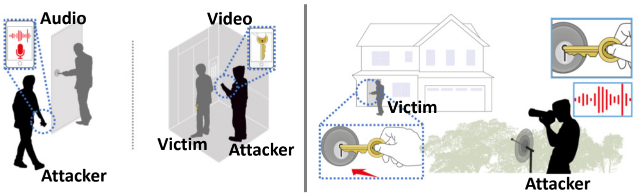
Acoustics to the Rescue: Physical Key Inference Attack Revisited

Soundarya Ramesh[1], Rui Xiao[1], Anindya Maiti[2], Jong Taek Lee[1], Harini Ramprasad[1], Ananda Kumar[1], Murtuza Jadliwala[3], and Jun Han[1]

[1]*National University of Singapore*, [2]*University of Oklahoma*, [3]*University of Texas at San Antonio*

Abstract

Lock picking and key bumping are the most common attacks on traditional pin tumbler door locks. However, these approaches require physical access to the lock throughout the attack, increasing suspicion and chances of the attacker getting caught. To overcome this challenge, we propose *Keynergy*, a *stealthy offline attack* that infers key *bittings* (or secret) by substantially extending and improving prior work that only utilizes a still image of the key. *Keynergy* effectively utilizes the inherent audible "clicks" due to a victim's key insertion, together with video footage of the victim holding the key, in order to infer the victim's key's bittings. We evaluate *Keynergy* via a proof-of-concept implementation and real-world experiments comprising of participants that perform multiple key insertions across a total of 75 keys with the related audio recorded using different microphone types placed at varying distances. We demonstrate that *Keynergy* achieves an average reduction rate of around 75% with an acoustics-based approach alone. When we combine both acoustics and video together, *Keynergy* obtains a reduced keyspace below *ten* keys for 8% of the keys (i.e., six keys out of 75 keys tested).



(a) Proximity Attacker

(b) Distant Attacker

Figure 1: Figure depicts potential attack scenarios of *Keynergy*. (a) depicts a *Proximity Attacker* capturing via the attacker's smartphone microphone and camera to capture the sound of key insertion and video of victim holding the key, respectively. (b) depicts a *Distant Attacker* employing a directional *parabolic* microphone and a *telephotography* camera.

1 Introduction

Pin tumbler locks constitute a majority of the market share in securing home and office doors, with a few manufacturers dominating the global market [23, 31, 45, 64]. Consequently, they have been a constant target of several known hobbyist-style attacks and academic proposals that have attempted to compromise their security. *Lock picking* and *key bumping* are the most common existing attacks, which are non-destructive techniques that manipulate a lock's internal components (known as *pins*) by inserting specialized instruments in order to unlock it without the possession of a valid key [17, 19, 70]. However, these techniques inherently have significant limitations. First, they require physical access to the lock throughout the attack, which raises suspicion and increases the chances of the attacker getting caught, especially with the prevalence of motion sensor enabled home security cameras [6, 7]. Furthermore, recent locks now ship with anti-picking and/or bumping features, rendering such attacks difficult, especially for laypersons. Besides, these attacks require sufficient amount of training while only granting one-time entry despite a successful attack [48, 49, 63, 70].

To overcome these limitations, one research effort in the literature proposed a stealthy offline attack that utilizes a still image of the victim's key (that the attacker captures via telephotography) to infer its bittings (or secret) [35]. While a novel attempt at a stealthy attack, it requires high-resolution images of immobile keys at a particular angle (e.g., lying flat on a surface), which is a significant restriction that renders it less practical. This assumption is not surprising because any movement while imaging would blur the bittings. In fact, the authors of this work agree that a more practical attack scenario, where an attacker captures video footage of a moving key (e.g., when a victim is holding a key), would cause a serious degradation to their attack accuracy [35].

The above phenomenon, also observed by us during our experiments, leads us to the following question: *Is it possible to design a realistic and stealthy offline physical key inference attack that overcomes these shortcomings and impractical assumptions of prior work?* In search for an answer to this question, we design *Keynergy* [1], a novel offline attack for inferring the bittings of a key that employs a combined acoustics-video side-channel. More specifically, *Keynergy* substantially extends and improves the prior image-only attack by utilizing inherent sounds of key insertions as the victim inserts her/his key into the lock and video footage capturing the victim holding the key. The attacker may obtain the sound and video recording separately, each from a variety of sources and later consolidate them together. Figure 1 depicts two exemplary scenarios for obtaining the acoustic and video signals to carry out the *Keynergy* attack, where the attacker: (a) records with her/his smartphone microphone while walking by the door. At a later time, the attacker records the video of the victim holding the key (e.g., in an elevator or a hallway). (b) employs a directional (parabolic) microphone and a telephotography camera from several meters away.

We design *Keynergy* by first extending the prior work [35] by leveraging blurred and distorted images (caused by significant key movements during recording) of the target key at different angles to obtain a plausible yet relatively large set of bitting values, thus reducing the overall key search space. We then utilize the audio signal of key insertion to further significantly reduce the keyspace to a small subset of keys. However, such further reduction is extremely challenging due to the following two reasons. First, the remaining keys in the initially reduced subset from video footages are likely to exhibit similar bitting patterns, making further reduction immensely challenging. Second, to exacerbate the problem, *Keynergy* needs to subsequently rely only on the sound signal to infer exceedingly fine-grained bitting depths that differ by sub-millimeters (i.e., 0.381 mm).

To solve the aforementioned challenges, *Keynergy* utilizes the audible "clicks" that occur as the lock's *pins* fall off the key's *ridges* (that exist due to cuts of the key's bittings) during insertion, to create a *click pattern* unique to the key. *Keynergy* then compares the obtained *click pattern* against *simulated patterns* (of resulting "clicks") of all possible keys that have been pre-computed (by the attacker) via simulation modeled after a *constant insertion speed* by utilizing the techniques from prior work [55]. However, the *unknown* and *inconsistent speed* of key insertion renders this comparison significantly difficult. We overcome this challenge by fusing across recordings from multiple key insertions of the same victim. Ultimately, *Keynergy* outputs a small subset of the most likely keys that resemble the victim's key.

Inferring the secret key bittings in this fashion would ultimately allow the attacker to replicate the corresponding key(s), for example, using 3D printing, in order to unlock the victim's door. *Keynergy*, by design, yields many advantages over the state-of-the-art attacks to compromise pin tumbler locks. For instance, *Keynergy* minimizes the attacker's physical access to the lock, thus reducing the risks of him/her being apprehended. It is also robust against locks with anti-picking and/or bumping features as *Keynergy* attacks the key itself and not the lock. Furthermore, *Keynergy* would enable those inexperienced in lock picking to launch this attack, granting them multiple unrestricted accesses to the victim's property.

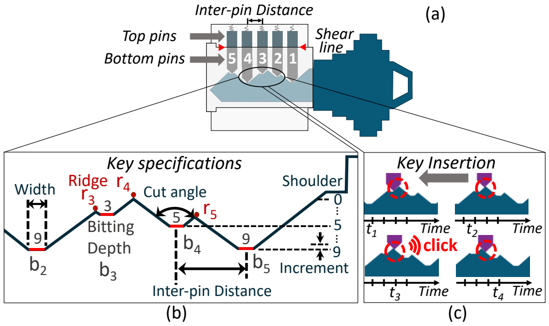


Figure 2: (a) depicts a pin tumbler lock consisting of *pins* $(p_1, \ldots, p_5)$. When the matching key is inserted, the pins exactly separate at the *shear line* to unlock; (b) depicts key *specification* parameters and bittings/ridges within a key. (c) depicts the origin of sound from the interaction between *bottom pins* and key *ridges* during key insertion. The time points $t_1 - t_4$ indicate instances as the relative position of the pin changes with respect to the ridge. "Click" occurs at time t_3.

We evaluate *Keynergy* by means of a proof-of-concept implementation and real-world experiments by recruiting participants that insert 75 different keys for a total of more than 3,600 insertions. The resulting key insertion audio is recorded with multiple microphone types placed at varying distances from the lock. From our empirical analysis, *Keynergy* achieves an average reduction of around 75% with the acoustics-based approach alone. When we combine both acoustics and video together, *Keynergy* obtains a reduced keyspace below *ten* keys for 8% of the keys (i.e., six keys out of 75 keys tested).

By means of this work, we hint at a new avenue of sensor side-channel attacks that combine information from different sensing modalities – such as microphone and camera in our case – abundantly available in today's era of Internet-of-Things and Cyber-Physical Systems. An individual modality may not provide sufficient information, but they could constructively complement each other to enable new attacks that easily surpass the well-studied risks from just the individual modalities. We hope that this paper would encourage the security community to explore new defense policies to thwart such potentially emerging attacks.

2 Primer on Pin Tumber Locks and Keys

Prior to presenting our attack design, we provide background on the construction of pin tumbler locks and keys. We also explain the cause of the sound produced during key insertion, resulting in a *click pattern*, which forms the basis of our attack.

[1] *Key* inference from the *synergy* between two sensing modalities

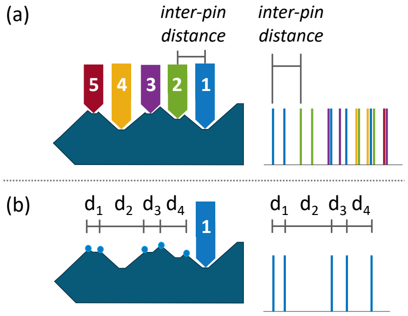


Figure 3: (a) depicts the *click pattern* from multiple pins during key insertion; (b) depicts the simulated *click pattern* from a single pin, p_1. Time-interval between adjacent clicks equals inter-ridge distance for a constant speed of insertion.

Pin Tumbler Lock. Pin tumbler locks, such as Schlage SC1, typically consist of five pin-pairs, shown as $p_1, \ldots, p_5$ in Figure 2(a). Each pair comprises the *top* and *bottom pins*, where adjacent pins are separated by an *inter-pin distance*. The top (or driver) pins are spring-loaded, and in their resting position, they block the rotation of the lock's plug. Bottom (or key) pins vary in length, corresponding to its key's *bittings*, or cut depths (i.e., secret). When a matching key is inserted, the bottom pins correctly sit on each of the key bittings, causing the pins to align on a *shear line*, thereby allowing the plug to rotate and unlocking the lock.

Key. The key of a pin tumbler lock has a unique *keycode*, denoted by a 5-digit number $b_5 \ldots b_1$ (e.g., *39359*). The keycode specifies the bitting depths, which are cut by adhering to the manufacturer's *specifications* [63]. The specifications mandate key parameters including, *number of bittings*, *depth values*, *increments*, *bitting width* and *cut angle*. We further explain their details in Appendix A. For Schlage SC1 keys, there are a total of *five* bitting positions and *ten depth values* (denoted by numbers 0-9), with each adjacent depth value separated by an *increment* of 15 milli-inch (0.381 mm) (see Figure 2(b)). Hence, the maximum number of keys possible is 10^5. However, in practice, the keyspace is close to 75% of the maximum, due to constraints imposed by the manufacturer's specifications. Among the many constraints, *Maximum Adjacent Cut Specification (MACS)* is an important constraint that bounds the difference between adjacent depths. For example, with $MACS = 7$ (also the case for Schlage SC1 keys), the difference between two adjacent bittings can be at most 7. Hence, a key with depths 08345 is not possible, as 0 and 8 yield a difference greater than 7. On applying all the constraints, the keyspace for Schlage SC1 keys reduces to 75,066, i.e., 75% of the theoretically maximum possible number of keys. We further enumerate all other constraints in Appendix B.

