [17], and grants multiple entries without leaving any traces. Overall, we make the following contributions:

- We introduce a novel attack, *SpiKey*, to infer physical keys with only a smartphone microphone.
- We present the design of the acoustics-based physical key inference attack by introducing and solving the corresponding challenges.
- We simulate based on real-world recordings and demonstrate significant reduction in search space from a pool of more than 330 thousand possible keys to *three* candidate keys for the most frequent cases.

2 LOCK AND KEY CONSTRUCTION

We briefly explain the construction of a pin tumbler lock and its key, as well as how the clicking sound occurs.

Pin tumbler lock comprises a set of six top and bottom *pins* (p_1, ..., p_6), each connected by a spring, hence moves vertically as a key is inserted. Bottom pins vary in lengths which correspond to the cut depths of a matching key. When such a key is inserted, the bottom pins are correctly positioned such that the top pins align on a shear line, allowing the key to turn, and ultimately unlocking the lock (depicted in Figure 2(a)). Adjacent pins are separated by an *inter-pin distance* (α_p).

Key comprises six *bitting* positions. For each position (b_i), the cut or *bitting depth* constitutes the "secret". Bitting depth is a discrete value ranging from 0 to b_{depth} (which ranges between 7-10 depending on key specifications). The bitting depths, $b_1 b_2 \dots b_6$, are together referred to as *keycode* (e.g., 393597). Figure 2(b) illustrates these parameters. The increase in successive depths (on the order of sub-millimeters) is referred to as *increment* (α_d). Width of each bitting position is *root cut* (α_w) and the distance between adjacent bittings is *bit spacing*, which also equals the inter-pin distance, α_p. *Cut angle* (θ) is the inclination between the two inclines, originating from the bitting positions. In addition, there is a constraint on the maximum permissible difference between adjacent bitting depths in order to prevent the inclines from reducing the *root cut* dimension, referred to as Maximum Adjacent Cut Specification, or *MACS* (μ) [22]. In this paper, we refer to one of the most widely

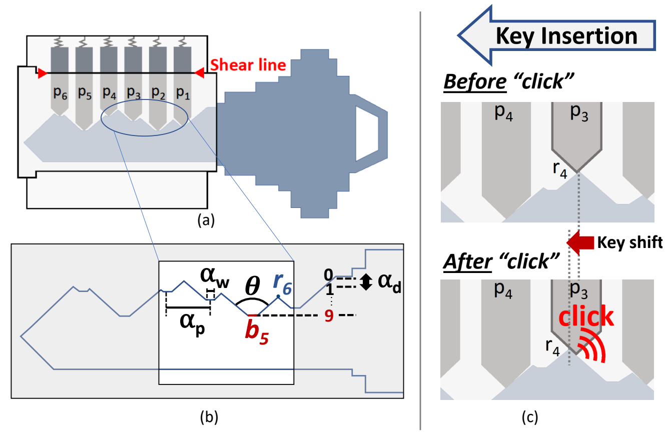

Figure 2: (a) With a correct key inserted, pins align on a shear line and unlocks the lock; (b) depicts key construction parameters; and (c) depicts key insertion producing *click* sound as a pin slips off of a key ridge.

used key type, *Schlage 6-pin C-keyway* keys (b_{depth} = 10, μ = 7). Hence, keycodes such as 230845 are not permitted because it has adjacent bitting depths that are greater than μ (e.g., difference of 0 and $8 > \mu = 7$). While an entire key space is 10^6 keys, MACS along with the bitting rules reduce it to 586, 584 [20]. We take advantage of such reduction in key space as *SpiKey* ultimately needs to reduce the key space to a subset of a small number of candidate keys.

