Auto-Key: Using Autoencoder to Speed Up Gait-based Key Generation in Body Area Networks

YUEZHONG WU, The University of New South Wales, Australia and CSIRO-Data61, Australia
QI LIN, The University of New South Wales, Australia and CSIRO-Data61, Australia
HONG JIA, The University of New South Wales, Australia and CSIRO-Data61, Australia
MAHBUB HASSAN, The University of New South Wales, Australia and CSIRO-Data61, Australia
WEN HU, The University of New South Wales, Australia and CSIRO-Data61, Australia

With the rising popularity of wearable devices and sensors, shielding Body Area Networks (BANs) from eavesdroppers has become an urgent problem to solve. Since the conventional key distribution systems are too onerous for resource-constrained wearable sensors, researchers are pursuing a new light-weight key generation approach that enables two wearable devices attached at different locations of the user body to generate an identical key simultaneously simply from their independent observations of user gait. A key challenge for such gait-based key generation lies in matching the bits of the keys generated by independent devices despite the noisy sensor measurements, especially when the devices are located far apart on the body affected by different sources of noise. To address the challenge, we propose a novel machine learning framework, called Auto-Key, that uses an autoencoder to help one device predict the gait observations at another distant device attached to the same body and generate the key using the predicted sensor data. We prototype the proposed method and evaluate it using a public acceleration dataset collected from 15 real subjects wearing accelerometers attached to seven different locations of the body. Our results show that, on average, Auto-Key increases the matching rate of independently generated bits from two sensors attached at two different locations by 16.5%, which speeds up the successful generation of fully-matching symmetric keys at independent wearable sensors by a factor of 1.9. In the proposed framework, a subject-specific model can be trained with 50% fewer data and 88% less time by retraining a pre-trained general model when compared to training a new model from scratch. The reduced training complexity makes Auto-Key more practical for edge computing, which provides better privacy protection to biometric and behavioral data compared to cloud-based training.

CCS Concepts: • **Security and privacy**; • **Computing methodologies** → *Neural networks*;

Additional Key Words and Phrases: Autonomic Symmetric Key Generation, Biometric Key Generation, Body Area Networks, Body Sensor Networks, Wearable Communications, Device Pairing, Machine Learning, Autoencoder, Transfer learning

Authors' addresses: Yuezhong Wu, The University of New South Wales, School of Computer Science and Engineering, Building K17, Kensington Campus, Sydney, NSW, 2052, Australia, CSIRO-Data61, Australia, yuezhong.wu@student.unsw.edu.au; Qi Lin, The University of New South Wales, School of Computer Science and Engineering, Building K17, Kensington Campus, Sydney, NSW, 2052, Australia, CSIRO-Data61, Australia, qi.lin@@unsw.edu.au; Hong Jia, The University of New South Wales, School of Computer Science and Engineering, Building K17, Kensington Campus, Sydney, NSW, 2052, Australia, CSIRO-Data61, Australia, h.jia@unsw.edu.au; Mahbub Hassan, The University of New South Wales, School of Computer Science and Engineering, Building K17, Kensington Campus, Sydney, NSW, 2052, Australia, CSIRO-Data61, Australia, mahbub.hassan@unsw.edu.au; Wen Hu, The University of New South Wales, School of Computer Science and Engineering, Building K17, Kensington Campus, Sydney, NSW, 2052, Australia, CSIRO-Data61, Australia, wen.hu@unsw.edu.au.

2474-9567/2020/3-ART32 $15.00
https://doi.org/10.1145/3381004

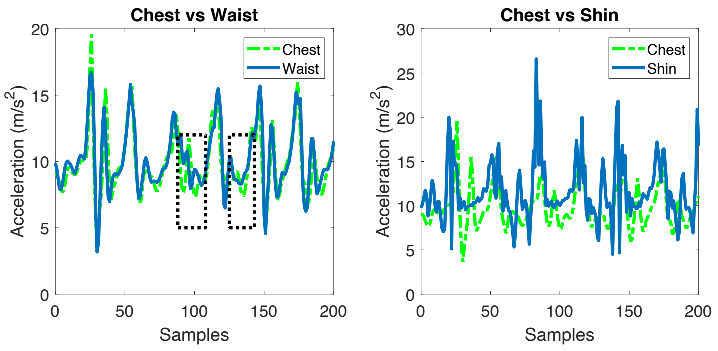


Fig. 1. Comparison of acceleration signals between chest-and-waist and chest-and-shin when a user is walking.

ACM Reference Format:
Yuezhong Wu, Qi Lin, Hong Jia, Mahbub Hassan, and Wen Hu. 2020. Auto-Key: Using Autoencoder to Speed Up Gait-based Key Generation in Body Area Networks. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 1, Article 32 (March 2020), 23 pages. https://doi.org/10.1145/3381004

1 INTRODUCTION

We are at the cusp of a revolution in wearable computing. There is rising popularity to wear various types of sensors and smart devices for better health, comfort, entertainment, and convenience. A recent survey [26] reveals that the market is already beaming with hundreds of different types of smart wearable products including smartwatches, smart bands, smart glasses, smart jewelry, electronic garments, skin patches, and so on. To further extend the capability and utility of wearable computing, new wireless communications standards have been released to connect multiple wearable sensors into a Body Area Network (BAN) [20]. However, as wearable devices capture sensitive personal health and lifestyle data, protecting wireless communications in BAN against potential eavesdropping has become a pressing issue to resolve.

Symmetric key cryptography is a common practice to secure communications against adversaries. It requires two parties to use a common secret key to encrypt and decrypt all communications between them, so an eavesdropper cannot access the data. Since storing secret keys permanently within mobile devices has a high risk of the keys being stolen, mobile communication networks employ dynamic distribution of keys among the communicating parties. Unfortunately, such dynamic key distributions have high overhead and often require communications with a trusted third party, which becomes too demanding for resource-constrained body sensors. Light-weight key distribution in BAN, therefore, has become a topic of intense research.

An interesting recent trend in the literature is to exploit unique biometric signals, such as gait and heartbeat, as a source of independent generation of symmetric keys in multiple body sensors [17, 25]. The key idea behind this approach lies in the fact that multiple wearable devices attached to the same human body can sense the same biometric signal at the same time. Thus, by converting the continuous biometric signal into discrete bits, two wearable devices can produce an identical bit string, which can be used as symmetric keys. This process of autonomic key generation is very attractive for BAN because biometric signals are unique and cannot be reproduced by an attacker, and they can be easily captured with low-cost sensors, such as accelerometers, that are already built into most wearable products.

The fundamental drawback of biometric key generation is that, due to the presence of various noise sources, the independent observations of the same biometric signal by two body sensors do not always match perfectly. This is particularly the case when the sensors are attached far apart to different body parts experiencing distinct localized noises. This phenomenon is illustrated in Figure 1, which, for a walking person, compares acceleration signals captured by accelerometers attached at chest, waist, and shin. We can see that the independent acceleration signals, which are supposed to capture the same gait signal of the user, are not exactly matching. The mismatch is particularly visible between chest and shin as the sensor attached to the shin is subject to significant noise due to the free motion of the leg. Unfortunately, such a signal mismatch increases the mismatch probability between the generated bit strings, which consequently requires more attempts to finally obtain two perfectly matching keys. The more attempts it requires to generate matching (symmetric) keys, the longer it takes to establish a secure communication session in BAN, thus damaging the overall user quality of experience with the technology.

In this paper, we propose to use machine learning to reduce signal and bit mismatch between two body sensors attempting to generate symmetric keys from gait observations. In particular, we propose a novel deep learning framework, called Auto-Key, that uses an autoencoder to help one body sensor predict the sensor data obtained at another body sensor and generate the key using the predicted sensor data. Our experiments with real body sensor data show that Auto-Key can significantly reduce the key generation time compared to previous methods that do not employ signal prediction.

The main contributions of our paper can be summarized as follows:

- For a walking person, we propose the concept of using acceleration sensor data obtained at one body location to predict the acceleration signal observed at a different body location and then use the predicted signal for key generation. We design an autoencoder framework to realize this prediction. To the best of our knowledge, this is the first attempt to speed up gait-based key generation using Machine Learning (ML).
- We implement and evaluate Auto-Key using a public dataset of acceleration signals collected from 15 walking subjects wearing accelerometers attached to six different body locations. Our results show that Auto-Key increases bit agreement rate by 16.5%, which speeds up key generation by more than 1.9X.
- We propose a transfer learning model that reduces the required training data and time of subject-specific autoencoders by 50% and 88%, respectively. Such reductions in training complexity make Auto-Key training more practical for edge computing, which provides better privacy protection to personal gait data compared to uploading them to third-party cloud servers.

The rest of the paper is organized as followed. Background and related work are reviewed in Section 2. In Section 3, We introduce the proposed machine learning framework in Auto-Key. Performance evaluation of Auto-Key is presented in Section 4 followed by a security analysis on Auto-Key in Section 5. The paper is concluded in Section 6.

2 BACKGROUND AND RELATED WORK

Autonomic symmetrical key generation exploits a common secret observation between the communicating parties to independently generate matching keys without requiring help from third parties. In recent years, researchers have explored a range of contexts for the secret observation, which includes vibrations from handshaking [7, 14], user heartbeat [11, 17, 24], and user gait [25, 29, 35]. While the actual algorithms for extracting keys from the observed signals vary, all these techniques have the same fundamental usage model and signal processing pipeline as illustrated in Figure 2.

