TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 10/18 pp105–116
DOI: 10.26599/TST.2021.9010088
Volume 28, Number 1, February 2023

# ETS-TEE: An Energy-Efficient Task Scheduling Strategy in a Mobile Trusted Computing Environment

Hai Wang*, Lu Cai, Xuan Hao, Jie Ren*, and Yuhui Ma

**Abstract:** A trusted execution environment (TEE) is a system-on-chip and CPU system with a wide security solution available on today's Arm application (APP) processors, which dominate the smartphone market. Generally, mobile APPs create a trusted application (TA) in the TEE to process sensitive information, such as payment or message encryption, which is transparent to the APPs running in the rich execution environments (REEs). In detail, the REE and TEE interact and eventually send back the results to the APP in the REE through the interface provided by the TA. Such an operation definitely increases the overhead of mobile APPs. In this paper, we first present a comprehensive analysis of the performance of open-source TEE encrypted text. We then propose a high energy-efficient task scheduling strategy (ETS-TEE). By leveraging the deep learning algorithm, our policy considers the complexity of TA tasks, which are dynamically scheduled between modeling on the local device and offloading to an edge server. We evaluate our approach on Raspberry Pi 3B as the local mobile device and Jetson TX2 as the edge server. The results show that compared with the default scheduling strategy on the local device, our approach achieves an average of 38.0% energy reduction and 1.6$\times$speedup. This greatly reduces the performance loss caused by mobile devices in order to protect the safe execution of applications, so that the trusted execution environment has both security and high performance.

**Key words:** trusted execution environment; mobile system; task scheduling strategy; optimization of energy efficiency

## 1 Introduction

Nowadays, mobile applications (APPs) have become the main carrier of Internet information services. According to a report, the number of monitored APPs in China's domestic market is 3.45 million in 2020[1]. In addition, affected by the COVID-19 epidemic in 2020, people's demands for telecommuting, online education, and online medical treatment keep growing. As of December 2020, the number of online education and online medical treatment users in China is 342 million and 215 million, respectively, accounting for 34.6% and 21.7% of the total users[1]. However, at the same time, a large amount of user privacy and important data related to the vital interests of users is generated in the process of using mobile APPs, and any device with such data may become a target of criminals. Personal information leakage is a primary network security problem encountered by mobile users[1]. IBM's 2020 annual data report shows that the average cost caused by data breaches to enterprises has reached $3.86 million[2]. In addition to the data storage vulnerabilities of developers, the privacy information of mobile devices also has the risk of being stolen. To ensure the security of user data in mobile devices, the traditional method uses encryption technology to store encrypted data locally or send it to other devices. However, some security problems are

• Hai Wang, Lu Cai, Xuan Hao, and Yuhui Ma are with the School of Information Technology, Northwest University, Xi'an 710127, China. E-mail: hwang@nwu.edu.cn; l_cai@stumail.nwu.edu.cn; x_hao@stumail.nwu.edu.cn; mayh@nwu.edu.cn.
• Jie Ren is with the School of Computer Science, Shaanxi Normal University, Xi'an 710061, China. E-mail: renjie@snnu.edu.cn.
∗ To whom correspondence should be addressed.
Manuscript received: 2021-11-03; accepted: 2021-11-18

encountered in this process: The data encryption process is easily monitored by malicious APPs. The key lacks unified management. Key distribution and ciphertext transmission may be attacked.

To address this problem, the Open Mobile Terminal Platform working group proposed the trusted execution environment (TEE) system for storing and processing user-sensitive information. Accordingly, major enterprises have achieved exclusive TEEs, such as Huawei iTrustTEE, ZTE ZTEE, and Samsung TEEgris. Furthermore, ARM Co. proposed TrustZone technology to build a TEE by providing a hardware (HW) isolation mechanism. TrustZone provides two virtual cores: nonsecure (NS) core and secure core. The NS kernel can only access the system resources of NS, whereas the security kernel can access all resources. The two are converted by the monitor mode[3]. APPs in the general world enter the monitor mode using a subset of Secure Monitor Call instructions or HW exception mechanisms. TrustZone technology is widely used for its high efficiency, lightweightness, and security, which can be deployed in heterogeneous mobile systems[4, 5] and smart homes[6], such as Qualcomm SEE architecture, Kinibi microkernel, Trusty, and open-source (OP)-TEE. Among them, OP-TEE is an open-source implementation of TEE developed by STMicroelectronics and Linaro. It has basic microkernel functions and is widely run in mobile systems[7]. Therefore, it has become the first choice for scholars to study the internal mechanisms of TEE. OP-TEE has the following features[7]:

**Isolation.** The TEE provides isolation from NS operating systems (OSs) and protects loaded trusted applications (TAs) from one another using an underlying HW support.