Ridges (r_i) form as the inclines (due to bitting depths) converge (Figure 2(b)). During key insertion, *SpiKey* utilizes *click* sound that occurs as a pin slips off the top of a ridge (Figure 2(c)). Due to the presence of multiple ridges and pins, we obtain a series of *clicks* introducing more challenges. *SpiKey* utilizes the clicks to ultimately infer the distance between adjacent ridges as *inter-ridge distance* (d_i) as *all keys conform to the aforementioned construction parameters.*

3 *SPIKEY* DESIGN

We present the design of *SpiKey* and illustrate the steps involved (Figure 3). When a victim inserts a key into the door lock, an attacker walking by records the sound with a smartphone microphone. *SpiKey* detects the timing of these clicks from the sound (Section 3.1). We then utilize the click timestamps to compute the adjacent inter-ridge distances given a constant insertion speed (Section 3.2). We use the computed distances to infer the relative differences of adjacent bitting depths (Section 3.3), which *SpiKey* exploits to ultimately obtain a small subset of candidate keys that includes the victim's keycode (Section 3.4).

3.1 Click Detection

We detect all click events from the audio recording. To provide a better understanding, we posted a video of a corresponding spectrogram of key insertion recording at http://bit.ly/2JciYB6. Prior to detecting clicks, we reduce the impact of low-frequency ambient noise, by subjecting it to a high-pass filter, to retain only frequencies above 15kHz that contains information about the clicks. Subsequently, we identify the starting point of each click, or its *onset*,

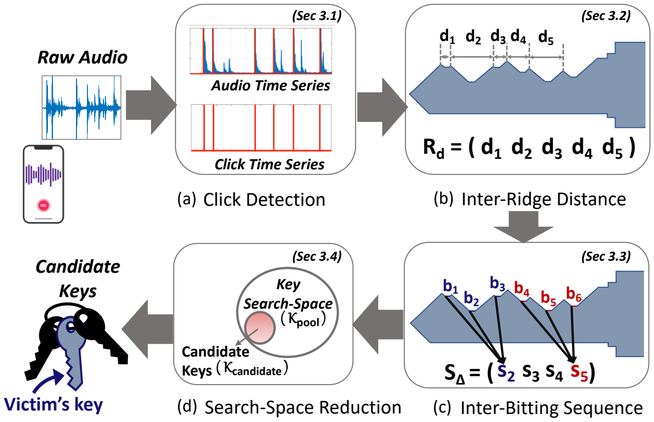

Figure 3: Figure depicts steps of *SpiKey* design to infer victim's key from audio recording of key insertion.

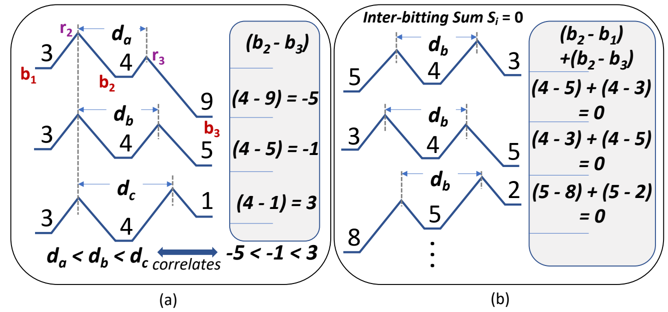

Figure 4: Figure depicts (a) the *correlation* between inter-ridge distances, d, and bitting depth differences $(b_i - b_{i+1})$ – e.g., as the value of $(b_2 - b_3)$ increases so does the distance between ridges, r_2 and r_3; (b) different bitting triplets with equal inter-ridge distance of d_b yield equal inter-bitting sum (which is 0 in this case).

in the pre-processed signal by applying change-point detection algorithm [12] on short time-windows around the computed peaks to account for their millisecond granularity. It finds the *least sum of standard deviations* across two regions that transition from low to high amplitude. We construct a click time series from the obtained click onsets (Figure 3(a)).

3.2 Inter-Ridge Distance Computation

We now take the click time series to infer the *inter-ridge distances* (Figure 3(b)). As a lock contains six pins, it adds additional challenges. For ease of explanation, we first present our approach for a simple but hypothetical *single-pin case* (i.e., a lock containing only one pin) and defer our explanation of the actual lock with all six pins (i.e., *multiple-pin case*) to Section 3.5 as our approach generalizes.