The key generation process in Figure 2 starts with one of the devices requesting the other to begin the process, which is confirmed by the other device with a response. At this point, both devices collect a time-series of the common signal and preprocess the data to remove noise as well as mark the start of the series to achieve synchronization between the two independently collected time series. For example, in gait-based key generation, some form of frequency filters are often employed to remove non-gait-related body vibrations, while synchronization is achieved by detecting clearly observable events, such as heel strike.

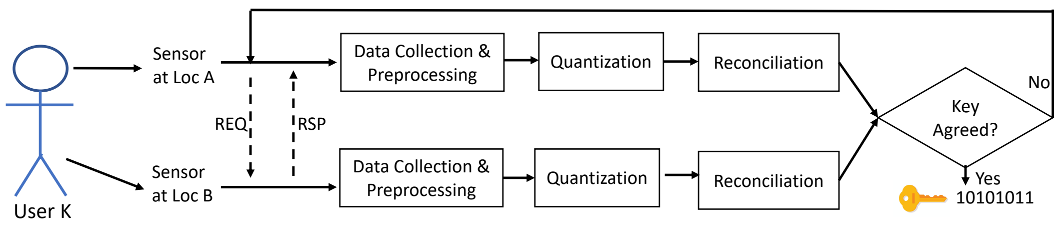


Fig. 2. Signal processing pipeline of existing autonomic symmetric key generation systems

A *quantization* algorithm is then applied to the processed time series to convert each time series data point to a binary digit, i.e., bit '0' or '1'. While most quantization algorithms [25, 29] use statistical gait features to generate the bits, Walkie-Talkie [35] exploits the shape of the gait curve to achieve better matching between the independently generated bit sequences [6].

In the quantization of Walkie-Talkie, the time series is first normalized to have a zero mean and unit length, i.e., all data points stay between -1 and 1. A guard band is then defined as the region between α and $-\alpha$ to convert each data point value, v, according to the following quantizer $\Theta(\cdot)$, which converts all values greater than α to '1', while values less than α are converted to '0' and values that fall within the guard band are simply discarded:

$$\Theta(v) = \begin{cases} 1, & v > \alpha \\ 0, & v < -\alpha \\ \textit{Discarded}, & -\alpha \le v \le \alpha \end{cases} \tag{1}$$

As the quantization is performed on two independently generated non-identical time series, the generated bit sequences exhibit minor mismatches. The reconciliation algorithm allows the pair of devices to exchange some specific information about the mismatch in an attempt to correct them. The fully matched symmetric keys are successfully generated if the reconciliation is successful; otherwise, the key generation process starts over and loops through the fundamental steps of data collection and preprocessing, quantization, and reconciliation, until a symmetric key is generated successfully.

For the quantizer of in Eq. (1), the speed of key generation is sensitive to the choice of α and there exists an optimum α that maximizes it. If α is too small, time series samples are hardly discarded at the expense of a high mismatch rate between the generated bit sequences, which forces the key generation process to loop through the process many times before successfully obtain a fully matched symmetric key. On the other hand, although the bit mismatch could be reduced by choosing a large α, it would discard a large number of data samples increasing the data collection time that would be required to obtain all the bits of a symmetric key.

To speed up key generation, previous research [25, 29, 35] mainly focused on the design and optimization of the quantization algorithms. In contrast, in Auto-Key, we propose to accelerate the key generation by improving the similarity of the two signals used by the quantizer. More specifically, we apply ML to predict sensor data at one body location from the data collected at another location, thus producing two time series that are more similar to each other compared to the raw time series. Our method, therefore, can be adopted as value-added processing in the existing key generation pipeline to further reduce the end-to-end key generation time.

3 PROPOSED MACHINE LEARNING FRAMEWORK IN AUTO-KEY

ML is the key feature of Auto-Key. which is used during key generation to improve the matching between the two time series independently used by two body sensors to produce the symmetric key. Auto-Key uses a two-tier

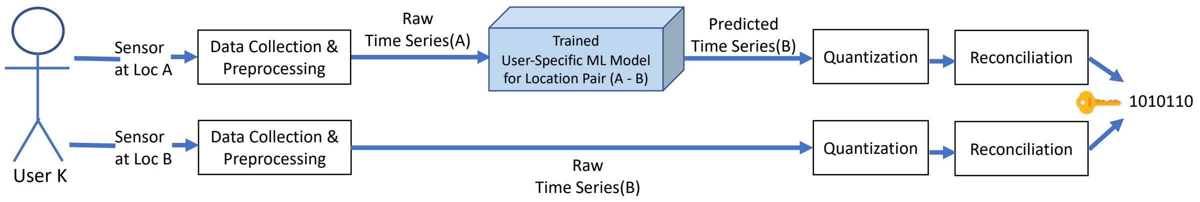


Fig. 3. Machine learning augmented key generation pipeline of Auto-key

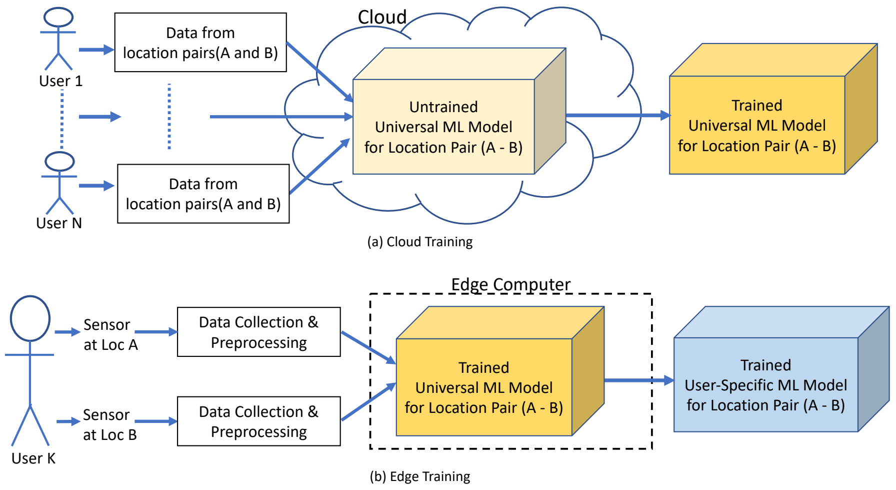

Cloud
User 1
Data from location pairs(A and B)
User N
Data from location pairs(A and B)
Untrained Universal ML Model for Location Pair (A - B)
Trained Universal ML Model for Location Pair (A - B)
(a) Cloud Training
Edge Computer
User K
Sensor at Loc A
Sensor at Loc B
Data Collection & Preprocessing
Data Collection & Preprocessing
Trained Universal ML Model for Location Pair (A - B)
Trained User-Specific ML Model for Location Pair (A - B)
(b) Edge Training

Fig. 4. The two-tier cloud-edge training framework used in Auto-key. The universal model is trained in the cloud (a), which is retrained for a specific user with minimal sample collection and training time in a private edge computer (b).

cloud-edge training framework to address the training overhead as well as the privacy concerns associated with cloud-based training.

3.1 ML-augmented Key Generation

We propose to augment the existing key generation pipeline with ML as illustrated in Figure 3. As we can see, for a given sensor/location pair, the pipeline for only one location is affected, while the other location continues to use the existing pipeline. Basically, for one location (location A in Figure 3), Auto-Key uses a pre-trained ML model to predict the time series of location B before it is fed to the quantizer. Thus for location A, the quantizer now works on the predicted time series instead of the raw time series, while the raw time series is quantized at location B. Assuming that the ML model can accurately predict the time series of B, we can expect a high matching probability between the two bit sequences after quantization, leading to successful and accelerated key generation without having to repeat the pipeline.

3.2 ML Model and Training

Training an ML model to observe sensor time series at one location and predict the corresponding time series at another body location is not only challenging in terms of designing the model, but the training itself has high user overhead in terms of training data collection and training computation time. Although the high training computation overhead could be technically addressed by training the model in the cloud, handing over private body-centric data to third party cloud operators raises privacy concerns [2, 16]. Auto-Key addresses the overhead and privacy challenges with a **two-tier cloud-edge training framework** as illustrated in Figure 4. In the two-tier framework, the manufacturer of wearable sensors trains a universal model in the cloud by using data from a large number of users (i.e., population samples). A specific user downloads such a universal model from the manufacturer's site and retrains it with only minimal sample collection and training time in a private edge device.

We refer to the proposed ML framework as Auto-Key because an autoencoder is used to design the universal model. To reduce the required sample size and training time for the edge-based retraining, Auto-Key applies the well-known concept of transfer learning on the universal autoencoder. The design of the autoencoder and the applied transfer learning are explained next.