Key Insertion Sound and Click Pattern Formation. A *ridge* on a key's blade (e.g., r_2, r_3, r_4) is the convergence of inclines from two adjacent bitting positions, as illustrated in Figure 2(b). Thus, for a 5-bittings key, there will be a total of *five* ridges. Ridges are an important key feature relevant to our *Keynergy* attack, as they are the primary source of sound produced during key insertion. Specifically, as depicted in Figure 2(c), during a key insertion, the bottom pins fall off the ridges resulting in a sharp "click" sound. Moreover, multiple ridges and pins result in a series of "clicks" producing a *click pattern* as illustrated in Figure 3(a). We post a spectrogram of exemplary key insertion sound in the following: `https://bit.ly/3pr5aFS`. *Keynergy* exploits the unique nature of the *click pattern* for each key to ultimately identify the correct victim key from a keyspace of all candidate keys. We perform a feasibility study (Appendix C) to verify the occurrence of *click patterns* in human key insertion audio.

3 Threat Model

We now outline the attacker's goals and capabilities, and further enumerate *Keynergy*'s assumptions.

Goals and Capabilities. The *goal* of the attacker is to launch a key inference attack by utilizing the *sound* as the victim is inserting the key, and *video* recording of the victim holding the key. The attacker launches a *stealthy offline attack* to infer the victim's key *bittings*, such that s/he can replicate the physical key with that information. To achieve this goal, the attacker may launch two different types of attacks, namely *proximity* or *distant* attacks, as shown in Figure 1.

When launching the *proximity attack* (Figure 1(a)), we assume that the attacker is able to secretly capture sound and video recordings in close vicinity of the victim by means of appropriate recording devices. For example, the attacker may walk by the victim and record the sound of victim's key insertion with her/his smartphone microphone. Alternatively, the attacker could record such sound by concealing small "spy" microphones within objects that are typically placed near the door, e.g., gardening pots or shoe racks. Similarly, the attacker may also secretly record a video of the victim holding the key by means of a smartphone camera when in close proximity to the victim, for example, in an elevator.

When launching the *distant attack* (Figure 1(b)), the attacker has the capability to gain access to the sound and video recordings of the victim's key insertions from a distance away (e.g., by hiding in the bushes or inside a parked car by utilizing a parabolic microphone, telephotography camera, and/or even a drone flying nearby. These devices are capable of capturing sound and video signals from a far away distance [35, 46, 56].

Assumptions. We assume that the attacker knows the location of the victim's door, as well as the make-and-model of the lock (visually apparent from the lock). We also assume that the attacker has the corresponding key specifications (publicly available) [9, 47]. Moreover, upon successfully deriving a small set of candidate keys, the attacker can replicate them by leveraging key code cutting machine [22] or a 3D printer [20, 34, 52]. Then the attacker needs a short physical access to the door to try the replicated keys to determine the actual key.

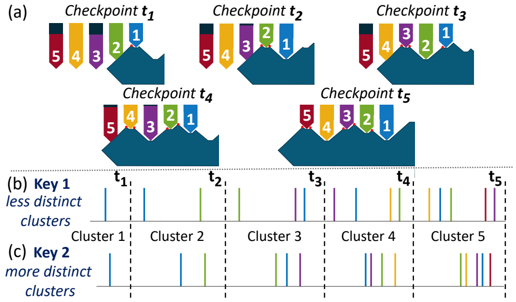


Figure 4: Figure depicts the key at *checkpoints* ($t_1,\dots,t_5$), i.e., timestamps at which pin p_1 rests at bittings $b_1,\dots,b_5$. Clicks occurring between two checkpoints are *clusters*. It also depicts two distinct keys with differing cluster distinctness.

4 Modeling Key Insertion Sound

When launching the *Keynergy* attack, the attacker compares the obtained *click pattern* from the victim's key insertion sound with the simulated click patterns (also referred as *simulated patterns*) that models key insertions based on key specifications and by assuming a *constant insertion speed*. In order to generate such *simulated patterns*, we utilize techniques from the prior work, *SpiKey* [55]. When obtaining *simulated patterns*, we observe the formation of click clusters from intermittent pauses, which we later utilize to overcome the challenges of variable insertion speeds in designing *Keynergy*.

4.1 Simulated Patterns

We model the *simulated patterns* by obtaining the *click pattern* of the first pin (based on the insertion sound as presented in Section 2) for a constant insertion speed of 1 inch/sec. Consequently, the time interval between clicks produced from the first pin case equals the *inter-ridge distance* (i.e., the horizontal distance between adjacent ridges; see Figure 3(b)). Hence, the problem of modeling the *simulated pattern* reduces to computing the *inter-ridge distance*.

To compute the *inter-ridge distance*, we obtain the position of every ridge on the key by taking advantage of the key's geometry and specifications (see Appendix A). Upon obtaining the first pin *click pattern*, we create the entire *simulated pattern* by repeating this *click pattern* with an offset of the *inter-pin distance* as shown in Figure 3(a).

4.2 Formation of Click Clusters

The obtained *simulated patterns* exhibit a pattern we define as *clusters*, which are click groups formed due to intermittent pauses that occur during the key insertion. As depicted in Figure 4(a), pin p_1 transiently rests on bitting positions b_1 through b_4, and ultimately on b_5 at the end of the insertion. We refer to each of the times when p_1 rests on the bitting positions $b_1,\dots,b_5$ as *checkpoints*, which we notate as $t_1,\dots,t_5$, as depicted in Figures 4(b) and (c) for *Key 1* and *Key 2*, respectively. Hence, all clicks that occur between two checkpoints belong to a single *cluster* (e.g., in the case of a 5-pin lock, the *simulated patterns* has exactly *five* clusters). Furthermore, the number of clicks increases by one in each succeeding cluster, i.e., the first cluster has one click, the second cluster has two clicks, and so on, yielding a total of 15 clicks. However, the number of *distinct* clicks within a cluster can be less than the maximum in the presence of overlapping clicks, i.e., due to simultaneous occurrence of multiple clicks. For example, a key with keycode 33333 (i.e., all identical bittings) will have only nine clicks in its *simulated patterns* (see Appendix D).

Cluster detection enables a more guided search for clicks due to the presence of an upper bound on the number of clicks per cluster (see Section 5). As *clusters* are a localized time region in the key insertion, the speed variations within each cluster tends to be lower in comparison to the entire insertion, leading to more reliable *click pattern* matching.

Presence of *distinct* clusters is crucial for the success of our attack. We observe that keys have varied cluster distinctness. Hence, we can categorize keys with more distinct clusters to be the *key type* that is more susceptible to *Keynergy* attack. For example, Key 2 exhibits more distinct clusters than Key 1 in Figures 4(b) and (c). From our analysis, we observe that 79% of all Schlage SC1 keys (i.e., 59,207 keys) have distinct clusters, constituting the more vulnerable key types. We provide more details of our analysis in Appendix F.

5 Attack Design and Implementation

We now first present an overview of *Keynergy*'s design, and then delve into the details of its main modules.

5.1 Attack Design Overview

Keynergy utilizes the *audio recordings* of multiple key insertions (e.g., n insertions over a period of time), computes the *time-interval between clicks* in each recording, and ultimately converts them into a single *click pattern* (i.e., a set of time-intervals) to compare it against the modeled *simulated patterns* of all keys in the keyspace. We present the individual modules that constitute the design of *Keynergy* in Figure 5.

First, the *Click and Cluster Detection* module identifies the presence of *clusters* and the corresponding *clicks*. Subsequently, *Synthesized Click Pattern Computation* module selects, for each cluster, the most representative insertion (out of n insertions), and stitches the selections of each cluster together to obtain a new *synthesized pattern*, which we compare against the *simulated patterns*. Prior to this comparison, we design the *Video Analysis* module, that utilizes the *video recordings* of the victim holding the key to reduce the keyspace to a smaller subset. Finally, the *Pattern Comparison* module takes as inputs the *synthesized pattern* as well as the *simulated patterns* of the reduced keyspace, to perform the

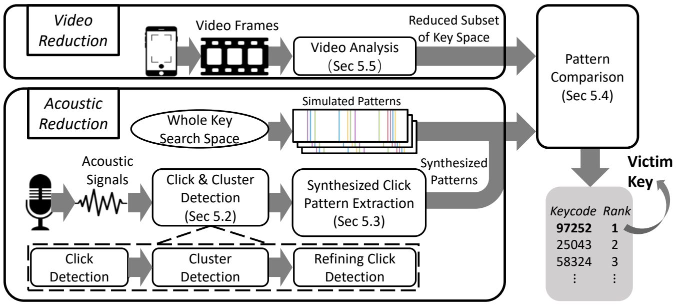


Figure 5: Figure depicts *Keynergy*'s design. For audio-based reduction, we utilize multiple key insertion recordings, and *synthesize* a representative *click pattern*, to compare against *simulated patterns* of keys to obtain a key rank-list. We obtain the subset of keys that we utilize for this comparison from video-based approach. The final predicted key-rank of the victim's key is likely to be among the top ranks.

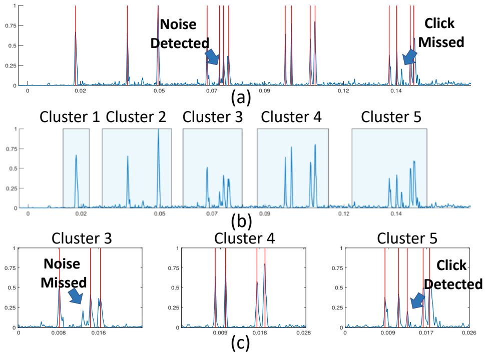


Figure 6: Figure depicts (a) detection of 15 clicks on the weighted spectral flux representation. (b) depicts the detection of clusters. (c) depicts how *Keynergy refines* the click detection results with the help of cluster boundaries.

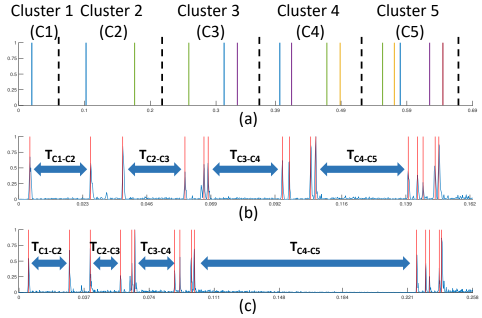


Figure 7: (a) Figure depicts the *simulated patterns* along with their clusters, $\{C_1,\dots,C_5\}$. (b) and (c) depict two instances of human key insertion of the same key, where T_{Ci-Cj} represents the time-interval between clusters, Ci and Cj. Clicks within a cluster occur more closely than clicks of different clusters. Also, the time-interval between clusters is inconsistent across different insertions. However, clicks *within* a cluster exhibit less variance in *click patterns* even across different insertions.

comparison and output a rank-list of keys, where a higher-ranked key corresponds to the victim key.