In a single-pin case, timestamps in the click time series correspond to interactions of a single pin (p_1) with all ridges of a key. Upon obtaining all the timestamps, t_1 to t_6, corresponding to clicks produced by ridges, r_1 to r_6, respectively, we compute a **sequence of inter-ridge distance**, $\mathbb{R}_d = (d_1, d_2, d_3, d_4, d_5)$ as $(t_2 - t_1, t_3 - t_2, t_4 - t_3, t_5 - t_4, t_6 - t_5) \cdot s_{key}$, where s_{key}, or speed of key insertion, can be computed from other parameters. We also defer the explanation of computing s_{key} in the multiple-pins cases to Section 3.5.

3.3 Inter-Bitting Sequence Computation

From a sequence of inter-ridge distances, $\mathbb{R}_d$, we *cannot* directly compute bitting depths as there is no direct correlation between the two. However, there exists a *correlation* between $\mathbb{R}_d$ and relative *differences* of adjacent bitting depths. We define and compute *inter-bitting sum* capturing bitting differences from $\mathbb{R}_d$, as a step towards identifying the candidates keys, i.e., keycode formed by multiple bitting depths.

3.3.1 Correlation between Inter-Ridge Distances and Bitting Depth Differences.

Recall from Section 2 that the formation of a ridge is due to inclines arising from its two adjacent bitting depths. In this regard, we observe that the precise location of the ridge is affected by *difference in depth* between these adjacent bitting positions, which in-turn affects inter-ridge distance. To see why, consider the bitting triplet, (b_1, b_2, b_3), and the corresponding ridges in-between, r_2 and r_3 (Figure 4(a)). Inter-ridge distance between r_2 and r_3 increases with increase in bitting depth difference, $(b_2 - b_3)$. Similarly, this distance also increases with increase in $(b_2 - b_1)$. Hence, in general, we observe that there exists a *correlation* between inter-ridge distance, d_i and the two bitting depth differences, $(b_i - b_{i-1})$ and $(b_i - b_{i+1})$.

Inter-bitting sum (s_i): For a bitting triplet (b_{i-1}, b_i, b_{i+1}), we define inter-bitting sum, s_i, as the **sum of bitting differences**, i.e., $s_i = (b_i - b_{i-1}) + (b_i - b_{i+1})$. For example, the triplet $(3, 4, 9)$ corresponds to an inter-bitting sum of $(4-3)+(4-9)$, which equals -4. Likewise, triplets $(3, 4, 5)$ and $(3, 4, 1)$ yield inter-bitting sums, 0 and 4 respectively. Values of inter-bitting sum are discrete, and constrained by the MACS, μ. For example, if $\mu = 7$ (i.e., $(b_i - b_{i-1})$ ranges from -7 to 7), inter-bitting sum is constrained to a total of 29 possible values from -14 to 14, where, a smaller s_i corresponds to a shorter inter-ridge distance, d_i. Many bitting triplets can correspond to the same inter-bitting sum (e.g., see Figure 4(b) depicting multiple triplets for $s_i = 0$).

However, as we do not know the bitting depth, we use the correlation of the inter-ridge distances, d_i, and the bitting differences to compute s_i, or sum of bitting differences. Specifically, s_i is related to d_i as: $s_i = (d_i - \alpha_p) \cdot (\frac{2 \cot(\theta/2)}{\alpha_d})$. *Due to the consistency in key-cutting parameters (bit spacing (α_p), cut angle (θ), and depth increment (α_d)),* we can compute inter-bitting sum, s_i, directly from the inter-ridge distance, d_i, that later caters to inferring candidate bitting depths.