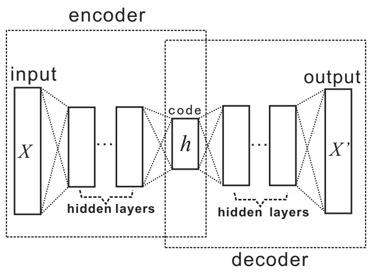


Fig. 5. Structure of a typical autoencoder

3.2.1 Autoencoder Design. Autoencoder is a type of neural networks typically used to learn a "compressed" representation for a set of data [4, 8, 10, 19]. As shown in Figure 5, an autoencoder consists of two components, encoder and decoder, each of which can have an equal number of hidden layers. The encoder learns to compress the input vector X to a shorter code h, while the decoder learns to reconstruct the input vector X by uncompressing h. The encoder and the decoder, respectively, are defined by two functions f and h as: $f(X) = h$ and $g(h) = X'$. A typical autoencoder aims to solve the loss function as follows:

$$\underset{f,g}{\arg\min} \quad \|X - g(f(X))\|^2. \tag{2}$$

Denoising autoencoder is a type of autoencoder that tries to reconstruct a clean "repaired" input from a corrupted one by learning to filter out the noise during its training [8, 32, 33]. Recent work has shown the surprising advantage of corrupting the input of autoencoders on pattern classification, speech recognition, and word representation [4, 19, 32]. In those research works, by corrupting the input, denoising autoencoders can extract robust features as code h. Suppose that the initial input is X, and the corrupted input is $\bar{X}$ created by noise s (i.e., $\bar{X} = X + s$). Then the loss function in Eq. (2) becomes:

$$\underset{f,g}{\arg\min} \quad \|X - g(f(\bar{X}))\|^2. \tag{3}$$

This loss function enables the denoising autoencoder to discard the noise s in the initial input.

When we have two sensors at two body locations producing two time series, one of them could be assumed as a corrupted version of the other. This assumption holds because both sensors are supposed to observe the identical body signal, such as the gait. Thus, we can use a denoising autoencoder to remove the "noise" from one of the time series by solving the loss function of Eq (3). The structure of the proposed denoising autoencoder is

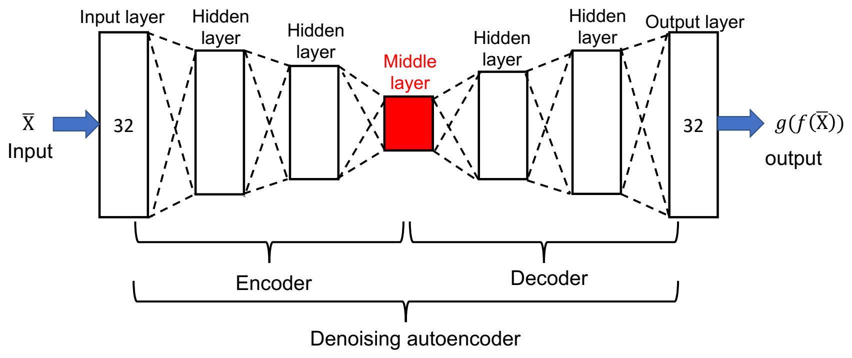


Fig. 6. Structure of the proposed denosing autoencoder

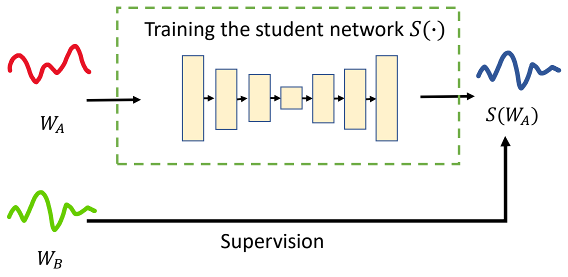


Fig. 7. Proposed teacher-student network for training the denoising autoencoder

shown in Figure 6. It has symmetric layering around the middle layer (in red), which produces the short code h. Both the input and the output layers have a dimension of 32, but the optimum dimensions for the hidden layers can be selected for different applications via experiments, as we have done in Section 4 for gait-based key generation. All layers are fully connected.

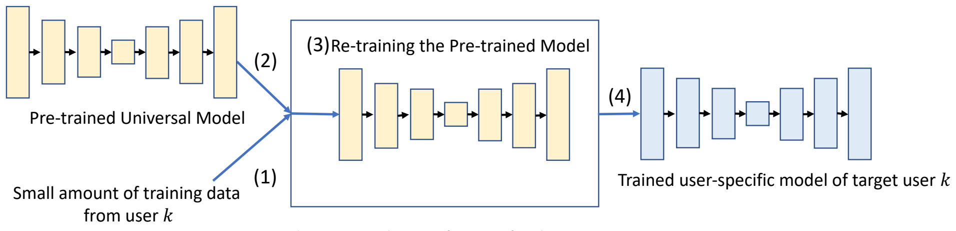
Next, we propose the Teacher-Student network in Figure 7 to train the denoising autoencoder. Let us consider two preprocessed time series windows, W_A at location A and W_B at location B. If our objective is to predict series W_B from W_A, we use W_B as the label of W_A. Here, W_B is the teacher providing supervision for training the denoising autoencoder that can be considered as the student network $\mathbf{S}(\cdot)$, which learns to predict W_B by taking W_A as input.

The training objective of the student network $\mathbf{S}(\cdot)$ is to minimize the difference between the student network's prediction $\mathbf{S}(W_A)$ and the input W_B as:

$$\arg\min_{\mathbf{S}} \mathbf{L}(W_B, \mathbf{S}(W_A)). \tag{4}$$

We define the loss as Mean Square Error (MSE) for every value in the input time series window and the corresponding value in the label windows as follows:

$$\mathbf{L}(W, W^p) = \frac{\sum_{i=1}^{n}(W_i - W_i^p)^2}{n}, \tag{5}$$


(3)Re-training the Pre-trained Model
(2)
Pre-trained Universal Model
(4)
(1)
Small amount of training data from user k
Trained user-specific model of target user k
Fig. 8. The procedure of transfer learning in Auto-Key

where $n = 32$ denotes the training window size, and w_i and w_i^p denote the i-th value in W_A and $\mathbf{S}(W_B)$ respectively.

3.2.2 Transfer Learning. In Auto-Key, we apply the concept of transfer learning [23, 31] to reduce the required training data, resource and computation time for training a user-specific autoencoder model. Such reductions make Auto-Key training more practical for edge computing, which provides better privacy protection to personal gait data compared to uploading them to third-party cloud servers [28].

We use the transfer learning with a technique called *fine-tuning transfer learning* [36], which is illustrated in Figure 8. The steps include: (1) target user k sends a small amount of training data to an edge device; (2) the edge device downloads the universal pre-trained autoencoder model produced from a large population samples in a central cloud server; (3) the edge device retrains all the layers in the pre-trained universal model; (4) The trained model for target user k (i.e. user specific model) is downloaded to k's wearable devices for use in key generation pipeline as shown in Figure 3.

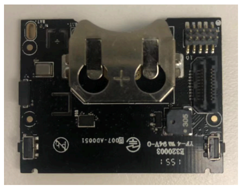
4 EVALUATION

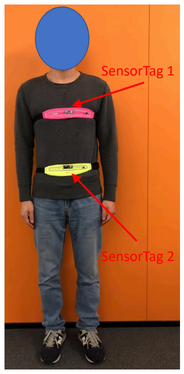
The detailed implementation of the key generation procedure used in the evaluation and security analysis in Section 5 is presented in Appendix A.

4.1 Goals, Metrics, and Methodology

The goals of the evaluation are twofold: 1) to evaluate the effects of different components of Auto-Key including autoencoder and transfer learning; 2) to evaluate the effects of different body locations including chest, waist, forearm, head, shin, and thigh.

4.1.1 Data Collection. We use the public dataset in [30] for position-aware activity recognition to evaluate the impacts of key generation performance of proposed autoencoder in different body locations. In this dataset, 15 subjects (8 males and 7 females, age 31.9±12.4, height 173.1±6.9, weight 74.1±13.8) performed different activities for approximately 10 minutes. They wore 7 types of sensors on 7 different body locations including chest, waist, forearm, head, shin, thigh and upper arm. We use the accelerometer data measured at 6 body locations (except for forearm) of the walking activity of all 15 subjects. The data of forearms is excluded because we find that the subjects frequently perform random motions with their forearms during walks which make the signals from forearms do not correlate to other signals. For example. the average Pearson correlation between signals from the forearm and chest is only about 0.0178 for the first 5 subjects in the public dataset, which indicates that the signals of forearm and chest are basically not correlated. The core idea of using autoencoder is to let the neural network to capture the internal correlations among signals from different body locations. If the two signals are not correlated, the autoencoder can not help the key paring process. The sampling frequency in this dataset is 50 Hz.


(a) SensorTag



(b) Body Locations of two SensorTags

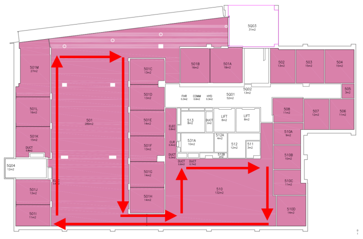

(c) In-house Walking Path

Fig. 9. In-house data collection

We collect an in-house dataset[1] to analyze the effect of transfer learning. Namely, 1) what are training data sizes that we can reduce by exploiting the universal model trained in the public dataset discussed earlier; 2) how much less training time in the edge devices that we can reduce by exploiting the universal model. For the in-house dataset, we recruited 6 subjects. As shown in Figure 9, we use SensorTags[2] shown in Figure 9(a) to collect accelerometer data on the subjects. The subjects are asked to wear 2 SensorTags on the chest and waist, respectively, and walk along an 88-meter indoor path shown in Figure 9(c) for approximately 10 minutes at their normal speed. The sampling rates of the SensorTags are 50 Hz and we collected 30,000 accelerometer samples from each subject.