5.2 Click and Cluster Detection

Click and Cluster Detection module is comprised of the following three *sub-modules*. First, the recordings of each insertion are input to the *Click Detection* sub-module to determine all potential clicks. It takes as input the audio recordings (for n insertions) to determine the *timing information* of all 15 clicks for each insertion, which is the maximum number of clicks in a 5-pin lock. The detected click timestamps are then input to the *Cluster Detection* sub-module to identify the five *clusters* present in each insertion. Subsequently, we utilize *Refining Click Detection* sub-module to fine-tune the click detection within each cluster as there may be incorrect clicks initially detected due to the low signal-to-noise ratio (SNR).

Click Detection. We identify timing information of clicks by detecting their *onsets*, or the instant that marks the beginning of clicks' energy increase [14]. However, as it is difficult to extract the onsets directly from the audio signal, we transform the audio signal to *weighted spectral flux* representation [14], where click onsets appear as amplitude peaks, which can then be identified using peak detection approaches. Weighted spectral flux, WSF, captures the increase in energy by comparing energies of adjacent time windows. More specifically, in order to compute WSF from the audio, we partition it into T overlapping frames, $\{F_1,\dots,F_T\}$, each with fixed time-interval ($\sim$ 0.7 ms), and obtain their *magnitude spectrum* $\{M_1,\dots,M_T\}$, which represents the energies at different frequencies computed as the absolute value of their discrete Fourier transform (DFT). We compute $WSF(t)$ as the *increase in energy* of the current frame, F_t, in comparison to an average of previous k frames (denoted by AM_{t-1}), weighted by their frequencies as: $WSF(t) = \sum_{f=f_0}^{f_1}\Big[f \times \mathcal{H}\big(M_t(f) - AM_{t-1}(f)\big)\Big]$ where $\mathcal{H}(x) = (x+|x|)/2$, returns non-zero values only for energy increases as they contribute towards identification of click onsets. Also, f_0 and f_1 indicate the frequency bins corresponding to minimum and maximum frequencies of interest. We consider frequencies above $15kHz$, as higher frequencies capture quick transitions in energy, which is important to determine precise timing of clicks [59].

Subsequently, we set a minimum distance between clicks in order to prevent choosing peaks in the noise floor and retain peaks that are above a threshold (i.e., fraction of the maximum amplitude). Finally, we select the largest 15 peaks to be the resulting clicks of the key insertion as depicted in Figure 6(a). We repeat this process across all n insertion recordings.

Cluster Detection. Taking as input all 15 click onsets, *Cluster Detection* sub-module outputs *five clusters* for each insertion.

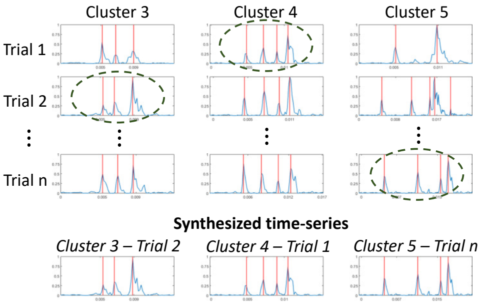


Figure 8: Figure depicts the clusters across different trials to ultimately obtain *synthesized pattern*.

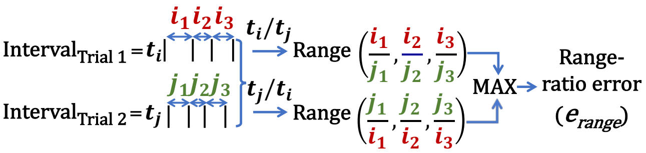

Interval$_{\text{Trial 1}}$ = t_i
$i_1 i_2 i_3$
t_i/t_j
Range $\left(\frac{i_1}{j_1}, \frac{i_2}{j_2}, \frac{i_3}{j_3}\right)$
Interval$_{\text{Trial 2}}$ = t_j
$j_1 j_2 j_3$
t_j/t_i
Range $\left(\frac{j_1}{i_1}, \frac{j_2}{i_2}, \frac{j_3}{i_3}\right)$
MAX
Range-ratio error
(e_{range})

Figure 9: Figure depicts the pairwise error computation between two trials, t_i and t_j, with corresponding time-intervals, $\{i_1, i_2, i_3\}$ and $\{j_1, j_2, j_3\}$. We compute the range of the two possible interval ratios (i.e., $Range(i, j)$ and $Range(j,i)$), the maximum of which constitutes the *range-ratio error* (e_{range}).

To obtain the five clusters, we leverage the observation that there is a relatively *long* pause between clusters, resulting in longer time-intervals between adjacent clicks that belong to neighboring clusters which are due to human insertion as well as presence of distinct clusters in the key as shown in Figure 7. Hence, we choose the four largest time-intervals as shown in Figure 6(b). For further analysis, we only leverage clicks from *Cluster* 3 onwards, as the first two clusters have too few clicks for pattern comparison. As every cluster is a localised time-region within the key insertion, we observe lower speed variations in each cluster as opposed to the entire insertion (Figures 7(b) and (c)), hence resembling the *simulated patterns* (Figures 7(a)) which is modeled based on constant insertion speed.

Refining Click Detection. Upon obtaining the clusters, we *refine* the click detection within each cluster for all n insertions, because the *Click Detection* sub-module may be inaccurate due to the following reasons: (1) clicks may be closely spaced, or even occur simultaneously (i.e., overlapped), hence producing peaks that are difficult to discern; (2) clicks may exhibit low energy, thereby leading to reduced amplitude of peaks; and (3) presence of noise, which may result in erroneous peaks being detected as clicks. We overcome these challenges by refining click detection with the help of clusters. Specifically, we make use of the upper bound on the number of clicks for each cluster, i.e., Cluster p has at most p clicks, hence preventing more than p clicks to be chosen within that cluster, while also aiding the selection of closely-spaced clicks when less that p clicks are initially chosen, e.g., we observe that in Figure 6(c), *Refining Click Detection* sub-module omits the noisy peak in Cluster 3, while it identifies a low amplitude click in Cluster 5, unlike the *Click Detection* sub-module that marks noise as click and vice versa (Figure 6(a)).

5.3 Synthesized Click Pattern Extraction

Despite the refined clicks from the previous module, correctly extracting all clicks within each insertion may still be error-prone due to the aforementioned sources of noise. *Synthesized Click Pattern Extraction* module solves this challenge by fusing information across multiple insertions (or *trials*) as it is unlikely for similar noise pattern to reoccur across different insertions. This module takes as input n trials and chooses one trial per cluster as a representative to ultimately synthesize a new *click pattern*, which we refer to as *synthesized pattern*, that most likely resembles an insertion with minimal noise. Specifically, this module chooses one representative trial per cluster (or $trial - cluster\ pair$), and merges across all three clusters to output the *synthesized pattern*. Figure 8 illustrates a set of trials, where we select the following $trial - cluster\ pairs$ to construct the final *synthesized pattern*: *Trials 2*, *1*, and n, for *Clusters 3*, *4*, and *5*, respectively.

To select the most representative $trial - cluster\ pair$ across all trials, we employ a two-stage approach. First, for each cluster, we only retain trials that contain the *mode* (or the most frequently occurring) of the number of clicks and discard the rest (as they may be more prone to missing clicks or having additional noisy clicks), e.g., in Cluster 4 of Figure 8, *Trials 1*, *2*, and n contain the maximum of *four* clicks. Second, we select a representative trial out of the retained trials. For this, we compute a pairwise error between all combinations of retained trials, to ultimately output the trial with the least error as the representative $trial - cluster\ pair$, (e.g., *Trial 1* for *Cluster 4*). For each pair of trials, we compare the corresponding time-intervals across each of the adjacent clicks within a cluster, e.g., in Figure 9, when comparing *Trial 1* with *Trial 2* for *Cluster 4*, we first compute the intervals of the two trials such that $Interval_{Trial1} = \{i_1, i_2, i_3\}$ and $Interval_{Trial2} = \{j_1, j_2, j_3\}$. Subsequently, we compute the ratio of corresponding time intervals, followed by its *range*, or the difference between the maximum and minimum ratios (i.e., $Range(Interval_{Trial1}, Interval_{Trial2}) = Range(i, j) = Max[\frac{i_1}{j_1}, \frac{i_2}{j_2}, \frac{i_3}{j_3}] - Min[\frac{i_1}{j_1}, \frac{i_2}{j_2}, \frac{i_3}{j_3}]$). In order to keep the error value consistent for different ordering of trials, we compute the maximum of $Range(i, j)$ and $Range(i, j)$, which we refer to as the *range-ratio error* (e_{range}). We leverage this ratio to compare click interval patterns between any two trials, without being affected by their different insertion speeds. Finally, we choose a representative trial which has the least sum of pairwise error with majority of the trials.

5.4 Pattern Comparison

Pattern Comparison module takes as input - *synthesized pattern*, *simulated patterns* of the *reduced keyspace* from *Video Analysis* module (see Section 5.5) to output a rank-list of keys, with a higher-ranked key being more likely to be the victim's key. This module compares the *synthesized pattern* against all of the *simulated patterns* within the *reduced keyspace*, specifically by comparing each of the clusters (i.e., 3, 4, and 5) separately, and then aggregating the comparison results across all clusters. We choose such because the clicks within a cluster exhibit low variations in speed as opposed to clicks across the entire insertion, thereby exhibiting closer resemblance to the *simulated patterns*, which is modeled based on constant insertion speed (see Figure 7). However, this comparison still poses some challenges due to remaining variability in speed and occasional click misses within clusters. To overcome this challenge, we compute two error functions to quantify their dissimilarity, namely, *pattern comparison* and *click detection errors* (or $e_{pattern}$ and e_{click}, respectively). Utilizing the error functions, this module ultimately outputs ranks of all keys.

Specifically, $e_{pattern}$ error computes *range-ratio error* (similar to Figure 9) to quantify the dissimilarity between *simulated patterns* of all keys and the *synthesized pattern*. Hence, keys with *simulated patterns* that exhibit similar patterns to *synthesized pattern* would be assigned lower $e_{pattern}$ values. However, there may be cases where *synthesized pattern* has missing clicks (e.g., when the clicks occur close together) rendering $e_{pattern}$ alone insufficient for ranking keys. Hence, upon a likely detection of missing clicks from the $e_{pattern}$ computation, we assign e_{click} as the largest click-interval adjacent to the potentially missed click(s). After assigning $e_{pattern}$ and e_{click} for all clusters of keys in the reduced keyspace, we sum up the two errors across the clusters and list the keys from lowest to highest error to obtain an aggregated rank-list.