3.3.2 Computing Inter-Bitting Sequence.

We compute a *sequence* of s_i values, from the inter-ridge distances, d_i, in $\mathbb{R}_d$. As an inter-bitting sum, s_i, constrains the value of its corresponding bitting triplet, a sequence of such sums constrains *all* triplets in a key, thereby significantly reducing the set of candidate keys. We define such a sequence as *inter-bitting sequence*, $\mathbb{S}_\Delta$. Figure 3(c) depicts $\mathbb{S}_\Delta = \{s_2, s_3, s_4, s_5\}$, where each s_i correlates with bitting triplets (b_1, b_2, b_3), (b_2, b_3, b_4), (b_3, b_4, b_5) and (b_4, b_5, b_6), respectively.

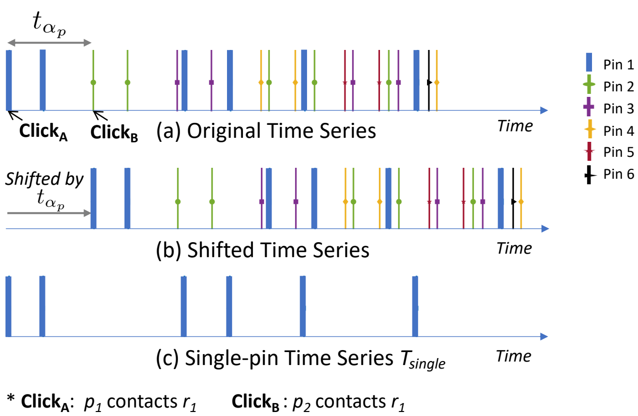

Figure 5: Figure depicts (a) a total of 21 clicks from six pins (e.g., p_1 with six ridges, ..., p_6 with one ridge); (b) a shift of the original time series by a constant time offset, t_{α_p}; and (c) subtracting (b) from (a) to reduce the problem to a simple single-pin case.

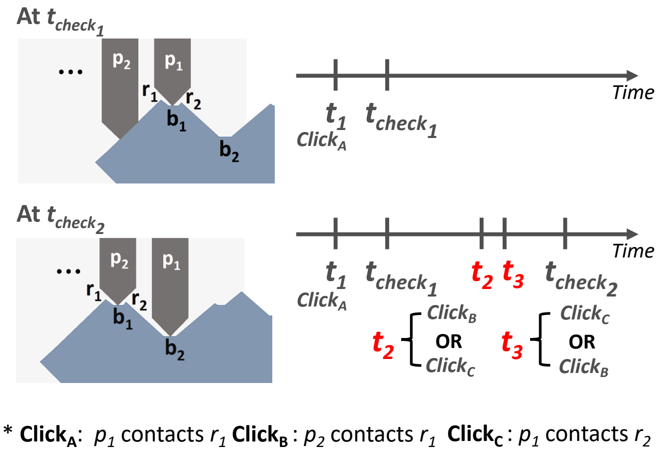

Figure 6: To identify the correct inter-pin time interval, t_{α_p}, we consider checkpoints t_{check_1} and t_{check_2}, to obtain timestamps of clicks. By t_{check_1}, $Click_A$ occurs (i.e., p_1 contacts r_1). By t_{check_2}, both $Click_B$ and $Click_C$ occurs (i.e., p_2 contacts r_1, and p_1 contacts r_2). However, the order in which these two clicks occur is unknown.

3.4 Key Search Space Reduction

Taking this inter-bitting sequence, $\mathbb{S}_\Delta$, we search for a subset of candidate keys (Figure 3(d)). As each inter-bitting sum, s_i, is a function of three adjacent bitting depths, by knowing the depths of first two bitting positions, we can deterministically obtain all other depths. For example, let $\mathbb{S}_\Delta = (4, -6, 4, 2)$, and let us choose $(b_1, b_2) = (4, 6)$. Then as $s_2 = (b_2 - b_1) + (b_2 - b_3)$, we obtain $b_3 = 2b_2 - b_1 - s_2 = 2 \cdot 6 - 4 - 4 = 4$. As we know, (b_2, b_3), we can use their values, along with s_3, to obtain b_4. By finding all remaining depths in this manner, we obtain the keycode 464886. However, we do not know the depths of the first two bitting positions. Hence, we compute all possible values for b_1 and b_2, and iteratively compute all remaining depths, based on $\mathbb{S}_\Delta$. We further discard all candidate keys that have invalid bitting depths (i.e., that do not satisfy the constraints of key specification such as MACS and bitting rules [20]), to finally yield a small subset of candidate keys.