4.1.2 Metrics. For a shared key generation protocol, we focus on the following two evaluation metrics:

(1) **Bit Agreement Rate (BAR)**: this represents the percentage of bits matching between two keys generated by two devices respectively, and is calculated as:

$$\text{BAR} = \frac{\text{The number of match bits in two keys}}{\text{The total number of bits in the key}} \times 100\%. \tag{6}$$

This metric evaluates the potential of two legitimate devices (i.e., Alice and Bob) agreeing on a same key.

(2) **Key Generation Rate (KGR)**: this represents the average number of agreed keys generated from the acceleration samples per minute. In the proposed Auto-Key, the key length is set as 128 bits. Therefore, KGR is defined as:

$$\text{KGR} = \frac{\text{The number of generated keys}}{M}. \tag{7}$$

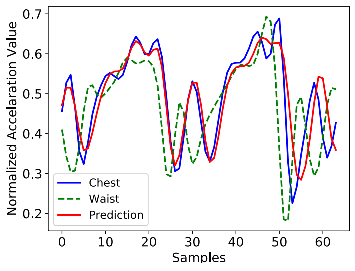
where M is the number of minutes. This metric evaluates how fast two devices on different body locations can generate secret keys.

The results are presented for the average values and 95% confidence levels of 10-fold cross-validation.

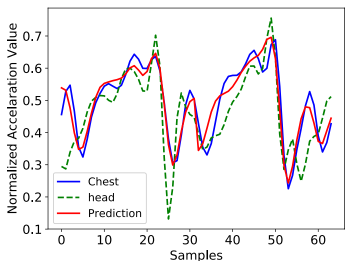
[1]Data collection involving human subjects has been approved by the ethics committee of the anonymous organization (HC17008).
[2]SensorTag:http://www.ti.com/ww/en/wirelessconnectivity/sensorta/2015/index.html

Table 1. Different dimension settings of autoencoder models (public dataset). We use signal observed from other body locations to predict that from chest.

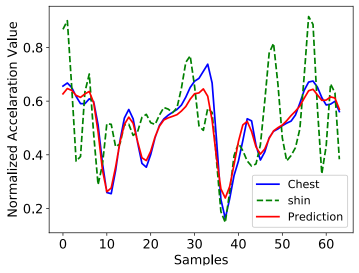
model name	model setting	MSE (Prediction)				
		head	upperarm	waist	thigh	shin
Model 1	32-16-8-4-8-16-32	0.00285	0.00336	0.00490	0.00370	0.00527
Model 2	32-32-16-8-16-32-32	**0.00244**	0.00286	0.00445	**0.00338**	**0.00526**
Model 3	32-32-32-16-32-32-32	0.0026	**0.00282**	**0.00444**	0.00352	0.00540
Model 4	32-32-32-32-32-32-32	0.0027	0.00334	0.00447	0.00394	0.00540
SVR	the kernel is RBF kernel	0.03336	0.05288	0.06965	0.06772	0.09622



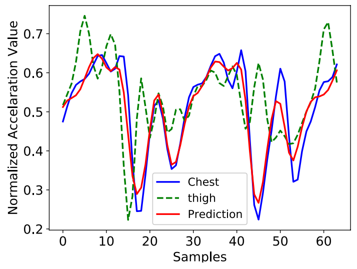
(a) chest - waist



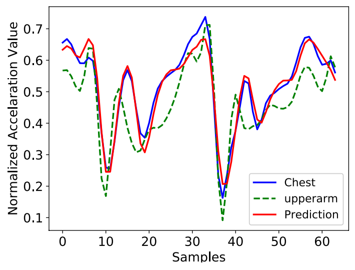
(b) chest - head



(c) chest - shin



(d) chest - thigh



(e) chest - upper arm

Fig. 10. The prediction signals vs. raw signals (public dataset)

4.1.3 The Implementation of Autoencoder. We implement the autoencoder using Keras [9] framework with TensorFlow [1] back-end. The autoencoder is trained with a batch size of 32. We use 7 layers totally in our autoencoder. All the layers of the autoencoder are fully connected. The encoder consists of 3 dense layers and the input dimension is 32, while the decoder also consists of 3 dense layers and the output dimension is also 32. We have tried different dimensions of hidden layers as shown in Table 1. The number denoted as red is the middle hidden layer which produces the short code h. On the left side of the middle layer, it is the dimension settings of the encoder. On the right side of the middle layer, it is the dimension settings of the decoder. We use MSE as the loss function. The training is stopped when no improvement can be found in 1,000 epochs. For comparison, we also implemented a popular conventional ML method, e.g. Support Vector Regression (SVR, with Radial Basis Function or RBF kernel), as the baseline.

4.2 The Impact of Autoencoder

Table 1 presents the MSE values of test dataset for *one subject* randomly selected in public dataset to compare the performance of different settings of the autoencoder models and select the best one for the rest of the paper. We train models for using acceleration signals observed at the head, waist, upper arm, thigh and thin to predict those observed at the chest. Thus, there are five pairs here: (head - chest), (upper arm - chest), (waist - chest), (thigh - chest) and (shin - chest).

Table 1 shows that all autoencoder models produce significantly better MSEs than those of SVR on all body locations, which indicates that the neural network based autoencoder model has superior performance than traditional ML such as SVR in signal prediction on different body locations. Furthermore, it can be seen that all models can reach small MSEs after training. Among them, Model 2 achieves the smallest MSEs when pairing chest device with those at the head, thigh, and shin, while Model 3 achieves the smallest MSEs when pairing chest device with those at upper arm and waist. In contrast, Model 1 produces the largest MSEs when pairing chest device with those at the head, upper arm, and waist, while Model 4 produces the largest MSEs when pairing chest device with those at the thigh and thin. The results indicate that choosing a compression level (i.e., the length of short code h) is important. If we compress the input too much (e.g., Model 1) or too little (Model 4), the autoencoder cannot learn to produce h as good quality as Models 2 and 3 to predict the signals observed at the chest.

The differences between Models 2 and 3 are insignificant when pairing chest device with those at the waist and upper arm, but Model 2 performs significantly better than Model 3 at the other three locations. Therefore, we choose Model 2 as our denoising autoencoder for the rest of the paper.

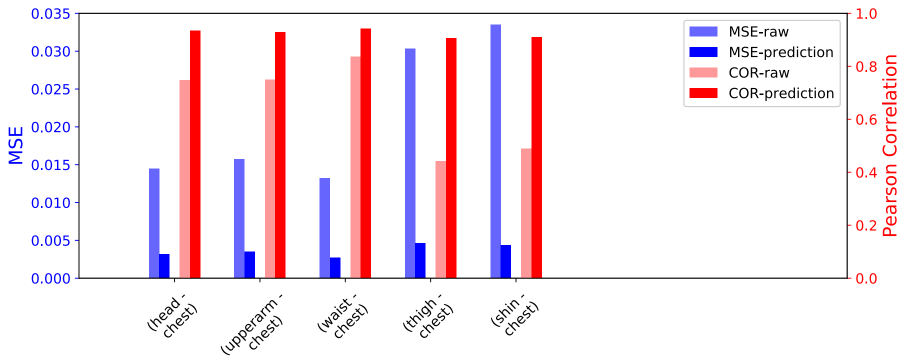
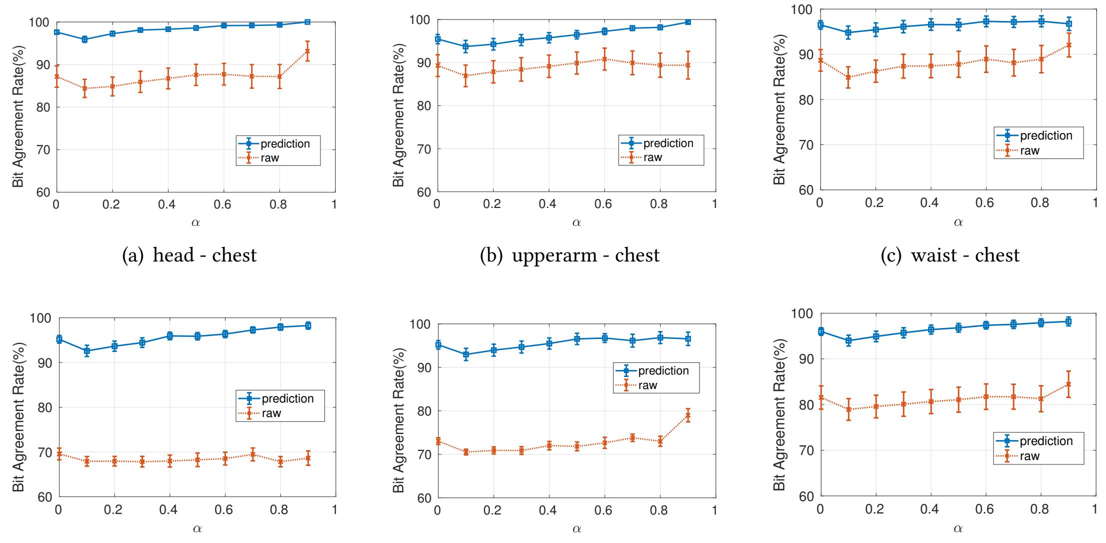


Fig. 11. MSE and Pearson correlation (public dataset)

In Figure 10, we plot the predicted signals against raw signals for some randomly chosen samples. The blue signals are measured from devices at the chest. The green signals are measured from devices at the other body locations (e.g., head, upper arm, waist, shin, and thigh). The red signals are the predicted signals observed at the chest when taking signals observed at the other body locations as input. It can be clearly seen that the predicted signals are significantly closer to those observed from the chest device than the raw signals observed at other body locations. Nevertheless, there are still some small differences between the predicted signal and the signals observed at the chest, which may produce some mismatched bits in the generated keys and the reconciliation component (see Appendix A.5.1) is used to correct them.