5.5 Video Analysis

We now combine information from video footages in order to achieve additional keyspace reduction. We first re-implement *Sneakey* [35] which performs image-based key-inference and extend it further to work with video footages capturing blurry key images due to the mobile key at unfixed angles. *Sneakey*'s implementation normalizes the key image by manually annotating *eight keypoint* locations (five and three from the key's head and blade, respectively) by the attacker, and transforms it to the respective *keypoints* on a *reference key* (i.e., another key of the same make-and-model that is known to the attacker). We extend this design to utilize only *four* keypoints (three and one on the key's blade and tip, respectively), to account for a more realistic attack scenario where the head of the key may be occluded as the victim is holding the key. Subsequently, we identify the five bitting locations and depths on the normalized image to yield the most likely *bittings*. Prior to applying the image-based inference, we choose the top three frames that exhibit least *blurriness* from the video recording by applying a variance of *Laplacian* operator, which measures the amount of edges present in images, to utilize it for a blur detection [51, 53]. Ultimately, this module outputs a reduced key search space to be input to *Pattern Comparison* module for further reduction.

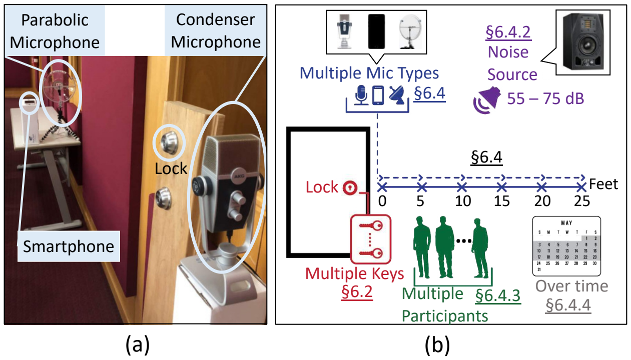


Figure 10: Figure depicts (a) the experimental setup with a custom-made door with Mic_{cond}, Mic_{parab}, and Mic_{phone}; (b) set of all varying experimental conditions.

6 Evaluation

We present the evaluation of *Keynergy* through comprehensive real-world experiments, demonstrating its feasibility.

6.1 Experimental Setup

Apparatus. Figure 10 illustrates our experiment setup, where we use a custom door setup with Schlage SC1 5-pin lock. This setup follows the standard door width of 45 mm with the lock installed at the conventional height of 42 inches above the ground [60]. There are a total of 59,207 vulnerable keys for the Schlage SC1 lock (which constitutes 79% of the original keyspace due to *distinct cluster-based filtering* as presented in Section 4). We use the following three different types of microphones with corresponding sampling rates (F_s):

- Mic_{cond}: AKG Lyra condenser mic ($F_s = 192kHz$) [1]
- Mic_{parab}: SoundShark Parabolic Collector with Countryman B3 Lavalier mic ($F_s = 192kHz$) interfaced with Behringer UMC202HD audio interface [2, 4, 8]
- Mic_{phone}: Google Pixel ($F_s = 44.1kHz$) [5]

In addition, we use *Adam Audio A3X studio monitor speaker* [11] with a flat frequency response from 60 Hz up to 50kHz for an accurate reproduction of different noise sources in Section 6.4. To evaluate the different attack scenarios motivated in Sections 1 and 3, we perform experiments by varying the position of the Mic_{parab} and the Mic_{phone} from

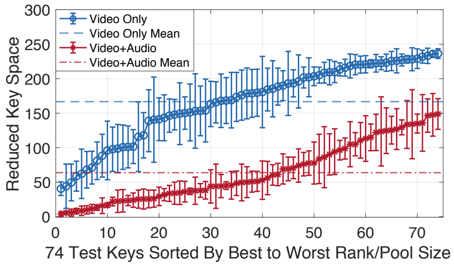


Figure 11: Figure depicts the overall performance comparing *Video-only* pool-size vs. *Keynergy* (*Video + Acoustics*) rank.

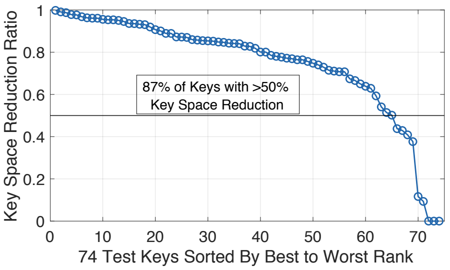


Figure 12: Performance of *Acoustics-only* approach.

5ft up to 25ft away from the door setup. We also perform our experiments in different locations including university's lecture hall, multipurpose room and a dormitory room.

We conduct the experiments with a total of 78 keys (where 75 and three keys are each used for test and train, respectively), with $10-12$ trials of insertions for each instance of experiments. We collect more than 3,600 insertions by recruiting a total of 13 participants over a span of three months. We conduct the experiments by adhering to our university's Institutional Review Board (IRB). We present the specific data collection methods accordingly in the subsequent subsections.

Performance Metrics. We define and utilize three metrics in order to measure *Keynergy*'s attack performance.

- *Key Rank* ($Rank_{key}$): Rank of each key in keyspace from *Keynergy*'s attack; A higher ranked key (e.g., *Rank 1*) is more likely to match the victim's key.
- *Keyspace Reduction Ratio* ($Ratio_{Reduction}$): Fraction of keys in the keyspace that yield lower ranks than the victim key's rank (e.g., if victim's key is predicted as *Rank 10*, then $Ratio_{Reduction} = \frac{59{,}207-10}{59{,}207} \approx 0.999$.
- *Search Pool Size* ($Pool_{search}$): Reduced key search space from video analysis (see Section 6.3.3). Keys within the pool are equally likely to be the victim key.

6.2 Attack Performance

We present *Keynergy*'s overall attack accuracy, and acoustics-only attack accuracy. We utilize 74 different Schlage SC1 keys, out of 75 randomly purchased keys, with one key filtered out due to the lack of distinct clusters (see Section 4.2). We collect key insertion audio when a single participant inserts all 74 keys for ten trials per key in a dormitory room with the representative Mic_{cond} located 1ft away from the door setup.

6.2.1 Overall Attack Accuracy

We present the overall results by combining audio and visual information for reducing keyspace as depicted in Figure 11. We plot in sorted order the keyspace reduction results of *Keynergy* (i.e., *Video + Acoustics* approach) depicted with '$*$' (red curve) across all 74 keys averaged over ten trials per key. We also plot the results of the baseline (i.e., *Video-only* approach) in sorted order, depicted with 'o' (blue curve) by utilizing the prediction error distribution again, on all 74 keys averaged over ten trials per key (based on real-world video analysis further explained and evaluated in Section 6.3.3). The *video-only* approach yields an average keyspace reduction (i.e., $Pool_{search}$) of 166 keys (with a recall of 92% ($\sigma = 62$) with a minimum of 15 and a maximum of 242 keys. *Keynergy* further significantly improves the results by achieving an average rank (i.e., $Rank_{key}$) of 63 with 92% recall ($\sigma = 47$). The combined acoustics-video approach achieves the smallest and largest average rank of 1 and 206, respectively, with *six keys achieving an average rank below 10* across ten different iterations. The results demonstrate around 62% improvement of *Keynergy* (i.e., *Video + Acoustics* approach) over the *Video-only* approach on average. We note that many of the 166 keys (in the reduced keyspace from video-only approach) contain similar bittings, rendering this further reduction to 60 keys significantly difficult. This fine-grained reduction is possible because *Keynergy* makes use of resulting *click patterns* that produce subtle differences, ultimately having acoustics complement the *Video-only* approach.

6.2.2 Acoustics-only Attack Accuracy

To further study the effects of the acoustics-based reduction, we evaluate the *Acoustics-only* approach by depicting the sorted $Ratio_{Reduction}$ on all 74 keys in Figure 12. Overall, this approach yields an average reduction rate of 75%, with 87% of keys (i.e., 65 keys) achieving more than 50% reduction. This result translates to an average $Rank_{key}$ of around 14,835, with highest rank of 119. This result demonstrates the utility of acoustics for key inference, while also depicting its insufficiency to realise a practical attack on its own. *Keynergy* overcomes this challenge by combining audio and video modalities, and achieves high reduction ratios (> 99%).

6.3 Modules Evaluation

We evaluate the different modules of our *Keynergy* design and use the results to justify our choice of model parameters.

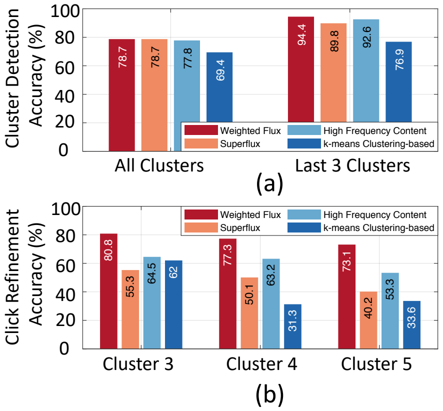


Figure 13: Performance of (a) different cluster detection approaches, (b) different click detection approaches.

6.3.1 Click and Cluster Detection Performance

To evaluate this module, we capture audio recordings from the Mic_{cond} at the university's lecture hall. We recruited three participants to insert three keys (different from the aforementioned 75 keys) with twelve trials per key, ultimately totaling 108 key insertions. In order to identify the best approach for cluster and click detection, we evaluate different onset detection techniques, namely – (a) *Weighted Flux* [14], that captures *differences* in high-frequency energies (see Section 5.2), (b) *Superflux* [18], that captures energy differences and is robust to signal fluctuations in frequency and loudness, (c) *High Frequency Content* [41], that captures high frequency energies, and (d) *K-Means Clustering* [37], which is a well-known clustering approach that identifies unique spectral energy distribution around click onsets. We consider a frequency range of $15-48kHz$ for the first three approaches as high frequencies capture sudden variations in energy, while we consider the entire frequency range for the clustering-based approach to identify additional lower frequency features that contribute to click detection. For all approaches, we fix the spectrogram window size to a low value of 127 (about 0.66 ms) with a 75% overlap between windows for better time resolution of clicks. Furthermore, we manually annotate the clicks and their corresponding clusters to utilize it as the ground truth.

Cluster Detection. We evaluate the performance of the *Cluster Detection* sub-module by plotting the cluster detection accuracy when varying across the different techniques. Specifically, we assign a score of *one* when the computed cluster boundaries include all of the manually annotated clicks belonging to the cluster, and *zero* otherwise. As depicted in Figure 13(a), *Weighted Flux* technique yields highest cluster detection accuracy of 78.7% across all clusters, and 94.4% across the last three clusters. From this sub-module, we empirically choose the optimal values of design parameters including an *amplitude threshold* of 0.15, and a *minimum duration between adjacent clicks* of 4 time windows (i.e., 0.66 ms).

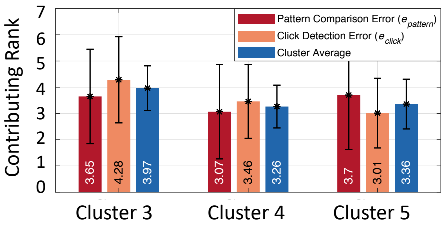


Figure 14: Figure depicts *contributing rank* of *cluster–error* pairs as well as each cluster, where a lower *contributing rank* implies higher contribution towards final $Rank_{key}$.