3.5 Handling Multiple-Pin Case

Recall that aforementioned examples were for a simplified but hypothetical single-pin case (i.e., a lock which contains only one pin). We now explain how we solve the case for an actual 6-pin lock (i.e., multiple-pin case).

3.5.1 Translating To A Single-Pin Case. As ridges are not equally spaced in the key, clicks due to different pins may occur at slightly different times. This results in an array of clicks, as depicted in Figure 5(a). More specifically, there are a total of 21 clicks because p_1 comes in contact with all six ridges (r_1, ..., r_6) to yield six clicks, while p_2 comes in contact with first five ridges to yield five clicks, and so on. In essence, the click time-series in the multiple-pin case, is equivalent to several single-pin click time-series *interleaved*, where each pin yields a similar but temporally offset click time-series. This offset which arises due to distance between two adjacent pins, is equal to the time taken by a ridge to move between the two, which we refer to as *inter-pin time interval*, t_{α_p} (Figure 5(a)). To simplify the problem, we first create a shifted version of the original click time-series that occurs t_{α_p} after it (Figure 5(b)). All clicks (excluding clicks due to p_1) in the original time series, have clicks that coincide (i.e., occur at the same time) in the shifted time-series. On eliminating all coinciding clicks in the original time-series, we retain clicks only corresponding to p_1, and hence obtain a single-pin time series. We notate this retained time-series as T_{single} (Figure 5(c)).

3.5.2 Computing Inter-Pin Time Interval (t_{α_p}). To create the aforementioned shifted time-series as depicted in Figure 5(b), however, we need the inter-pin time interval, t_{α_p}. Time interval between first click of p_1 (with r_1) and first click of p_2 (also with r_1) equals t_{α_p}. We compute t_{α_p}, by obtaining the timestamps of both these clicks. Correspondingly, the click time-series yields a set of timestamps $\{t_1, t_2, \ldots, t_{21}\}$. In order to obtain the click timestamps, we consider two time checkpoints, t_{check_1} and t_{check_2}, which indicate the time at which p_1 and p_2 both rest on the first bitting position, b_1, respectively (Figure 6). By checkpoint t_{check_1}, the only completed click is $Click_A$ (first click of p_1) at t_1. By checkpoint t_{check_2}, the additional completed clicks are $Click_B$ (p_2 contacts r_1), and $Click_C$ (p_1 contacts r_2), although their *order of occurrence is unknown.* Owing to this uncertainty, the timestamp corresponding to $Click_B$ is either t_2 or t_3 (and vice versa for $Click_C$). Hence, resulting two candidates for t_{α_p} are $(t_2 - t_1)$ and $(t_3 - t_1)$. Subsequently, we obtain both their respective single-pin click time-series, T_{single}, and choose the one with six timestamps, to identify the correct t_{α_p} value.

3.5.3 Computing Speed of Key Insertion (s_{key}). Recall from Section 3.2 that we also need to compute the speed of key insertion, s_{key}, to compute the inter-ridge distance. We compute $s_{key} = \alpha_p / t_{\alpha_p}$, as we now know both of these values.