To quantify the relationship between predicted and raw signals (ground truth). We calculate the Pearson correlation coefficients [5] between them. The coefficients are in the range of [-1, 1], where 1 is a totally positive



(a) head - chest (b) upperarm - chest (c) waist - chest

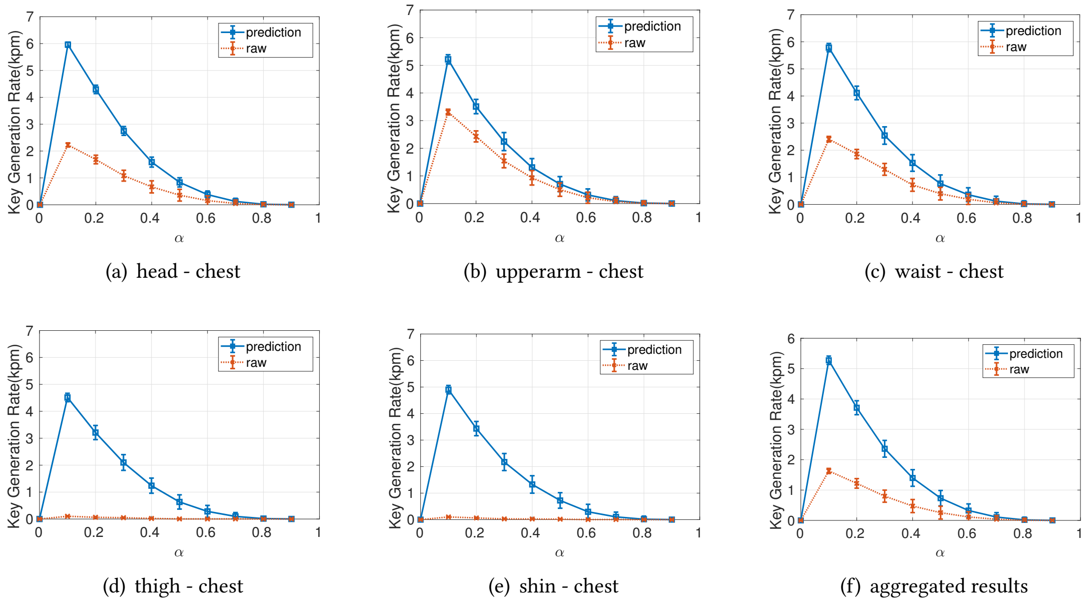
(d) thigh - chest (e) shin - chest (f) aggregated results

Fig. 12. Bit agreement rates of pairing the signals observed at 5 different body locations (i.e., waist, head, shin, thigh and upperarm) with those observed from chest and the aggregated/average results. Public dataset.

linear correlation, -1 is a totally negative linear correlation, and 0 is no linear correlation. A high correlation coefficient means two acceleration sequences have a strong linear correlation and thus may be used to generate keys. A low correlation means two acceleration sequences are not correlated. If we use them to generate keys, the generated keys may have many mismatched bits. In Figure 11, it can be seen that the correlation coefficients between the predicted signals and the ground truth raw signals are significantly higher than those between the raw signals observed at two body locations. This indicates that the proposed autoencoder can improve the correlation between two acceleration sequences by prediction. The signals observed at the locations of the head, upper arm and waist have a higher correlation with those observed at chest because they are on the body trunk and are close to the chest. The signals observed at the other locations, i.e., thigh and shin, are on the limbs that have lower correlations with those observed at the chest. The motions of limbs can add significantly more noise to the measurements from the thigh and shin. We also plot MSE in Figure 11, which draws a similar conclusion. Namely, for all pairs, autoencoder can significantly reduce the MSE of 2 acceleration sequences by prediction. Locations on the body truck produce smaller MSE than those on the limbs.

Figure 12 plots the BAR of different pairs at different body locations. The blue line is the BAR of the keys generated by using the predicted and raw signals, and the red line is the BAR of the keys generated by using the raw signals at two different body locations (without prediction). It can be clearly seen that BARs of all 5 pairs have been improved significantly for all α values by prediction. For the locations on the body truck such as waist, head, and upper arm, the improvement of BAR is smaller than those locations on the limbs such as shin and thigh because the BARs produced by raw signals are already high (e.g., 85%) for the locations on the body truck. On average, the proposed method based on autoencoder can improve BAR by 16.5% (Figure 13(f)).

In Figure 13, the blue line represents the KGR of using the prediction and raw signals, while the red line represents the KGR of using raw signals observed at two body locations directly. Similar to Figure 13, this figure



(a) head - chest (b) upperarm - chest (c) waist - chest

(d) thigh - chest (e) shin - chest (f) aggregated results

Fig. 13. Key generation rates of pairing the signals observed at 5 different body locations (i.e., waist, head, shin, thigh and upperarm) with those observed from chest and the aggregated/average results. Public dataset.

shows that KGR can be significantly improved by using prediction instead of raw signals. Figure 13(f) plots the average values among all body locations, and it shows that the value of α has a high impact of the KGR improvement because a larger alpha value produces a higher BAR (see Figure 12) at the cost of discarding much more samples. Figure 13 also shows that the largest improvement was produced when α is 0.1. The KGR increases from 1.8 to 5.2 keys per minute (kpm) on average. Similar to Figure 12, we can see that for those locations on the body trunk (e.g., waist, head, and upper arm), the improvement in KGR is relatively smaller than those on the limbs (e.g., shin and thigh). We note that the KGRs of pairing the raw signals observed at the chest with those observed at the thigh or shin directly is very close to 0 (the key generation protocol almost does not work!). This is because the gait signals are highly corrupted by the noise at those locations. However, with the proposed autoencoder, the KGRs has been improved to 4.7 and 4.9 kpm, respectively, which represents more than 25 times improvement.

In summary, our results show that: 1) the denoising autoencoder can remove the noise among the gait signals observed at different body locations and predict the signal observed at one location by using those observed at another location. 2) By using the predicted gait signals to generate a key, the BAR and KGR can be significantly improved by 16.5% and more than 1.9 times, respectively. 3) The optimal value of the guard band ratio (α) is 0.1 to produce the fastest KGR.

4.3 The Impact of Transfer Learning

In this section, we first train universal autoencoder models by using the public dataset. The universal models are trained on a server with a GeForce RTX 2080 Ti graphics card from NVIDIA[3]. Then, we download the universal

[3]https://www.nvidia.com/en-us/geforce/graphics-cards/rtx-2080-ti/

models to a common desktop computer with an Intel Core i7-8700 CPU[4] and without graphic cards, which is served as an edge device. Finally, for each subject in our in-house dataset, we have 30,000 acceleration samples, which are divided into 9 parts (8 for training and one for testing). For each subject, we use different percentages of training data (see in Table 2) to retrain the universal model in the edge device to obtain subject-specific models. For comparison, for each subject, we also produce models from scratch (without transfer learning) using all training data and use them as the baseline models.

Table 2. The effects of different train data sizes to the performance of proposed transfer learning

Training data size	MSE(e^{-03})	BAR	KGR	Training Epochs	Training Time (s)
10%	4.71	90.24	2.40	72	0.86
30%	3.48	94.59	3.86	225	6.80
50%	2.84	96.52	4.69	712	32.86
70%	2.67	96.82	4.90	543	32.96
90%	2.61	97.38	5.14	511	38.67
100%	2.52	97.65	5.13	692	64.25
baseline(937 samples)	2.80	97.07	5.01	3,646	276.77

In Table 2, we compare the MSE on test data, BAR, KGR, epochs and training time when using different percentages of training samples. The results are presented for the average values of 6 subjects. Retraining a universal model via transfer learning has three benefits as follows. 1) It has a lower MSE starting point than training from scratch (see Figure 14 for an example). 2) It reduces training epochs (time). For example, when using 50% training samples, the training finishes in 32.86 seconds, and it takes training from scratch more than 276 seconds (approximately 88% reduction). 3) It reduces training data sizes. For example, when using 50% training samples(468 samples), the training reaches a small MSE which is comparable with that of baseline using all training samples(947 samples).

To balance the training overhead and performance, Auto-Key recommends reducing the training data size by 50%. In such the setting, the BAR and KGR are 96.52% and 4.69 kpm, respectively. So the minimum training dataset for Auto-Key has 468 samples. Since the window size of each sample is 30 and the sampling frequency is 50 Hz, it will take a 5-minute walk for a user to generate 468 samples needed for the training dataset. According to [22], the frequency of human walking steps is between 1.8 Hz and 2.8 Hz. Therefore, it requires the user to walk between 540 and 840 steps. In [3], the average steps that a person walks per day are 3,400 during weekdays and 3,500 during weekends respectively in the USA. Therefore, the user may produce the training dataset *passively* with less than 30% of her daily walk. Figure 14 plots the test MSE of one randomly selected in-house dataset subject over time.

4.4 The Impact of Location Variance

We investigate the impact of location variance on the performance of Auto-Key since there may be small variance when a user wears a wearable device on the same body location at different times. In this experiment, we collected data from a subject in the middle of her waist and chest respectively and trained a model to predict her chest's signal by using her waist's signal. Then we collected another set of data by moving the device on chest 1 cm, 3 cm, 5 cm away from the original spot respectively while keeping the other device in the original spot of the waist. Table 3 shows the performance of Auto-Key decreases with the increasing of deviation, unsurprisingly.