Click Detection Refinement. We evaluate the click detection refinement accuracy within each cluster by comparing detected clicks against the manually annotated clicks. We assign scores based on the ratio of correctly detected clicks but penalize any wrong clicks by assigning *zero* for the entire cluster. Figure 13(b) shows that *weighted-flux* technique yields highest refinement accuracy for all three clusters with 80.8%, 77.3% and 73.1%, respectively. We obtain *prominence threshold* of 0.2, and a *minimum duration between adjacent clicks* of 5 time windows (i.e., 0.83 ms). The increase in the minimum duration between adjacent clicks (from 0.66 ms to 0.83 ms) in the refinement stage is to thwart detection of noisy peaks in the vicinity of real clicks.

6.3.2 Pattern Comparison Performance

Recall from Section 5.4 that *Keynergy* aggregates two error functions – $e_{pattern}$ and e_{click} – for each cluster (i.e., *Clusters* 3, 4 and 5), to perform pattern comparison to achieve $Rank_{key}$. We evaluate individual contributions of the six *cluster – error* pairs towards computing $Rank_{key}$. We utilize the data collected for Section 6.3.1, and assign a *contributing rank* to each of the six pairs ($Contr_{pair}$, ranging from 1 to 6) by comparing their resulting individual $Rank_{key}$. For example, for a given victim key, if $\{Cluster\ 4 - e_{click}\}$ pair yields the highest individual $Rank_{key}$, then it is *most contributing* towards the final rank, hence assigned $Contr_{pair} = 1$.

Figure 14 depicts the average $Contr_{pair}$ of each *cluster – error* pair averaged over 74 input victim keys. $\{Cluster\ 5 - e_{click}\}$ pair yields the highest average $Contr_{pair}$ of 3.01, which can be attributed to higher chances of missing clicks in *Cluster 5*, thereby helping to filter out many candidate keys in the keyspace to yield the highest $Rank_{key}$. Similarly, to identify the contribution of different clusters, we compute per-cluster *contributing rank* ($Contr_{cluster}$) by taking an average across the $Contr_{pair}$ for its $e_{pattern}$ and e_{click}. We observe that *Cluster 4* yields the highest average $Contr_{cluster}$ of 3.26 mainly because it achieves a good trade-off amongst the three clusters, by having sufficiently unique *click patterns* for achieving low $e_{pattern}$, as well as having a fair chance of missing clicks, which can aid in filtering out keys based on e_{click}.

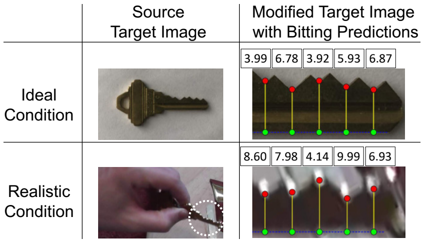


Figure 15: Source key image and its key-bit depth on transformed image in ideal and realistic conditions.

Scenario	Prediction Error		Average Key Space Reduction ($Pool_{search}$)
	Mean (μ)	Standard Deviation (σ)	
Ideal	0.123	0.087	**1**
Realistic	-0.64	1.28	**166.16**

Table 1: Comparison of ideal vs. realistic visual domain

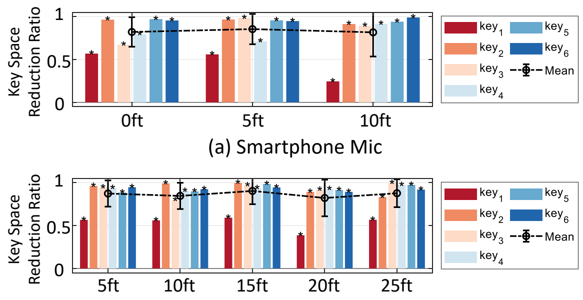
6.3.3 Video Analysis Performance

Recall from Section 5.5 that we re-implement the prior work on image-based key inference (*Sneakey* [35]), by extending it as a video-based inference. We evaluate our implementation to demonstrate that our (1) image-based inference implementation is comparable to that of *Sneakey* for an *idealistic scenario*; and (2) video-based inference implementation for a *realistic scenario* yields significantly lower accuracy.

Specifically, we demonstrate that image-based inference performs well with images capturing an immobile key at a certain angle (e.g., lying flat on surface) which do not contain any blurriness (ideal scenario). We capture ten birds-eye view images (two images for five distinct Schlage SC1 keys) lying flat on a table with a resolution of 1000 × 500. Figure 15 depicts such an image as well as its perspective transformation via a reference image with predicted bitting values. The prediction of bittings for all ten images correctly matches the target key bittings (after rounding up/down). Table 1 depicts the mean bitting prediction error, $\mu = 0.123$ and $\sigma = 0.087$.

On the contrary, we demonstrate that the video-based inference does not perform well with video frames capturing a moving key with uncontrolled angle (e.g., a person holding the key) inherently causing blurriness from motion (realistic scenario). We use five YouTube videos [65–69] that depict the key insertion of Schlage SC1 keys with varying camera angles and backgrounds (we post the key insertion segments used for our analysis here: `https://bit.ly/3pr5aFS`). We utilize these videos because they contain the ground-truth bitting information. We compute and choose the top three frames with least blurriness across the five videos resulting in 15 images with resolutions of 1080p for three videos and 720p for the rest. Hence, we obtain a mean bitting prediction error of $\mu = -0.64$ and $\sigma = 1.28$ (depicted in Table 1).

Moreover, we model Gaussian distributions based on the μ and σ values to simulate the average keyspace reduction ($Pool_{search}$) for ideal and realistic conditions, by sampling an error value from the distribution and adding it to true bittings across the 74 keys, across ten trials. While the idealistic condition results in a $Pool_{search}$ of just *one* on average (i.e., correctly identifying the victim key), the realistic condition yields a $Pool_{search}$ of around 166 keys, highlighting the impracticality of the *Video-only* approach (also depicted in Table 1).



(a) Smartphone Mic

(b) Parabolic Mic

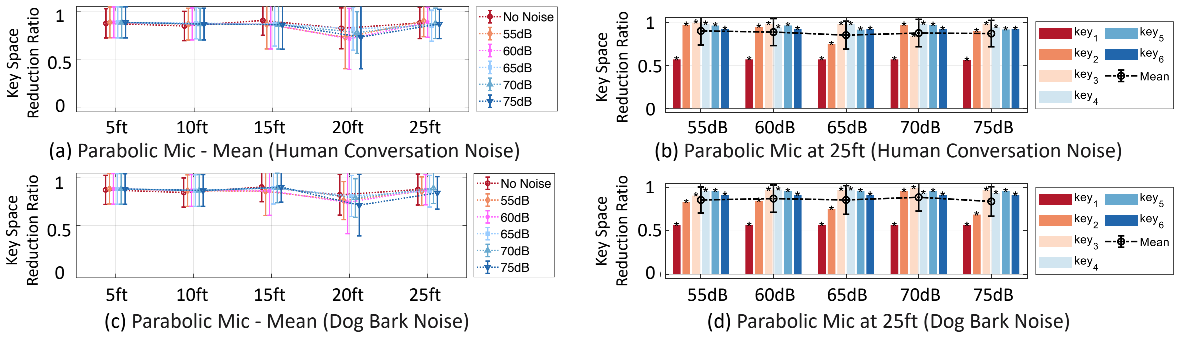
Figure 16: Attack success rates for Mic_{phone} and Mic_{parab} at different distances from the door.

6.4 Differing Experimental Conditions

We now evaluate *Keynergy* over several factors including attack distance, noise level, microphone type and varying participants. In order to evaluate different conditions, we choose keys with different ***key types***, i.e., with different ***cluster distinctness***, in order to understand their effect on the attack accuracy. Recall from Section 4 that presence of distinct clusters is integral to *Keynergy*'s attack. Hence, we select six keys $\{key_1, \ldots, key_6\}$, sorted from low to high *cluster distinctness* scores, where a high score indicates the presence of more distinct clusters (defined in Appendix F). Although there are several other factors including *human insertion* and *errors in click detection* that may additionally affect the results, they do not influence the vulnerabilities of *key types*.

6.4.1 Attack Scenario 1: *Proximity Attack*

We evaluate Mic_{phone} for varying distances from 0ft up to 10ft (or 3m) as depicted in Figure 16(a). We achieve an average $Ratio_{Reduction}$ of 90% ($\sigma = 10\%$) across all distances and keys (excluding key_1), which demonstrates attack feasibility up to 10ft. However, key_1 achieves a low $Ratio_{Reduction}$ of 46% ($\sigma = 18\%$) due to incorrect cluster detection, which can be attributed to its low *cluster distinctness*. Furthermore, we observe cluster misdetection for key_3 at 0ft, and click misses for key_4 at 5ft, both of which are likely to be due to human factors, and not the attack distance. However, for distances beyond 10ft, click amplitudes approach the noise floor resulting in detection of fewer clicks, leading to less reliable results.



(a) Parabolic Mic - Mean (Human Conversation Noise)

(b) Parabolic Mic at 25ft (Human Conversation Noise)

(c) Parabolic Mic - Mean (Dog Bark Noise)

(d) Parabolic Mic at 25ft (Dog Bark Noise)

Figure 17: Figure depicts the attack success rates for Mic_{parab} with two different noise sources, namely human conversation ($noise_{talk}$) and dog barking ($noise_{bark}$) at varying distances from the door and noise amplitudes.

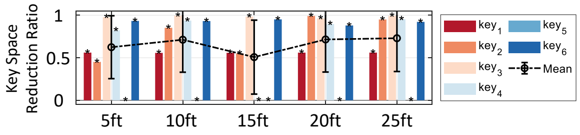


Figure 18: Attack success rates for Mic_{parab} with key bunch ($noise_{bunch}$) at different distances from the door.

6.4.2 Attack Scenario 2: *Distant Attack*

We utilize the Mic_{parab} to conduct the following three experiments, namely differing (1) *Door–Microphone Distance*, (2) *Noise Types*, and (3) *Ambient Noise Levels*.

Varying Door-Microphone Distances. We vary the distance from 5ft to 25ft, in increments of 5ft. As depicted in Figure 16(b), we obtain an average $Ratio_{Reduction}$ of 93% ($\sigma < 5\%$) for keys (except key_1) across all distances. Similar to the evaluation on phone, we observe poor cluster detection performance for key_1 due to its low *cluster distinctness*. Furthermore, in keys, key_2 to key_5, we observe occasionally low $Ratio_{Reduction}$ scattered across all distances, e.g., for key_3 at 10ft and key_2 at 25ft due to misses in click detection. In general, with increasing distance, click sounds may become fainter due to lower signal-to-noise ratio, hence increasing click detection errors. However, as Mic_{parab} achieves a high average reduction of 94% ($\sigma = 7\%$) even at 25ft distance (for all keys except key_1), this conveys the feasibility of utilizing such microphones for long-distance attacks.

Varying Noise Types and Ambient Noise Levels. We evaluate our approach by introducing three common noise sources: human conversation ($noise_{talk}$), barking dog ($noise_{bark}$) and sound due to a bunch of keys ($noise_{bunch}$), that may interfere with the key insertion sound.