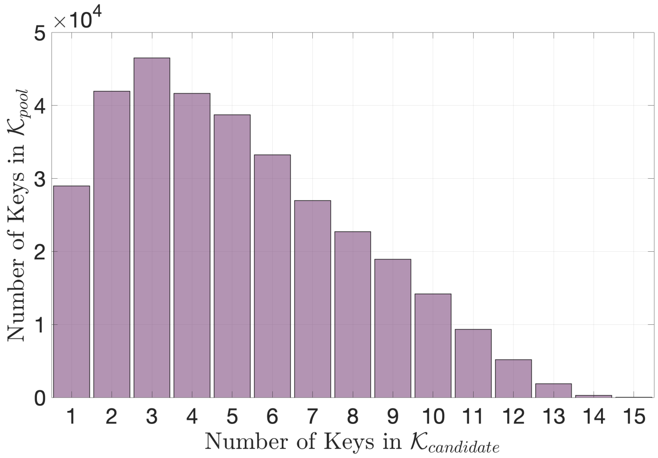

Figure 7: Histogram depicts number of elements in $\mathcal{K}_{candidate}$ obtained for all 330, 424 keys in $\mathcal{K}_{pool}$.

3.5.4 Overlap Filter. For some keys, upon translating the multiple-pin case to a single-pin case as described above, they may result in less than six timestamps in the translated T_{single}, rendering the aforementioned methods insufficient. This is because clicks of multiple pins coincide, or *overlap*, when distance between ridges happens to be a multiple of inter-pin spacing, α_p. To solve this problem, *SpiKey* implements an *Overlap Filter* after the *Click Detection* module, by checking if the total number of clicks equals to 21. *SpiKey* proceeds with the attack if the detected clicks pass this filter. We further discuss the implications of this filter in Section 5.

3.6 Handling Missing Ridges

Thus far, we utilize the clicks of ridges to reduce the search space in inferring the victim's key. However, there are a small proportion of keys, in which certain ridges are absent. This happens when the inclines arising from two adjacent bitting positions, converge beyond the key blade height (i.e., maximum height allowed within a key) and create a "*plateau*". In such cases, key insertion and withdrawal result in clicks at different ends of the *plateau* respectively (as clicks only occur when a key pin falls off an elevated position). We solve this problem by taking an *average of inter-ridge distances obtained in insertion and withdrawal cases.*

4 FEASIBILITY STUDY

We now present our feasibility study and its results.

4.1 Simulation Setup and Implementation

We perform our analysis on *Schlage 6-Pin C-keyway* [15]. We define $\mathcal{K}_{pool}$ as the set of all keys that are vulnerable to our attack, and $\mathcal{K}_{candidate}$ as the small subset of keys that is output by *SpiKey*, which guarantees to contain *the correct victim key.* $\mathcal{K}_{pool} = 330, 424$ keys as *SpiKey* filters overlaps (Section 3.5.4). For all keys in $\mathcal{K}_{pool}$, we model their real shape (i.e., identify bitting depths and ridges), based on key specifications, and obtain inter-ridge distances from $0.0310 - 0.2814$ inches. As *clicks* occur from real-world acoustic signals as depicted in Figure 3(a), we simulate such click time-series for all possible victim keys in the pool, and obtain their corresponding set of candidate keys, $\mathcal{K}_{candidate}$. We set the speed of key insertion/withdrawal, s_{key} to be 1 *inch/s* in all cases.

4.2 Preliminary Results

As a first step towards feasibility study, we evaluate *SpiKey* based on the number of elements in the set of *candidate keys*, $\mathcal{K}_{candidate}$, which are reduced from $\mathcal{K}_{pool}$. Figure 7 depicts a histogram of the number of elements in $\mathcal{K}_{candidate}$ for all keys in $\mathcal{K}_{pool}$. Given the click time-series of all 330, 424 keys as separate input to *SpiKey*, we are able to provide for each input a subset of candidate keys, where the number of elements range from $1 - 15$. This means that, on average, *SpiKey* is able to provide 5.10 candidate keys *guaranteeing inclusion of the correct victim key* from a total of 330, 424 keys, with 3 candidate keys being the most frequent case. This histogram demonstrates the impact of *SpiKey* as we further observe that *SpiKey* guarantees reducing more than 94% of keys (313, 780 keys) to less than 10 candidate keys.