[4]https://ark.intel.com/content/www/us/en/ark/products/126686/intel-core-i7-8700-processor-12m-cache-up-to-4-60-ghz.html

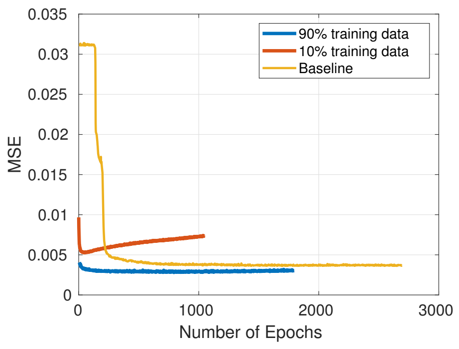


Fig. 14. The test MSE of one randomly selected in-house dataset subject over time.

However, even the deviation distance is 5 cm, the BAR is 85.3% which is similar to that of the baseline without Auto-Key prediction. We note that 5 cm represents a large distance deviation for the same wearable device worn on the same body location, which rarely happens in reality. If the deviation distance is larger than 5 cm, the device should be considered in a new body location and retrain a new autoencoder for the prediction.

Table 3. The impact of location variance

Metric	Baseline1 using raw data (no deviation)	Baseline2 with model (no deviation)	Device on chest deviated by		
			1 cm	3 cm	5 cm
BAR(a = 0.1)	86.1%	94.9%	90.5%	88.6%	85.3%

4.5 The Overhead of Auto-Key Training and Inference

In this section, we will analyze the computational overhead of Auto-Key.

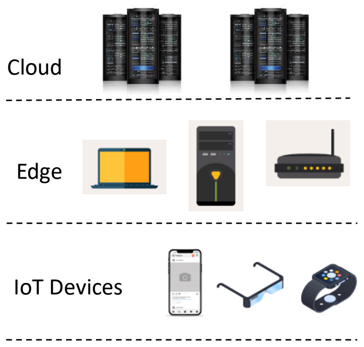


Fig. 15. An example architecture of edge computing

Figure 15 shows a popular architecture of edge computing, which consists of three levels as in [15]. The devices in BAN are wearables and mobile devices which can be considered in the third level "Internet of Things (IoT) devices". Devices at the "Edge" are defined as edge devices or edge computers. Although there are different types of edge devices depending on the application scenarios, the training of Auto-Key is operated in a smart home scenario. So the edge devices can be a desktop and laptop computer owned by the user. When the training starts, the desktop computer first downloads the universal model from the manufacture's cloud server and collect train data from user's mobile devices. Then the model is re-trained on the desktop computer. Note that sensitive personal biometric (gait) data are stored (and computed) in the edge devices owned by the users instead of the public cloud to protect her privacy. After the model is re-trained, the user's mobile devices may download it. Therefore, we use a typical desktop with Intel Core i7-8700 CPU as our edge device prototype. Directly training in IoT devices is also another choice, though the limited computational capabilities of current IoT devices are the bottleneck. Table 4 shows the training time of two different devices. It can be clearly seen that training on a desktop is approximately 8 times faster than a Raspberry Pi 3. Therefore, it is more appropriate for the re-training tasks to be run in the edge devices instead of in the IoT devices directly to significantly reduce training time and provide a better user experience.

Table 4. Training times on different devices

Training devices	Average training time per model in 712 epochs (ms)
A desktop computer with a Intel Core i7-8700 CPU	32.86
A Raspberry Pi 3 Model B+ with a 64-bits quad-core ARMv8 CPU	259.12

The inference is another overhead of the Auto-Key key generation process. Table 5 shows the average inference times for a 128-bit key on different devices. It can be seen that inference overhead is insignificant compared to that of the training. The inference of the model on a Raspberry Pi takes only 0.64 ms which is 2.7 times slower than the desktop computer. The inference time in an iPhone X increases to 26.58 ms, which is still small.

Table 5. Inference times on different devices

Inference devices	Average inference time for a 128 bits key (ms)
A desktop computer with a Intel Core i7-8700 CPU	0.23
A Raspberry Pi 3 Model B+ with a 64-bits quad-core ARMv8 CPU	0.64
Apple iPhone X	26.58

5 SECURITY ANALYSIS

5.1 Trust Model

Similar to previous work [6, 25, 29, 35], we assume that all devices that can measure the motion signals of a user directly are legitimate, which means all the devices worn on the user's body are trustworthy. The user is assumed to be capable of detecting such attacks if an attacker tries to place an adversary device or modify the legitimate devices on the user's body so that the attackers have no access to the legitimate devices and their data.

5.2 Adversary Model

We assume that attackers attempt to breach Auto-Key by violating confidentiality and compromising integrity while ensuring the availability is out if the scope of this paper.

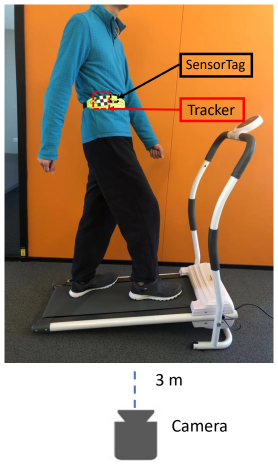


Fig. 16. Video side-channel attack data collection

We assume that the gait signal is private information that cannot be obtained by an attacker, who may use it to train a similar neural network. If the attacker can somehow obtain the gait signal of a user(e.g. by hacking a wearable device), then she can use it to pair with the legitimate device directly. Preventing hacking the legitimate device is out of the scope of this paper. Then, we assume that an attacker (i.e., Eve) has the full knowledge of the key generation mechanism in Appendix A and has full control of the communication channel except for jamming the channel. Therefore, Eve may eavesdrop the message between Alice and Bob, and impersonate a legitimate device to derive the secret key that secures the communication channels in BAN. We also assume that Eve can eavesdrop the message transmitted via the public channel in the reconciliation step (see Appendix A.5.1). So, Eve can derive a key from various means and she can also use reconciliation to correct the mismatched bits for this key. We assume that the attackers may have the access to the universal autoencoder model trained with the population dataset, but don't have the access to user-specific transfer-trained models because the user-specific models are always kept confidentially in the users' (edge and IoT) devices.

We analyze three types of attacks in this paper similar to recent work [6]:

(1) **Video Side-Channel attacks**. Recent works have shown that video-based acceleration signals extraction methods can accurately extract accelerations signals from the video that records the walking user [39]. We assume that Eve may use this approach to extract acceleration information and use the extracted acceleration information to generate keys for either Man In The Middle (MITM) or spoofing attacks.
(2) **Active Mimicking attacks**. Eve can observe and study the walking style of the genuine user. She generates a key by mimicking the walking style of the genuine user and use this key to pair with genuine user's devices.
(3) **Passive attacks**. Eve generates a key based on her own gait signals and uses this key to pair with the genuine user's devices.

In the experiments, we recruited 6 subjects to collect data and conduct experiments as below:

(1) For legitimate device, each subject is wearing two SensorTags on chest and waist as in Figure 9(b). For each subject, we trained a model to use data observed at the chest to predict data observed at the waist. We use the waist's prediction based on the data observed at the chest to pair with the data observed at the waist.

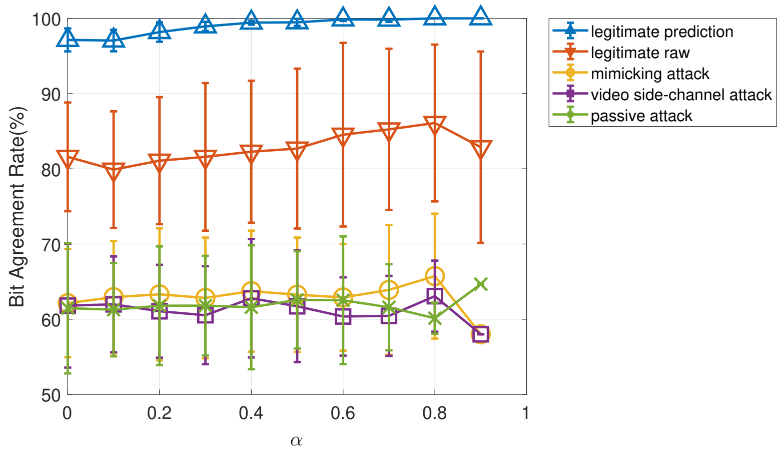


Fig. 17. Three types of attacks directly on same locations

(2) For side-channel video attacks, the data collection setup can be seen in Figure 16. Each subject wears a Sensortag and a marker (for vision-based tracking) next to the Sensortag on their waist to collect the acceleration data since it is reported in [6] that the waist is one of the most vulnerable locations for the side-channel video attacks. The subjects are asked to walking on a treadmill for 50 steps. In the meantime, a slow-motion camera at 60 frames per second films the walking and tracks the marker worn on the subject's waist. Then Eve will extract estimated acceleration data via two-times differentiation from the video using computer vision-based techniques such as [39] and then try to use the extracted acceleration data to generate keys to pair with the device (i.e., SensorTag) next to the marker.
(3) For passive attacks, we grouped 6 subjects into 3 pairs. Each subject attaches a SensorTag at their waist. All subjects are asked to walk freely for 1 minute in their own walking style. Then two subjects in pair use their own acceleration data to try to pair with each other.
(4) For active mimicking attacks, we grouped 6 subjects into 3 pairs like the passive attacks above. Each subject attaches a SensorTag at their waist. Each participant is asked to be an active attacker to mimic her partner's walking style and try to imitate it. Then the attacker uses her acceleration signals to pair with her partner's device.
(5) Since above attacks scenarios show that active mimicking attack is the most powerful one with the highest BAR (see Figure 17 for more details), we use the active mimicking attack to investigate whether the universal autoencoder model can be exploited by the attackers to increase attack performance (i.e., BAR). To this end, we grouped 6 subjects into 3 pairs similar to the active mimicking attacks discussed above. Each subject wears two Sensortags on her waist and chest, respectively. Each participant is asked to be an active attacker to mimic her partner's walking style and try to imitate it. The mimicked gait data from the waist are input to a trained universal autoencoder model to predict the gait data observed on the chest. Then the mimicking attacker uses the prediction signals to pair with the device on her partner's chest.