**Small footprint.** The TEE should remain small enough to reside in a reasonable amount of on-chip memory as found on Arm-based systems.

**Portability.** The TEE aims at being easily pluggable to different architectures and available HW and has to support various setups, such as multiple-client OSs or multiple TEEs.

The realization of the specific functions of OP-TEE needs to rely on client application (CA) and TA. The core functions are realized in the TA, and the CA obtains the results through the interface of the TA function. Figure 1 presents the communication process between the TA and CA. A complete function call is initiated by the CA: The CA first opens the OP-TEE driver file to obtain the operation handle connected to the trusted

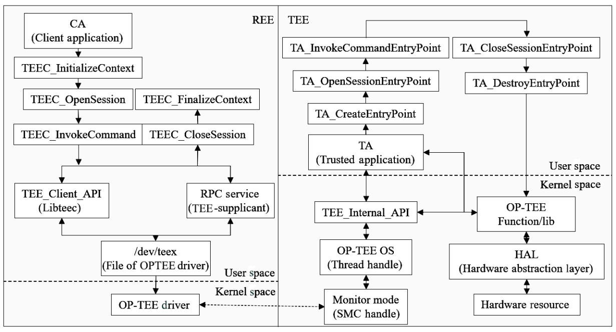


**Fig. 1 Communication process between TA and CA.**

environment (TEEC_InitializeContext). Then, OP-TEE builds a channel to call the TA (TEEC_OpenSession) based on the universally unique identifier (UUID) provided by the CA. Then, the UUID, instructions, and parameters of the specific function are passed to the TA (TEEC_InvokeCommand). At this point, OP-TEE completes the preparation, and the TA executes the call to the corresponding functions according to the instructions. When the TA is executed and the execution result is returned to the CA, OP-TEE enters the end: First, the communication channel between the CA and TA is closed (TEEC_CloseSession). Then, the previously established handle is removed (TEEC_FinalizeContext).

To ensure the security of mobile APPs, they first need to establish a connection with TEE and then communicate with the TA in TEE to inform the TA what task needs to be done. Such an operation leads to a huge overhead for mobile devices (see Section 2.3 for details). In this paper, we propose an adaptive task scheduling strategy to optimize the allocation of system resources so that the mobile system can maintain a short response time and high throughput.

The rest of this paper is organized as follows: Section 2 introduces the background, including the research status at home and abroad, scheduling algorithm, and experimental motivation of this paper. Section 3 summarizes the idea of the ETS-TEE strategy and introduces the implementation process with the fingerprint recognition task as an example. Section 4 introduces the dataset and evaluation indicators. Section 5 introduces the selection process of the prediction model in detail and compares and analyzes the experimental results. Section 6 gives the conclusions.

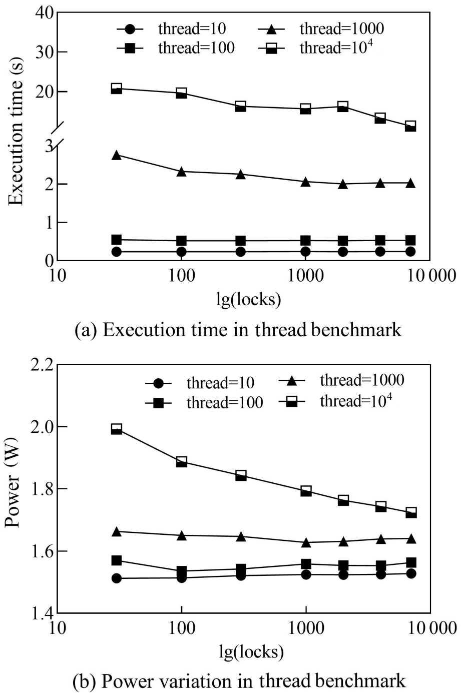
## 2 Background

### 2.1 Related work

TEE is widely used. Over the years, scholarly research on TEE has mainly focused on safety protection. Fan et al.[8] proposed a fine-grained access control scheme based on ciphertext-policy attribute-based encryption and TEE to protect user information and reduce sensitive information attacks. Lee and Park[9] proposed the software framework SofTEE that supports the trusted execution of user APPs. By entrusting the privileged operation in the kernel to a special module called the security monitor, the confidentiality and integrity of the user information in the APP can be ensured. To reduce the switching overhead between the kernel and security monitor, the address space identifier management mechanism is proposed. In addition, Oh et al.[10] combined TEE with field-programmable gate arrays to ensure the security of remote computing by providing a trust anchor for the APP. Wang et al.[11] proposed Hybridchain, which combines blockchains with TEE. Hybridchain decouples computation from consistency, uses hierarchical networks, and executes most of the weight computation from outside the chain to minimize the computational burden and delay of execution on the chain.