5 DISCUSSION

We now present relevant discussion points of *SpiKey*.

Impact of *SpiKey*: We demonstrate the impact of *SpiKey* as it generalizes to different *types* (i.e., make and model) of keys as long as insertions yield clicks and the keys conform to particular specifications, even though we only analyzed on a single type, namely *Schlage 6-pin C-keyway*. Furthermore, we also demonstrate the impact of *SpiKey* despite reduced number of vulnerable keys. Recall from Section 3.5.4 that *SpiKey* only proceeds with the attack if multiple-pins do not create any *overlapped* clicks, thereby reducing the total number of vulnerable keys to 56.3% (330, 424 of 586, 584 keys), which makes more than half of all possible keys vulnerable.

Real-World Considerations: An attacker needs to consider the following to deploy *SpiKey*. First, we assume that the attacker has the knowledge of the type of lock and key by examining the exterior of the lock. Second, we assume that the speed does not vary from start to end of a key insertion (or withdrawal) in order to correctly infer the inter-ridge distances. This assumption may not always hold in real-world, hence, we plan to explore the possibility of combining information across multiple insertions.

Extending Attack Model: As another part of future work, we may extend the threat model to construct more powerful attacks. We may exploit other approaches of collecting click sounds such as installing malware on a victim's smartphone or smartwatch, or from door sensors that contain microphones [7, 21] to obtain a recording with higher signal-to-noise ratio. We may also exploit long distance microphones to reduce suspicion [1, 19]. Furthermore, we may increase the scalability of *SpiKey* by installing one microphone in an office corridor and collect recordings for multiple doors.

6 RELATED WORK

Various attacks on physical lock systems have been proposed in the past [4, 8, 13, 14]. A popular attack on pin tumbler locks is *lock picking*, where the bottom pins are raised up to the shear line using a pair of specialized tools, called *pick* and *tension wrench*, which are inserted into the keyway [17, 18]. Another subcategory of lock picking is *lock bumping*, which makes use of tools such as *bump key* and a hammer, to separate the top and bottom pins at the shear line, for a split second [23, 25]. *SpiKey* is inherently robust against many of the drawbacks of lock picking and bumping, because *SpiKey* only involves passively recording the sound of victim's key insertion. Hence, *SpiKey* enables a *layperson* to launch the attack without requiring any special expertise nor tools other than a smartphone, hence significantly *reducing suspicion*. *SpiKey* is inherently robust against anti-picking lock features [17, 22] that are equipped with many of the modern locks because *SpiKey* simply infers the key without exploiting the lock. Furthermore, upon a successful *SpiKey* attack, one can create or 3D print the key to grant him/herself multiple entries without leaving any traces of the attack [5]. Researchers recently proposed to infer keycode directly from an image of the key [10, 11, 13]. While an image-based attack can be stealthy [13], the attacker's success is dependent on factors such as image clarity and angle of view. However, *SpiKey* may complement image-based key-inference attacks, as we make use of victim's key insertion, an inevitable part of the unlocking mechanism.

7 CONCLUSION

We present *SpiKey*, a novel attack that infers the keycode or "secret" of a physical key by utilizing only a smartphone microphone to capture the time difference between inherent *click* sounds produced when the victim inserts the key into the lock. *SpiKey* inherently provides many advantages over lock picking attacks, including lowering attacker effort to enable a layperson to launch an attack without raising suspicion. We evaluate *SpiKey* with a proof-of-concept simulation, based on real-world acoustic data, and demonstrate that *SpiKey* can reduce the search space from a pool of more than 330 thousand keys to just *three* candidate keys for the most frequent case.

8 ACKNOWLEDGEMENTS

This research was partially supported by a grant from Singapore Ministry of Education Academic Research Fund Tier 1 (R-252-000-A26-133).

REFERENCES

[1] Ampflab. 2019. Microphone Long Range Audio Surveillance. http://ampflab.com.