Many unrealistic advantages are given to the side-channel video attackers deliberately. For example, the subject is asked to walk on the treadmill so that the camera can have a close and stable observation on the trajectory of the marker worn at the waist and generate "clean" acceleration signals of the subject's gaits; The distance between the treadmill and camera is fixed to 1.5 m, which also make the estimation of trajectory as well as the accelerometer values much easier. Each genuine subject is asked to attach an explicit marker next to her SensorTag so that the feature-based computer vision gait tracking tasks become easier and more accurate. Considering the advantages given to the side-channel video attackers, we anticipate that it will be significantly more difficult for a real-world attacker to obtain similar results as reported in this paper.

In [21], it has been shown that an attacker is unable to mimic the unique gait pattern of a subject to get sufficiently similar data to break the key generation protocol. While in [35], it has been shown that the passive attacker can not succeed either since the gait is unique for each person. However, for completeness, we still collect data to produce the results of these two types of attacks. In Figure 17, the BAR of these two attacks is only slightly above 60% after reconciliation. In comparison, the BAR using (proposed autoencoder) prediction is all more than 96%, which makes it easy to produce a successful key pair for legitimate devices. Our results show that the attacker fails to produce any successful key pairs.

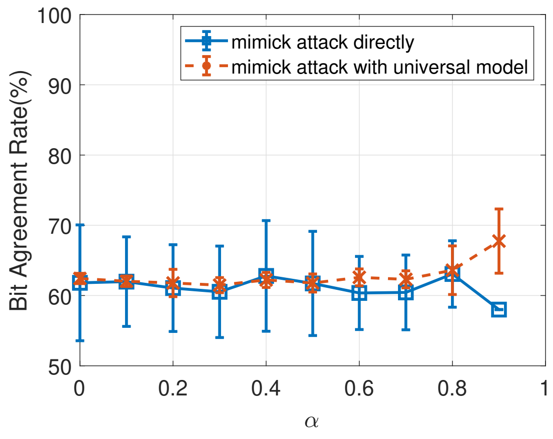


Fig. 18. Attacks with trained population model.

One interesting observation from Figure 17 is that the video side-channel attacks produce approximately 60% BAR, which is worse than these of mimicking attacks. In [6], it has been shown that such a video side-channel attacker may obtain sufficiently accurate gait estimation to generate successful key pairs with a legitimate device. That difference between the results reported in this paper and those in [6] is mainly caused by that we use magnitude values of acceleration readings from three axes in Auto-Key (see Appendix A.2.1), while [6] uses the acceleration data from gravity direction only. Normal cameras cannot accurately produce tracking information in 3D spaces. While vision-based 3D motion tracking systems such as Vicon can produce highly accurate tracking in 3D spaces, they are very expensive (e.g, in the order of US$100,000) and need careful set-up calibration and markers on the subject.

Since Figure 17 shows that mimicking attack is more effective (produces higher BAR) than the other two types of attacks, we further investigate if the attackers can improve mimicking attacks by exploiting the trained population autoencoder model. Figure 18 shows that the BAR is not improved significantly by such enhanced mimicking attacks. Moreover, our studies show that the enhanced mimicking attacks with trained population models also fail to produce any successful key-pairing with legitimate devices. Therefore, Auto-key is resilient to such attacks.

Finally, the proposed Auto-Key increases the differences of BAR between legitimate devices and those of attackers significantly as shown in Figure 17. For example, when $\alpha = 0.1$, the difference is increased from 20% to 36% when we use prediction instead of raw signals, which makes the gait-based key generation systems stronger against different attacks because such difference increases enable us to choose the parameters of reconciliation (see Appendix A.5.1) that have less error correction capabilities. For example, we may choose a smaller number (e.g., 40) instead of 60 chosen in our implementation in Appendix A.5.1. Consequently, it will be difficult for the attackers to pair with legitimate devices, though it may also decrease the KGR between two legitimate devices. Similar to the passive and mimicking attacks discussed earlier, we note that the attacker fails to produce any successful key pairs with side-channel video attacks with our current implementation in Appendix A.

5.3 Randomness of the Final Key

The bits in the final keys should be random for strong security. To validate the randomness of keys generated by Auto-Key, we apply the NIST suite of statistical tests to all the keys generated using both datasets. The NIST statistical tests give the p-values of different random sub-tests. The p-value indicates the probability that the key sequence is generated by a random process. If p-value is less than 1%, the randomness hypothesis is rejected which means the key is not random. The test results of Auto-Key are presented in Table 6, which shows that all the tests have p-values that are greater than 1%.

Table 6. NIST Statistical Test

NIST sub-test	p-value
Frequency	0.082731
FFT Test	0.574149
Longest Run	0.420442
Linear Complexity	0.985606
Block Frequency	0.822883
Cumulative Sums	0.148170
Approximate Entropy	0.886438
Non Overlapping Template	0.889227

6 CONCLUSION

We propose an autoencoder based signal pre-processing step to speed up gait-based key generation. We prove that using acceleration sensor data obtained at one body location to predict the acceleration signal observed at a different body location can be achieved by using autoencoder. We further show that by using the predicted signals can speed up key generation. The bit agreement rate is increased by 16.5% and the key generation rate is increased by more than 1.9X. We further use transfer learning to reduce the required training data by 50% and the required training time by 88% to obtain a user-specific autoencoder model for a new user by retraining a pre-trained universal model. Finally, we also analyze the security of the proposed approach against various attacks.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

[2] Abebe Abeshu and Naveen Chilamkurti. 2018. Deep learning: the frontier for distributed attack detection in fog-to-things computing. *IEEE Communications Magazine* 56, 2 (2018), 169–175.

[3] Tim Althoff, Jennifer L Hicks, Abby C King, Scott L Delp, Jure Leskovec, et al. 2017. Large-scale physical activity data reveal worldwide activity inequality. *Nature* 547, 7663 (2017), 336.

[4] Sarath Chandar AP, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*. 1853–1861.

[5] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. In *Noise reduction in speech processing*. Springer, 1–4.

[6] A. Bruesch, L. Nguyen, D. Schürmann, S. Sigg, and L. C. Wolf. 2019. Security Properties of Gait for Mobile Device Pairing. *IEEE Transactions on Mobile Computing* (2019), 1–1. https://doi.org/10.1109/TMC.2019.2897933

[7] Cory Cornelius and David Kotz. 2011. Recognizing whether sensors are on the same body. In *International Conference on Pervasive Computing*. Springer, 332–349.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[9] Antonio Gulli and Sujit Pal. 2017. *Deep Learning with Keras*. Packt Publishing Ltd.

[10] Chaoqun Hong, Jun Yu, Jian Wan, Dacheng Tao, and Meng Wang. 2015. Multimodal deep autoencoder for human pose recovery. *IEEE Transactions on Image Processing* 24, 12 (2015), 5659–5670.

[11] Chunqiang Hu, Xiuzhen Cheng, Fan Zhang, Dengyuan Wu, Xiaofeng Liao, and Dechang Chen. 2013. OPFKA: Secure and efficient ordered-physiological-feature-based key agreement for wireless body area networks. In *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2274–2282.

[12] Suman Jana, Sriram Nandha Premnath, Mike Clark, Sneha K Kasera, Neal Patwari, and Srikanth V Krishnamurthy. 2009. On the effectiveness of secret key extraction from wireless signal strength in real environments. In *Proceedings of the 15th annual international conference on Mobile computing and networking*. ACM, 321–332.

[13] Ari Juels and Madhu Sudan. 2006. A fuzzy vault scheme. *Designs, Codes and Cryptography* 38, 2 (2006), 237–257.

[14] Darko Kirovski, Mike Sinclair, and David Wilson. 2007. The martini synch. *Technical Report MSR-TR-2007-123, Microsoft Research* (2007).

[15] H. Li, K. Ota, and M. Dong. 2018. Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. *IEEE Network* 32, 1 (Jan 2018), 96–101. https://doi.org/10.1109/MNET.2018.1700202

[16] Ping Li, Jin Li, Zhengan Huang, Tong Li, Chong-Zhi Gao, Siu-Ming Yiu, and Kai Chen. 2017. Multi-key privacy-preserving deep learning in cloud computing. *Future Generation Computer Systems* 74 (2017), 76–85.