- *$noise_{talk}$ and $noise_{bark}$:* To simulate the noise due to human and dog, we utilize 20 seconds of publicly available high-quality recording ($F_s = 96kHz$) [25,26], and play them at different noise levels (i.e., $55dB - 75dB$ in increments of $5dB$) through an Adam Audio A3X studio monitor speaker [11] for accurate sound reproduction. As human chatter and dog barking sounds are independent of key insertion, we record them separately using the Mic_{parab}, and combine the key insertion audio with randomly selected equal-duration noise. Hence, across different noise levels, the key insertion recording remains the same, while only the noise varies, hence removing the variability due to key insertion for the analysis.

From Figures 17(a) and 17(c), we observe that $noise_{talk}$ and $noise_{bark}$ at all noise levels have little impact on the $Ratio_{Reduction}$. Although these noise sources have energies up to $25kHz$ which can negatively affect our onset detection, the influence is to a lower degree as their energies are mostly concentrated below $5kHz$ for both $noise_{talk}$ and $noise_{bark}$. Furthermore, from Figures 17(b) and 17(d), we observe that the impact of noise sources, $noise_{talk}$ and $noise_{bark}$, respectively, is minimal even at an attack distance of 25ft. We present all the results across different distances and noise levels in Appendix H (see Figure 25).

- *$noise_{bunch}$:* Unlike the sounds due to human and dogs, the sound of key bunch is dependent on the key insertion action, and is difficult to be recorded separately. Consequently, we capture audio recordings by inserting each of the six keys into the lock, while having two additional keys in the key bunch. $noise_{bunch}$ consists of significant energies up to $48kHz$, which is similar to that of the "click" sound. Hence as depicted in Figure 18, presence of this noise degrades the $Ratio_{Reduction}$ from 93% (in the noise-free scenario) to 68% ($\sigma = 41\%$) on average for all keys (except key_1), demonstrating a significant degradation in attack's success. $noise_{bunch}$ increases the noise level of the key insertion sound, resulting in many false positives in click detection, and even leading to incorrect cluster detection among keys with relatively high *cluster distinctness* (e.g., key_5). Despite the above challenges, if the effect of $noise_{bunch}$ is less intense in some key insertions, it may still succeed as our attack combines information across multiple key insertions (e.g., key_3, across all distances).

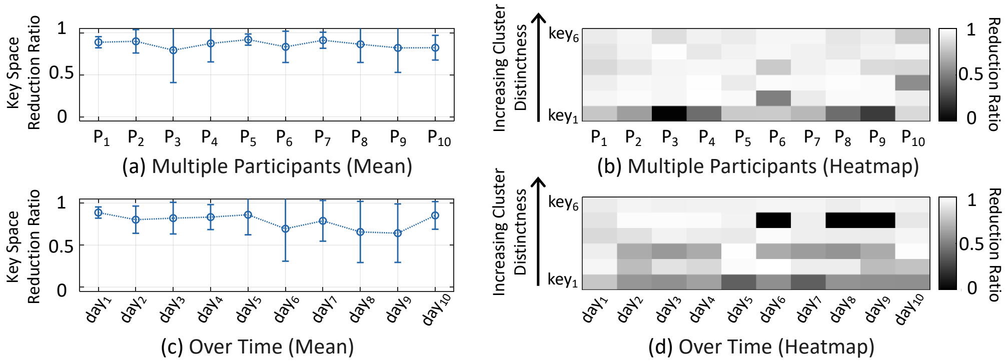


Figure 19: Attack success rates with two different parameters, namely *across multiple participants* and *over multiple days*.

6.4.3 Variations Across Multiple Participants

To understand the dependency of *Keynergy*'s accuracy on human factors, due to attributes such as age, sex and key gripping preferences, we recruit 10 participants, $\{P_1,...,P_{10}\}$ (demographics in Appendix E), where participant P_1 corresponds to the participant in Section 6.2.2, and capture audio recordings from each of them, with the Mic_{cond}, for a total of six keys, $\{key_1,...,key_6\}$, in increasing order of cluster distinctness. From Figure 19(a), we observe a high $Ratio_{Reduction}$ of 86% ($\sigma = 19\%$) across all participants and keys, while achieving an even higher $Ratio_{Reduction}$ of 92% ($\sigma = 10\%$) without key_1. Similar to previous results, from Figure 19(b), we observe that key_1 exhibits lower average $Ratio_{Reduction}$ of 57% due to poor cluster detection for four (out of ten) participants. Participants, P_6 and P_{10}, encounter issues in cluster detection for keys $\{key_2, key_3\}$ respectively, although they individually achieve more than 87% $Ratio_{Reduction}$ for the rest of the keys. While some participants achieve low reduction for certain keys, the $Ratio_{Reduction}$ across different participants tends to depend more on specific instances of insertions than the demographic characteristics and personal traits of the participants.

6.4.4 Variations Over Time

We capture audio recordings with the Mic_{cond} across ten days, $\{day_1,...,day_{10}\}$, where day_1 corresponds to the subset of data presented in Section 6.2.2. Figure 19(c) depicts the mean $Ratio_{Reduction}$ across the ten days of 78% ($\sigma = 25\%$). High variability in the results can be attributed to two keys, key_1 and key_5 as depicted in Figure 19(d). As expected, key_1 achieves a low $Ratio_{Reduction}$ of 56% ($\sigma = 12\%$) across all days due to its low *cluster distinctness*. On the other hand, key_5 occasionally detects additional clicks in the noise floor, which results in no reduction, although it achieves correct cluster detection due to its high *cluster distinctness*.

7 Discussion

We now discuss limitations of *Keynergy*, potential countermeasures against it, and its generalizability to other locks.

Limitations. Despite *Keynergy*'s considerable keyspace reduction under several experimental conditions, it has the following limitations. *Keynergy*'s attack accuracy is affected by high-frequency noise, although it remains robust to most common noises (such as human-chatter and dog-bark). In addition, our approach requires microphones with frequency response above $20kHz$ (present in most smartphones) in order to detect clicks, rendering consumer IoT devices with low-end mics such as smart doorbells unsuitable, despite their proximity to the door lock (see Appendix G). In the same vein, prolonged usage of keys can affect our inference by smoothing ridges in keys, thereby degrading the sharpness of click sounds and their detection accuracy. However, certain keys in our experiments have been inserted well over 300 times, with little impact on their click pattern, hence indicating the effectiveness of our attack for long durations. Lastly, we believe that despite our best attempts to design an inference framework that handles varying insertion speeds and mic types, its accuracy can be improved, not with availability of better hardware, but with modeling of human factors behind key insertions.

Countermeasures. We envision the following countermeasures. First, *physical modifications* that modify the target lock design to make them attack-resistant could be implemented, e.g., lock companies may produce keys with noise-dampening material (similar to 3D printed keys [20]), to reduce key insertion sound. However, such mitigation would require changes in manufacturing, and would not protect vulnerable keys already in circulation. The lock industry could also transition to more secure pin-tumbler lock designs, such as the Bowley locks [3] which have no ridges that cause click sounds, hence making them potentially immune to audio-based key inference. Second, from our analysis, lock manufacturers can identify vulnerable keys (i.e., keys with *distinct clusters*), and avoid their production/sale. However, removing all such keys, i.e., 79% of keys (see Appendix F) would likely introduce new attack avenues due to reduced keyspace. Hence, manufacturers need to strike a balance by discarding keys with higher *cluster distinctness* as they may be more susceptible to *Keynergy*'s attack, while maintaining a sufficiently large keyspace. Third, we envision injecting noise to corrupt key insertion sounds. This can be achieved by first detecting the key insertion event (from video footage of outdoor cameras or smart doorbells), and playing inaudible sounds of frequency greater than $15kHz$, using devices such as smart doorbells. Furthermore, noise signals should exhibit temporal variations (in frequency or amplitude), as *Keynergy*'s acoustic inference utilizes energy *differences* over time for detecting clicks, hence making constant noise an ineffective defense. Alternately, instead of detecting key insertion, inaudible noise can be played continuously, although this consumes more power.

Generalizability. Although empirical evaluations of *Keynergy*'s framework were conducted on Schlage 5-pin locks, due to the similarity of the pin tumbler lock design across the industry, *Keynergy* can be easily tailored to attack other common lock models, including those with more than five pins. Our preliminary analyses on Kwikset and Yale 5-pin locks, as well as Schlage 6-pin locks show promise. As part of future work, we consider extending *Keynergy*'s design for tackling high-security pin tumblers such as Mul-T-Lock cylinders that have telescoping pins design (i.e., pin within pin), hence requiring inferring ten bittings in place of five, and Medeco Biaxial, that have keys with angled bitting cuts, thereby necessitating guessing angles together with bittings [54].

Keynergy's approach of leveraging time-intervals between audible clicks has broader applicability beyond lock security. In the past, researchers have designed "acoustic barcodes" by creating structured patterns on objects which when swiped produce a series of click sounds, where the timing between adjacent clicks encodes information [29]. On similar lines, we believe click timing information can be exploited for communicating secret information via covert channels.

8 Related Work

We now present related work that investigates the security of physical locks, and acoustic side-channels.

Physical Locks Security. There have been several attacks compromising the security of lock mechanisms in a non-destructive manner, which can be broadly divided into two categories. The first type of attacks requires physical access to the lock during attack execution, such as bumping, lock picking and rights amplification in pin tumbler locks [16, 48, 49, 70]. The second type, to which *Keynergy* belongs, is *stealthy offline attacks* that involve passively capturing sensor information in order to infer the keycode [32, 33, 35, 39, 55, 61]. One such work *Sneakey* proposes to use a telephotography camera to infer bittings based on still images [35]. Although a novel approach, *Sneakey* makes several unrealistic assumptions, including requiring a high-resolution image of a stationary key placed at a certain angle, thus greatly reducing its practical feasibility. Another related effort, *SpiKey* [55], proposes a key inference framework that employs simulations of acoustic emanation from key insertions. One of *SpiKey*'s main drawbacks is that it assumes a *constant insertion speed*, and thus would not work in practical settings where users insert keys with varying speeds. *Keynergy* addresses these challenges that arise due to *unknown* and *inconsistent* key insertion speeds, and in addition, achieves reasonable key space reduction even at distances up to 25 feet at varying noise levels.

Acoustic Side-Channels. Several sensor-based side-channel attacks have been proposed to infer confidential information such as cryptographic keys [28], ATM pins [43], keystrokes [12, 38, 40, 42], taps on a touch screen [21, 50] and stylus pen writing [36] among many others. Specifically, within the audio domain, researchers have exploited acoustic leakage from various physical components including laptop's power supply unit [28], computer screens [27], keyboards [10, 71], 3D printers [30, 62] and DNA synthesizers [24] to infer private information. Different from the above works, *Keynergy* utilizes the sound emanated from physical locks and keys during the event of key insertion to infer the key's secret code.

9 Conclusion

We propose *Keynergy*, a novel *stealthy offline attack* that infers the victim's key *bittings* by extending and improving the prior image-only attack by utilizing the audible clicks captured during victim's key insertion. *Keynergy* combines insufficient information from each of the sensing modalities, namely audio and video, which complement each other to yield a novel and practical side-channel attack. *Keynergy* overcomes the shortcomings of traditional attacks on pin tumbler locks of requiring physical access to the lock throughout the attack, which increases the chances of the attacker getting caught. We conduct proof-of-concept real-world experiments by recruiting 13 participants and testing with 75 different keys, totaling more than 3,600 insertions. We examine the impact of varying real-world conditions, including eavesdropping distance and ambient noise levels across different microphone types. With acoustics alone, *Keynergy* obtains an average keyspace reduction of 75% and on combining acoustics and visual information, *Keynergy* achieves a reduction in keyspace below *ten* keys for 8% of the keys (i.e., six keys out of 75 keys tested).