Lee and Park[9] and Oh et al.[10] mentioned the program execution overhead in their studies. However, there are only a few studies on the TEE energy efficiency, which is an important part of system analysis[12]. Suzaki et al.[13] constructed the compiler-based performance measurement method TS-Perf. TS-Perf accesses HW timestamp counters in TEE and REE and maintains accurate logs. The measurement code is inserted into the TEE binary file by the compiler option. TS-perf mainly measures the performance of the TEE internal application programming interface, matrix multiplication, memory access, and storage access, and the comparison results show the performance difference between TEE and REE. Suzaki's measurement method is rigorous, and the results are obvious; the method aims to understand the performance of different task loads in TEE and to provide the internal operating overhead information for programmers and architecture developers. Amacher and Schiavoni[14] studied the switching between TEE and REE, secure storage, and CPU benchmark in TEE, but the measurement was not comprehensive. Similarly, no optimization method was proposed. Based on the research of Suzaki et al.[13] and Amacher and Schiavoni[14], we extended the benchmark, added tests on high concurrent programs, implemented and deployed a TA, executed basic operations in the TA, and scheduled threads in the corresponding CA. Each thread opened up and interacted with the TA once, and each thread sequentially allocated mutually exclusive locks. When 10 threads, 100 threads, $10^3$ threads, and $10^4$ threads were opened, the operation of the program was tested with the increase in mutually exclusive lock resources. Figure 2 shows the changes in the thread benchmark test execution time (Fig. 2a) and power (Fig. 2b) with the changes in the thread number and mutual exclusion (mutex) number.

The experimental results show that, on the whole, the larger the number of threads opened is, the greater the time and energy consumption of the program is. That is, the performance of TEE thread scheduling mainly depends on the number of threads opened by the program. When the number of mutually exclusive locks is less than the number of threads, mutually exclusive locks are "in short supply" for threads. Therefore, threads must wait for the execution of other threads to be completed. After releasing mutually exclusive locks, they can be used, resulting in the consumption of the program running time and energy consumption. When the number of mutually exclusive locks is greater than or equal to the number of threads, threads do not have to wait, can successfully execute, and will not have a significant impact on the execution time. However, creating new mutexes and initializing them is also a resource-consuming operation, but the proportion is small and the longest execution time is 2.04 ms



(a) Execution time in thread benchmark

(b) Power variation in thread benchmark

**Fig. 2 Experiment with thread benchmark.**

(when $10^4$ mutexes are applied to run $10^4$ threads simultaneously).

From another point of view, TEE can be understood as dual OSs, so the optimization of these systems can be analogous to TEE. Xie et al.[15] proposed a method combining preemptive priority scheduling and time slice rotation scheduling to schedule tasks in dual OSs. Ran[16] proposed a task scheduling algorithm based on the improved deep Q-learning algorithm. Chen et al.[17] used Hypervisor to sample the running state parameters of virtual resources and used a long short-term memory recurrent neural network to establish the virtual machine energy consumption model. Inspired by the above studies, ETS-TEE is proposed to optimize the TA execution energy efficiency. By using the deep learning algorithm and considering the complexity of TA tasks, TA tasks are dynamically scheduled between local device modeling or unloading to edge servers.

## 2.2 Scheduling policy

Linux supports SCHED_OTHER, SCHED_FIFO, and SCHED_RR scheduling strategies to select one in multiple executable processes and allocate system resources for it[18]. The details are presented as follows: Linux has the default scheduling strategy, which always leads to high energy consumption. Therefore, in TEE, we use the machine learning algorithm to select the optimal scheduling mode for TAs to reduce the extra overhead of TEE.

**SCHED_OTHER.** The scheduler traverses the tasks in the ready queue and calculates the dynamic priority of each task by Eq. (1).

$$Priority = counter + 20 - nice \tag{1}$$

where *counter* is the execution time of the task on the CPU and *nice* is the *Priority* of the task (between –20 and 19). When the task with the largest *Priority* is run, when the *counter* is reduced to zero, or when the CPU is voluntarily abandoned, the task will be placed at the end of the ready queue (the time slice is run out) or waiting for the queue (the CPU is abandoned due to the waiting for resources).