[17] Qi Lin, Weitao Xu, Jun Liu, Abdelwahed Khamis, Wen Hu, Mahbub Hassan, and Aruna Seneviratne. 2019. H2B: Heartbeat-based Secret Key Generation Using Piezo Vibration Sensors. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks (IPSN '19)*. ACM, New York, NY, USA, 265–276. https://doi.org/10.1145/3302506.3310406

[18] Yanpei Liu, Stark C Draper, and Akbar M Sayeed. 2012. Exploiting channel diversity in secret key generation from multipath fading randomness. *IEEE Transactions on information forensics and security* 7, 5 (2012), 1484–1497.

[19] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. 2013. Speech enhancement based on deep denoising autoencoder.. In *Interspeech*. 436–440.

[20] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour. 2014. Wireless Body Area Networks: A Survey. *IEEE Communications Surveys Tutorials* 16, 3 (Third 2014), 1658–1686. https://doi.org/10.1109/SURV.2013.121313.00064

[21] M. Muaaz and R. Mayrhofer. 2017. Smartphone-Based Gait Recognition: From Authentication to Imitation. *IEEE Transactions on Mobile Computing* 16, 11 (Nov 2017), 3209–3221. https://doi.org/10.1109/TMC.2017.2686855

[22] M Pat Murray. 1967. Gait as a total pattern of movement: Including a bibliography on gait. *American Journal of Physical Medicine & Rehabilitation* 46, 1 (1967), 290–333.

[23] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22 (2010), 1345–1359.

[24] Masoud Rostami, Ari Juels, and Farinaz Koushanfar. 2013. Heart-to-heart (H2H): authentication for implanted medical devices. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 1099–1112.

[25] D. Schürmann, A. Brüsch, S. Sigg, and L. Wolf. 2017. BANDANA-Body area network device-to-device authentication using natural gAit. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 190–196. https://doi.org/10.1109/PERCOM.2017.7917865

[26] S. Seneviratne, Y. Hu, T. Nguyen, G. Lan, S. Khalifa, K. Thilakarathna, M. Hassan, and A. Seneviratne. 2017. A Survey of Wearable Devices and Challenges. *IEEE Communications Surveys Tutorials* 19, 4 (Fourthquarter 2017), 2573–2620. https://doi.org/10.1109/COMST.2017.2731979

[27] Youssef El Hajj Shehadeh and Dieter Hogrefe. 2015. A survey on secret key generation mechanisms on the physical layer in wireless networks. *Security and Communication Networks* 8, 2 (2015), 332–341.

[28] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. 2016. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal* 3, 5 (Oct 2016), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

[29] Y. Sun, C. Wong, G. Yang, and B. Lo. 2017. Secure key generation using gait features for Body Sensor Networks. In *2017 IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. 206–210. https://doi.org/10.1109/BSN.2017.7936042

[30] T. Sztyler and H. Stuckenschmidt. 2016. On-body localization of wearable devices: An investigation of position-aware activity recognition. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 1–9. https://doi.org/10.1109/PERCOM.

2016.7456521

[31] Lisa Torrey and Jude Shavlik. 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI Global, 242–264.

[32] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. ACM, New York, NY, USA, 1096–1103. https://doi.org/10.1145/1390156.1390294

[33] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.* 11 (Dec. 2010), 3371–3408. http://dl.acm.org/citation.cfm?id=1756006.1953039

[34] Yunchuan Wei, Kai Zeng, and Prasant Mohapatra. 2013. Adaptive wireless channel probing for shared key generation based on PID controller. *IEEE Transactions on Mobile Computing* 12, 9 (2013), 1842–1852.

[35] W. Xu, G. Revadigar, C. Luo, N. Bergmann, and W. Hu. 2016. Walkie-Talkie: Motion-Assisted Automatic Key Generation for Secure On-Body Device Communication. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 1–12. https://doi.org/10.1109/IPSN.2016.7460726

[36] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3320–3328. http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf

[37] Junqing Zhang, Trung Q Duong, Alan Marshall, and Roger Woods. 2016. Key generation from wireless channels: A review. *IEEE Access* 4 (2016), 614–626.

[38] Junxing Zhang, Sneha K Kasera, and Neal Patwari. 2010. Mobility assisted secret key generation using wireless link signatures. In *Infocom, 2010 Proceedings IEEE*. IEEE, 1–5.

[39] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. 2012. Real-Time Compressive Tracking. In *Computer Vision – ECCV 2012*, Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 864–877.

[40] Xiaojun Zhu, Fengyuan Xu, Edmund Novak, Chiu C Tan, Qun Li, and Guihai Chen. 2013. Extracting secret key from wireless link dynamics in vehicular environments. In *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2283–2291.

A IMPLEMENTATION DETAILS OF KEY GENERATION PROTOCOL

A.1 Synchronization

We adopt a heel strike-based method in [35] to temporally align signals. The heel strikes, referring to peaks in gait curves, can be detected and used as an anchor point to eliminate the time offset between two pairing parties.

A.2 Signal Preprocessing

A.2.1 Magnitude. After synchronization, two legitimate devices (i.e., Alice and Bob) begin to sample the acceleration data at the sampling frequency of 50 Hz. Since the devices may be placed in arbitrary rotations, and to reliably generate symmetric key across different devices placed in different rotations, we need a device rotation-invariant basis for the three-axis acceleration data. Therefore, we compute and use the summation of absolute magnitudes of the acceleration instead of the three-axis data individually.

A.2.2 Band pass Filter. According to the study in [25], human motion usually lays below 12 Hz and there is some unexpected correlation between arbitrary acceleration signals in low frequencies (e..g., < 0.5 Hz). Therefore, we apply a band-pass filter of 0.5 Hz - 12 Hz to keep the relevant information in the acceleration signal only.

A.3 Prediction

After signal preprocessing, the acceleration samples are segmented with non-overlapping data entries of 32 samples. Each entry is then input into the trained autoencoder model $\mathbf{S}(\cdot)$ as discussed in Section 3.2. The length of the prediction results is also 32, which will then be concatenated before the quantization step.

A.4 Quantization

Auto-Key applies the guard band-based quantization method in [35] as discussed in Section 2, to quantize the gait samples and encode them into bits.

The default setting for the quantization is: window size = 10 samples, $\alpha = 0.1$. We choose a window size of 10 to increase the randomness of the produced keys as recommended in [6]. α is the guard band ratio can be tuned in $[0, 1)$. When α is 0, no bits will be discarded by a guard band. α can not be 1 as all bits will be discarded. We choose $\alpha = 0.1$ to maximize KGR as discussed in Section 4.

A.5 Information Reconciliation

A.5.1 Reconciliation. Two acceleration signals may consist of the energy from different motions of different body locations and noise. As a result, there may exist mismatched bits in two independently generated keys. So we often get $Key_{Alice} \approx Key_{Bob}$, not $Key_{Alice} = Key_{Bob}$ after quantization.

A typical key generation system usually has an information reconciliation stage allowing two parties to exchange a certain amount of information to correct the mismatch between two parties [27, 37]. The conventional information reconciliation protocols include Cascade [12, 34, 40], and error-correcting code(ECC)-based method [18, 35, 38]. Note that fuzzy vault [13, 25] can be broadly considered as an extension of the ECC-based method. The state-of-the-art information reconciliation protocol is a recently developed Compressive Sensing-based method [17]. By adjusting the compression rate or sensing dimension, the CS-based information reconciliation protocol allows the user to determine the mismatch correction capabilities, making it more flexible than conventional methods. AutoKey adopts the state-of-the-art compressive sensing-based reconciliation method. For the benefit of space, we omit the details of the reconciliation method and the readers are encouraged to refer to [17] for the details. We choose the size of the sensing matrix as 60×128, which represents a compression of more than 50%.

A.5.2 Message Authentication Code. Since Alice and Bob do not share an authenticated channel during the reconciliation procedure, an attacker Eve may modify the messages transmitted between two legitimate devices. To protect the integrity of the message, we adopt the message authentication code (MAC) method to verify the massage y_{Bob} is indeed sent by Bob. Bob transmits $L_{Bob} = (y_{Bob}, MAC(Key_{Bob}, y_{Bob}))$ instead of only y_{Bob}. Upon receiving L_{Bob}, Alice will compute Key^{*}_{Alice} to correct all mismatches. The purpose of the information reconciliation stage is to eliminate all mismatches, so Key^{*}_{Alice} is expected to be the same as Key_{Bob}. Therefore, if Alice obtains $MAC(Key^{*}_{Alice}, y_{bob}) \neq MAC(Key_{Bob}, y_{Bob})$, Alice can conclude that the key pairing is unsuccessful and ask for a new pairing attempt. Since Eve does not know the bits in Key_{bob} and any modification on y_{Bob} will fail the MAC verification. As such, Eve may only block the key pairing by violating the availability, she cannot damage the integrity by modifying the message.

A.5.3 Privacy Amplification. Information reconciliation achieves higher reliability but also reveals partial information to Eve since some information is transmitted over a public channel and can be eavesdropped by Eve. In Auto-Key, we use one of the universal secure hash functions (i.e., SHA2-256) to further increase the randomness of the final keys and resiliency against attacks.

Finally, Bob and Alice got the same final key after privacy amplification. Then the final key can be used by symmetric-key algorithms such as AES-128 to ensure secure communications between Alice and Bob. After privacy amplification, Alice and Bob should produce the same secret keys: $Key^{''}_{Alice} = Key^{''}_{Bob}$. But if the two keys do not match with each other, the key generation will restart from data collection.