10 Acknowledgements

We thank our shepherd and anonymous reviewers for their insightful comments. This work is supported in part by grants from the Singapore Ministry of Education Academic Research Fund Tier 1 (R-252-000-A26-133 and R-252-000-B40-114) and the US National Science Foundation (NSF) under award number 1943351.

References

[1] AKG Lyra condenser microphone. https://www.akg.com/lyra.html, 2020.

[2] Behringer UMC202HD audio interface. https://bit.ly/38qkuN3, 2020.

[3] Bowley Lock Company Inc. https://www.bowleylockcompany.com/, 2020.

[4] Countryman B3 omnidirectional lavalier microphone. https://bit.ly/39d2d4Z, 2020.

[5] Google Pixel. https://www.gsmarena.com/google_pixel-8346.php, 2020.

[6] Nest Hello Doorbell. https://store.google.com/us/product/nest_hello_doorbell, 2020.

[7] Ring Doorbell. https://ring.com/, 2020.

[8] Sound Shark parabolic microphone. https://kloverproducts.com/shop/sound-shark/no-mic/sound-shark/, 2020.

[9] Yale key specifications. https://bit.ly/2Xhj5lh, 2020.

[10] Dmitri Asonov and Rakesh Agrawal. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy*, pages 3–11. IEEE, 2004.

[11] Adam Audio. A3x studio monitor. https://www.adam-audio.com/en/ax-series/a3x/, 2020.

[12] Davide Balzarotti, Marco Cova, and Giovanni Vigna. Clearshot: Eavesdropping on keyboard input from video. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 170–183. IEEE, 2008.

[13] Banggood. Linear actuator. https://bit.ly/3hQx8rv, 2020.

[14] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on speech and audio processing*, 2005.

[15] Marie Black. 360 d819 video doorbell. https://www.techadvisor.co.uk/review/360-video-doorbell-3700933/, 2019.

[16] Matt Blaze. Rights amplification in master-keyed mechanical locks. *IEEE Security & Privacy*, 2003.

[17] Matt Blaze. Notes on picking pin tumbler locks. https://www.mattblaze.org/papers/notes/picking/, 2016.

[18] Sebastian Böck and Gerhard Widmer. Maximum filter vibrato suppression for onset detection. In *Proc. of the 16th Int. Conf. on Digital Audio Effects (DAFx). Maynooth, Ireland (Sept 2013)*, volume 7, 2013.

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

[20] Ben Burgess, Eric Wustrow, and J Alex Halderman. Replication prohibited: attacking restricted keyways with 3d-printing. In *USENIX Workshop on Offensive Technologies*, 2015.

[21] Liang Cai and Hao Chen. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. *HotSec*, 11(2011):9, 2011.

[22] Will Christensen. Key machines that cut it. https://bit.ly/2MHVKHB, 2020.

[23] Adam Clark Estes. The history and future of locks and keys. https://gizmodo.com/the-history-and-future-of-locks-and-keys-1735694812, 2015.

[24] Sina Faezi, Sujit Rokka Chhetri, Arnav Vaibhav Malawade, John Charles Chaput, William Grover, Philip Brisk, and Mohammad Abdullah Al Faruque. Oligo-snoop: A non-invasive side channel attack against dna synthesis machines. In *NDSS*, 2019.

[25] FreeSound. Dog barking noise. https://freesound.org/people/felix.blume/sounds/199261/, 2013.

[26] FreeSound. Human conversation noise. https://freesound.org/people/rampartian/sounds/236786/, 2014.

[27] Daniel Genkin, Mihir Pattani, Roei Schuster, and Eran Tromer. Synesthesia: Detecting screen content via remote acoustic side channels. In *IEEE S&P*, 2019.

[28] Daniel Genkin, Adi Shamir, and Eran Tromer. Acoustic cryptanalysis. *Journal of Cryptology*, 2017.

[29] Chris Harrison, Robert Xiao, and Scott Hudson. Acoustic barcodes: passive, durable and inexpensive notched identification tags. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, 2012.

[30] Avesta Hojjati, Anku Adhikari, Katarina Struckmann, Edward Chou, Thi Ngoc Tho Nguyen, Kushagra Madan, Marianne S Winslett, Carl A Gunter, and William P King. Leave your phone at the door: Side channels that reveal factory floor secrets. In *ACM CCS*, 2016.

[31] IBISWorld. Door lock & lockset manufacturing industry in the us - market research report. https://bit.ly/38jaPrq, 2019.

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

[33] Keys4Classics. Homepage. http://www.keys4classics.com, 2019.

[34] David Lawrence, Eric Van Albert, and Robert Johnson. Key decoding and duplication attacks for the schlage primus high-security lock. https://bit.ly/396A27v, 2013.

[35] Benjamin Laxton, Kai Wang, and Stefan Savage. Reconsidering physical key secrecy: Teleduplication via optical decoding. In *ACM CCS*, 2008.

[36] Yihao Liu, Kai Huang, Xingzhe Song, Boyuan Yang, and Wei Gao. Maghacker: eavesdropping on stylus pen writing via magnetic sensing from commodity mobile devices. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pages 148–160, 2020.

[37] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 1982.

[38] Anindya Maiti, Oscar Armbruster, Murtuza Jadliwala, and Jibo He. Smartwatch-based keystroke inference attacks and context-aware protection mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, 2016.

[39] Anindya Maiti, Ryan Heard, Mohd Sabra, and Murtuza Jadliwala. Towards inferring mechanical lock combinations using wrist-wearables as a side-channel. In *ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 111–122, 2018.

[40] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. (sp) iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security*, 2011.

[41] Paul Masri. *Computer modelling of sound for transformation and synthesis of musical signals.* PhD thesis, University of Bristol, 1996.

[42] John V Monaco. Sok: Keylogging side channels. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 211–228. IEEE, 2018.

[43] Keaton Mowery, Sarah Meiklejohn, and Stefan Savage. Heat of the moment: Characterizing the efficacy of thermal camera-based attacks. In *Proceedings of the 5th USENIX conference on Offensive technologies*, 2011.

[44] Google Nest. Nest hello video doorbell. https://store.google.com/us/product/nest_hello_doorbell, 2020.

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

[46] Phillip Nichols. The ps and qs of parabolic microphones. https://bhpho.to/3nqbo70, 2019.

[47] The Locksmith Security Association of Michigan. Key tech manuals. https://www.lsamichigan.org/tech_manuals.html, 2020.

[48] Deviant Ollam. Ten things everyone should know about lockpicking & physical security. https://bit.ly/35ead4b, 2008.

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

[50] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. Accessory: password inference using accelerometers on smartphones. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, pages 1–6, 2012.

[51] José Luis Pech-Pacheco, Gabriel Cristóbal, Jesús Chamorro-Martinez, and Joaquín Fernández-Valdivia. Diatom autofocusing in brightfield microscopy: a comparative study. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. IEEE.

[52] Dave Pedu. 3d printing real-world keys. https://hackaday.io/project/27631-3d-printing-real-world-keys, 2020.

[53] Said Pertuz, Domenec Puig, and Miguel Angel Garcia. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*, 46(5):1415–1432, 2013.

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

[55] Soundarya Ramesh, Harini Ramprasad, and Jun Han. Listen to your key: Towards acoustics-based physical key inference. In *International Workshop on Mobile Computing Systems and Applications*, pages 3–8, 2020.

[56] Best Reviews. Listening devices for long distance listening. https://bit.ly/3hRLisE, 2019.

[57] Ring. Ring peephole camera. https://bit.ly/38jGpVS, 2020.

[58] Ring. Ring video doorbell 3 plus. https://bit.ly/2Xlfnac, 2020.

[59] Xavier Rodet and Florent Jaillet. Detection and modeling of fast attack transients. In *ICMC*, 2001.

[60] Wade Shaddy. What is the standard doorknob height on a door? https://bit.ly/2XeCinI, 2020.

[61] Rory Smith and Tilo Burghardt. DeepKey: Towards End-to-End Physical Key Replication from a Single Photograph. In *German Conference on Pattern Recognition*, pages 487–502. Springer, 2018.

[62] Chen Song, Feng Lin, Zhongjie Ba, Kui Ren, Chi Zhou, and Wenyao Xu. My smartphone knows what you print: Exploring smartphone-based side-channel attacks against 3D printers. In *ACM CCS*, 2016.

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

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

[65] Key Insertion Video. How to master key a lock. https://www.youtube.com/watch?v=USXsU3p1uRM, 2011.

[66] Key Insertion Video. How to rekey a schlage lever lock. https://www.youtube.com/watch?v=S1unNV7YtjA?, 2014.

[67] Key Insertion Video. How to master rekey a schlage deadbolt changing the combination of a pin tumbler lock using two keys. https://www.youtube.com/watch?v=lP9WcVy7M1A&t=40s, 2016.

[68] Key Insertion Video. How to remove the door knob to a schlage lock. https://www.youtube.com/watch?v=jbk2jQS3PRQ, 2018.

[69] Key Insertion Video. Mark's locksmith- kwikset vs schlage locks! which is better. https://www.youtube.com/watch?v=owxdiDy0k5U, 2018.

[70] Barry Wels and Rop Gonggrijp. Bumping locks. https://toool.nl/images/7/75/Bumping.pdf, 2005.

[71] Li Zhuang, Feng Zhou, and J Doug Tygar. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security*, 13(1):1–26, 2009.

Appendix A Key Specifications

In Section 2, we explain the key *specifications* of pin-tumbler locks, which are responsible for the discrete nature of bittings, and hence its associated fixed keyspace. In this section, we elaborate on the different specification parameters, and their importance for our acoustic attack. The part of the key that enters the lock is known as the *key blade*, depicted in Figure 20, with its two ends known as the *shoulder* and the *tip*, respectively. Each *key blade* (for a particular lock system) has a unique profile, which is dictated by the *bitting specification* that is fixed by the manufacturer to ensure proper functioning of the key [63]. These *bitting specifications*, listed in the specification sheet provided by the lock manufacturer, indicate the number of bitting positions (generally 5-6), possible depth values per bitting (generally 7-10), width of each bitting (i.e., *cut root*), as well as, the angle at the bitting position (i.e., *cut angle*). In addition, each bitting depth is restricted to a set of uniformly-spaced discrete-valued depths, which differ from each other by an *increment*, which is also specified in the *bitting specification*. Also, the distance between two adjacent bittings should be equal to the distance between two adjacent pins in the lock, i.e., its *inter-pin distance* (as shown in Figure 20), for its proper functioning. The *consistency* of key specification parameters is crucial to our attack as the modeling for *simulated patterns* would not be possible otherwise.