[2] Matt Blaze. 2016. Notes on Picking Pin Tumbler Locks. https://www.mattblaze.org/papers/notes/picking/.

[3] Ryan Brown. 2019. Why criminals don't pick locks. https://www.art-of-lockpicking.com/criminals-dont-pick-locks/.

[4] Ben Burgess, Eric Wustrow, and J Alex Halderman. 2015. Replication prohibited: attacking restricted keyways with 3D-printing. In *9th USENIX Workshop on Offensive Technologies (WOOT'15)*.

[5] Datagram. 2009. Lockpicking Forensics. https://www.blackhat.com/presentations/bh-usa-09/DATAGRAM/BHUSA09-Datagram-LockpickForensics-PAPER.pdf.

[6] Adam Clark Estes. 2015. The History and Future of Locks and Keys. https://gizmodo.com/the-history-and-future-of-locks-and-keys-1735694812.

[7] Google. 2019. Nest Support. https://support.google.com/googlenest/answer/9250972?hl=en-CA.

[8] HITBSecConf. 2017. A Guide to Key Impressioning Attacks. https://conference.hitb.org/hitbsecconf2017ams/sessions/most-impressive-a-guide-to-key-impressioning-attacks/.

[9] IBISWorld. 2019. Door Lock & Lockset Manufacturing Industry in the US - Market Research Report. https://www.ibisworld.com/united-states/market-research-reports/door-lock-lockset-manufacturing-industry/.

[10] KeyMe. 2019. KeyMe Homepage. https://www.key.me.

[11] Keys4Classics. 2019. Keys Cut to Code. http://www.keys4classics.com/info/cut_notes.html.

[12] Marc Lavielle. 2005. Using penalized contrasts for the change-point problem. *Signal processing* 85, 8 (2005), 1501–1510.

[13] Benjamin Laxton, Kai Wang, and Stefan Savage. 2008. Reconsidering physical key secrecy: Teleduplication via optical decoding. In *Proceedings of the 15th ACM conference on computer and communications security*. ACM, 469–478.

[14] Anindya Maiti, Ryan Heard, Mohd Sabra, and Murtuza Jadliwala. 2018. Towards Inferring Mechanical Lock Combinations using Wrist-Wearables as a Side-Channel. In *Proceedings of the 11th ACM WiSec*.

[15] LSA Michigan. 2019. Schlage Bitting Specifications. https://www.lsamichigan.org/Tech/SCHLAGE_KeySpecs.pdf.

[16] CBS News. 2018. Yale Locks. https://www.cbsnews.com/news/almanac-yale-locks/.

[17] Deviant Ollam. 2008. Lockpicking and Physical Security. https://www.blackhat.com/presentations/bh-europe-08/Deviant_Ollam/Whitepaper/bh-eu-08-deviant_ollam-WP.pdf.

[18] Deviant Ollam. 2012. *Practical lock picking: a physical penetration tester's training guide*. Elsevier.

[19] Klover Products. 2019. Sound Shark. https://kloverproducts.com/sound-shark/.

[20] Graham Pulford. 2007. *High-security mechanical locks: an encyclopedic reference*. Butterworth-Heinemann.

[21] TechCrunch. 2018. LookOut SmartDoor Viewer. https://tcrn.ch/2PakE1C.

[22] M.W. Tobias. 2000. *LOCKS, SAFES, AND SECURITY: An International Police Reference Two Volumes*. Vol. 1. Charles C Thomas Publisher.

[23] M.W. Tobias. 2007. Opening Locks in Ten Seconds or Less. https://conference.hitb.org/hitbsecconf2007dubai/materials/D1%20-%20Marc%20Weber%20Tobias%20-%20The%20Insecurity%20of%20Mechanical%20Locks.pdf.

[24] Karim H. Vellani. 2019. *Strategic security management: a risk assessment guide for decision makers*. CRC Press.

[25] Barry Wels and Rop Gonggrijp. 2005. Bumping Locks. http://toool.nl/images/7/75/Bumping.pdf.