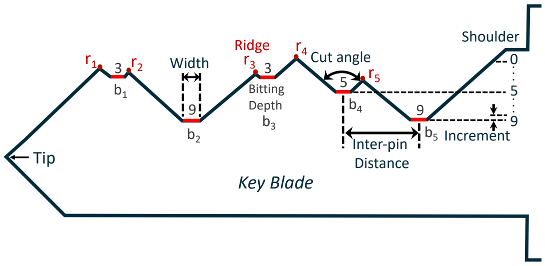


Figure 20: Figure depicts the *key blade*, *bittings* ($b_1, \ldots, b_5$), key *specification* parameters, and also *ridges* ($r_1, \ldots, r_5$), which are crucial for producing sound during key insertion.

Appendix B Constraints on Keyspace

In Section 2, we briefly explain the factors responsible for constraining the keyspace of pin-tumbler locks. In this section, we elaborate on the various constraints added both for security as well as usability reasons, which ultimately result in considerable reduction to the keyspace. For any given lock and key model, due to the discrete nature of bitting depths, there is an upper bound on the maximum number of possible keys. For example, Schlage SC1 keys have 5 cut positions and 10 possible depths. Thus, the maximum number of Schlage SC1 keys that are possible is 10^5. However, in practice, the keyspace is close to 75% of the maximum, due to additional constraints imposed by the manufacturers for guaranteeing correct functioning of the keys, for example, MACS (explained in Section 2). Apart from MACS, additional constraints, also known as *coding rules* [54], are imposed for usability reasons and for preventing trivial duplication by sight. We list these constraints: ① only two adjacent bittings can be of same depth, ② a total of three or fewer bittings can be of same depth, ③ three or more bittings must be of different depth, and ④ sequence of bittings should not monotonically increase from

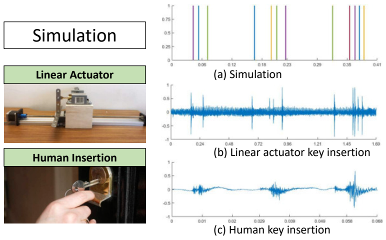


Figure 21: (a) depicts the *click pattern* obtained by simulating a constant insertion speed, (b) depicts audio time-series obtained from a constant insertion speed with a linear actuator, and (c) depicts an audio time-series of human key insertion.

the shoulder to the tip-end of the key.

Appendix C Feasibility of Click Pattern Detection

In order to confirm that the *click pattern* is observed upon a key insertion, we perform a preliminary feasibility study to compare the *simulation* based on constant insertion speed against two experiments – a custom designed setup using a linear actuator [13], such that it inserts a key into the lock at a constant speed of about 0.24 inches/sec), as well as a human insertion with an average speed of 5.4 inches/sec (about 20 times faster). Figure 21(a) depicts part of the resulting *click pattern* from the simulation, while Figures 21(b) and (c) depict the corresponding audio time-series produced by the linear actuator and human insertion, respectively. We observe that time-series from the linear actuator closely resembles the *click pattern*, although obtaining such a pattern from human key insertion is challenging due to the much higher and inconsistent insertion speed. As we explain in Section 4, human key insertion causes smaller group of clicks to aggregate together into what we refer to as “clusters”, within which the average speed is higher (10.8 inches/sec), but is also more consistent. Furthermore, the time-interval between clicks is at least 0.66 ms (see Section 6.3.1), which causes the minimum detectable click-distance to be 7.2 milli-inch, resulting in some distinct clicks (separated by distances as small as 3 milli-inch) in simulation to be missed during detection in human insertion.

Appendix D Simulated Patterns of Keys with Identical Bittings

As shown in Figure 22, a key with keycode 33333 (i.e., identical bittings values) has nine clicks (max = 15) in its *simulated patterns*, with two clicks in each cluster except the first. Such a pattern occurs because clicks due to all ridges, except ridge r_1, overlap (i.e., occur simultaneously). Owing to the geometry of the key, when the adjacent bittings are equal, the horizontal position of the ridge in-between, is exactly at their midpoint (e.g., when bittings, $b_1 = b_2$, ridge r_2 is formed at the center of b_1 and b_2). More generally, the position of a ridge is determined by the difference between adjacent bittings. In the case of keys with all identical bittings, ridges $\{r_2, r_3, r_4, r_5\}$ are located at the same location (i.e., midpoint) w.r.t their adjacent bittings, hence causing the clicks due to them to overlap. On the other hand, the position of ridge r_1, due to its unique placement at the tip of the key (see Figure 20), is determined by the value of a single bitting (b_1) alone. Hence, clicks due to the first ridge do not overlap with clicks due to rest of the ridges. We note that we discuss a key with equal bittings here for exemplary purposes. In reality, such keys are not valid as they do not satisfy the constraints listed in Appendix B.

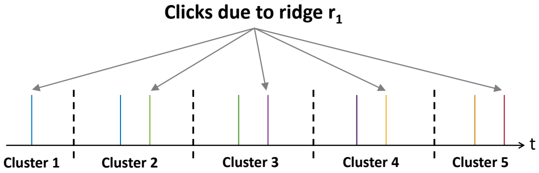


Figure 22: Figure depicts the *simulated patterns* of a key with keycode 33333, having nine clicks. In each cluster, clicks due to ridge r_1 are *distinct* from clicks due to all other ridges which occur simultaneously, owing to the key’s geometry.

Appendix E Participant Demographics

Table 2 provides the demographics of ten participants who took part in the testing phase where we study the effects of human attributes on keyspace reduction (see Section 6.4.3), as well as three participants who took part in the training phase. The participants consist of four females and nine males, with their ages ranging from 24 to 37. Of the thirteen participants, eight of them use physical keys on a daily basis. However, from our results, there seems to be no correlation between the regular usage of keys and the attack accuracy.

Participant	Gender	Age (years)	Height (cm)	Weight (kg)	Uses physical key regularly?
P_1	F	24	165	58	Yes
P_2	M	24	170	64	Yes
P_3	F	24	167	63	No
P_4	M	25	178	88	Yes
P_5	M	26	176	70	No
P_6	M	24	165	55	No
P_7	M	23	171	63	Yes
P_8	F	31	171	60	Yes
P_9	M	27	173	65	No
P_{10}	M	24	172	70	Yes
S_A	F	34	165	65	Yes
S_B	M	37	175	78	Yes
S_C	M	36	173	83	No

Table 2: Table presents the demographics of all participants.

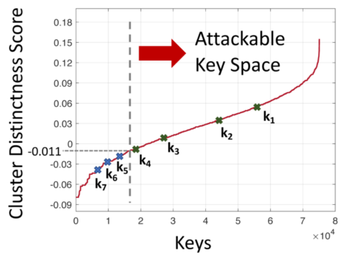


Figure 23: Figure depicts that 79% of keys which lie above −0.011 *cluster distinctness score* form the attackable keyspace for *Keynergy*. We obtain this threshold score by performing empirical analysis on key insertion of keys $k_1 - k_7$.

Appendix F Attackable Keyspace Determination

Recall from Section 4 that the presence of distinct clusters is necessary for our acoustic inference attack. In order to determine the set of keys that satisfy this constraint, we plot a *cluster distinctness* score that determines the distinctness of clusters based on the simulated model, for all keys in the keyspace as shown in Figure 23. We compute *cluster distinctness* score as the difference between the shortest inter-cluster duration (i.e., time-interval between clusters) and the longest intra-cluster duration (time-interval between clicks within a cluster), where a larger score indicates more distinct clusters. We observe that about 70% of keys have distinct clusters (i.e., score > 0). However, in order to determine if this distinctness holds true for real key insertions, we select seven keys ($k_1 - k_7$) that are outside of the 75 keys in our test set (see Section 6.2). We obtain the recordings from three participants for each key. Of the seven keys, three are above and four below the zero mark. We observe that $k_1 - k_4$ have distinct clusters, confirming the presence of larger durations between clusters in real insertion as compared to simulation. Hence, we determine the keyspace by identifying the point of steepest descent between keys, k_4 and k_5, which yields a score threshold of -0.011 and a keyspace of 59,207 keys (79% of 75,066 keys).

Appendix G Smart Doorbell Analysis

To investigate the possibility of an attacker who can remotely access a smart doorbell installed on the victim's door, we test the attack utility of key insertion audio recorded from multiple smart doorbell models [15, 44, 57, 58]. Figure 24 depicts the spectrogram of key insertion from two popular models – Ring Video Doorbell 3 Plus and Google Nest Hello. From our analysis, we infer that all the doorbells we investigate are equipped with low-quality microphones designed for human voice capture (i.e., having low-frequency response only up to $8kHz$), hence making them unsuitable for capturing fine-grained click timing information. Furthermore, due to the lack of options to change the audio quality in these devices, the doorbells save them in lossy Advanced Audio Coding (or AAC) format, which further degrades signal quality.

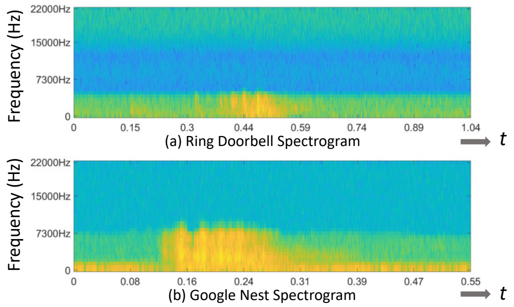


Figure 24: Figure depicts spectrogram of key insertion recorded using Ring 3 Plus and Nest Hello doorbells.

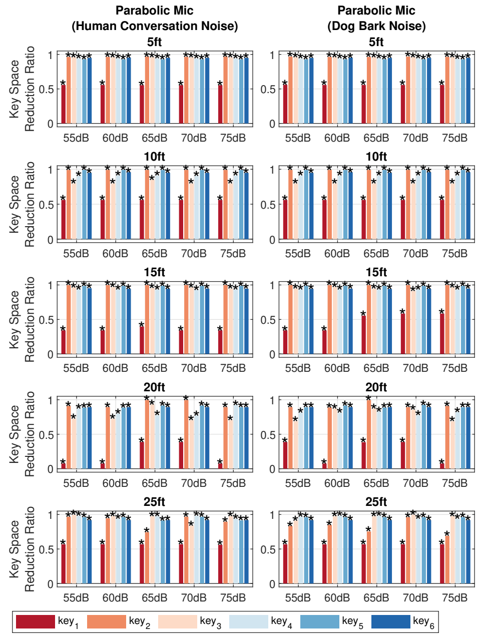


Figure 25: Figure depicts the impact of noise on Mic_{parab} at distances 5ft to 25ft on the $Ratio_{Reduction}$, for two types of noise – human conversation and dog barking sound.

Appendix H Noise Analysis of Parabolic Mic

In Section 6.4.2, we present the average reduction ratio ($Ratio_{Reduction}$) for Mic_{parab} across all keys for distances, namely 5ft to 25ft. In Figure 25, we depict the $Ratio_{Reduction}$ of individual keys at all distances, for two different noise types – human conversation ($noise_{talk}$) and dog barking sound ($noise_{bark}$). Similar to the analysis in Section 6.4.2, we observe low variance in $Ratio_{Reduction}$ across different noise levels, across all distances. These results illustrate that *Keynergy* is robust to low-frequency noise sources.