**SCHED_FIFO.** The scheduler traverses the ready queue and calculates the scheduling weights in Eq. (2) according to the real-time priority.

$$Weight = 1000 + rt_{priority} \tag{2}$$

where $rt_{priority}$ is the real-time priority (between 1 and 99). The highest *Weight* task is selected using the CPU, and the CPU is always occupied until a higher-priority task is ready or given up (waiting for resources).

**SCHED_RR.** The scheduler traverses the ready queue, calculates *Weight* according to Eq. (2), and selects the task with the highest *Weight* to use the CPU until the end of a time slice. If the round robin (RR) task time slice in the ready queue is zero, then the time slice of the task is set according to the nice value, and the task is placed at the end of the ready queue.

As can be seen from Eq. (2), the acquisition of *Weight* is only related to $rt_{priority}$. If two or more processes have the same $rt_{priority}$, then the same *Weight* will occur. At this point, the selection of a process to be executed is unfair for other processes because they have the same priority. In TEE, to avoid the above problems and ensure the fairness of trusted task scheduling, the "TA execution times" is introduced as a calculation item in the calculation of the process priority. The high frequency of the TA execution reflects that trusted tasks are often used by users. Therefore, the trusted task should be given a higher priority to improve the user experience. Specifically, the unique UUID of each TA is used to generate a symbol to identify the number of uses of the TA and reset the value after a period of time. We define the *TA_counter*() function, which is called each time the process weight is calculated to obtain the number of execution $TA_{counts}$ of the current TA. Based on this, a first-in, first-out (FIFO) based on the TA execution frequency (FIFO TAF) and an RR based on TA execution frequency (RR TAF) are proposed on the basis of the Linux scheduling strategy. Both of them use Eq. (3) to calculate the weights of TAs.

$$weight = 1000 + rt_{priority} + TA_{counts} \tag{3}$$

Then, the highest *weight* task is selected using the CPU until a higher-priority task is ready or a time slice is over. The improved scheduling model not only improves the fairness of trusted task scheduling but also realizes personalized customization based on different user habits for TEE. Algorithm 1 shows the operation logic of the improved priority calculation function *TA_goodness*().

## 2.3 Motivation

The mobile user experience is highly sensitive to

**Algorithm 1 Function *TA_goodness* algorithm**

```
Input: $x$ is the set of TAs
Output: $MAX(Weight)$
1: for each $x_i \in X$ do
2:     $TA_{counts} = TA\_counter(x_i\_UUID)$;
3:     $Weight = 1000 + rt_{priority} + TA_{counts}$ // Eq. (3)
4: end for
5: for each $x_i \in X$ do
6:     select $MAX(Weight_i, Weight_j, \ldots, Weight_n)$
7: end for
```

performance-energy constraints[19, 20], but there are only a few system analyses on the performance of OP-TEE until now. The motivating example attempts to perform a comprehensive analysis on OP-TEE by taking the text encryption operation, which is a commonly used operation on mobile devices. We use cipher block chaining (CBC), electronic codebook (ECB), and counter (CTR) to encrypt the 4 KB text content in the OP-TEE and REE, respectively, and calculate the elapsed time and energy consumption. In addition, without considering the communication overhead between the CA and TA, only the time for the TA to perform core functions is measured separately. The experimental results are shown in Fig. 3.

Figures 3a, 3c, and 3e present that the average times of the complete running of the CBC (143.28 ms), ECB (138.92 ms), and CTR (131.04 ms) encryption programs in OP-TEE were 39.91, 41.35, and 39.95 times higher than the times spent in REE (CBC: 3.59 ms, ECB: 3.36 ms, and CTR: 3.28 ms), respectively. The average times to execute the TA (core TA) were 5.91, 5, and 2.73 times higher than those in REE. Figures 3b, 3d, and 3f show the energy consumption of the OP-TEE and RPi 3B (change to REE) for performing the same function. The results show that the energy consumption and latency of the three commonly used encryption algorithms in OP-TEE are higher than those in REE. This is due to the OP-TEE execution encryption algorithm through the allocation of encryption resources, key loading, initialization vector (IV) reset, and input buffer encryption to the output buffer in four processes. Each process requires permission verification before execution. The OP-TEE definitely provides a reliable runtime environment, but the runtime performance penalty and energy consumption are not negligible.

Figure 4 presents the latency and energy of CBC with three different task scheduling strategies. The FIFO has the minimum delay, and RR achieves the lowest energy loss among all of the strategies.

Essentially, the optimal task scheduling strategy is different and not always the default strategy[21, 22]. Hence, we leveraged the machine learning techniques and predicted which strategy to use for the coming task based on the task workload and user experience.

## 3 Our Approach

Our goal is to reduce the overhead of executing tasks in TEE. Thus, we propose ETS-TEE. Figure 5 presents the

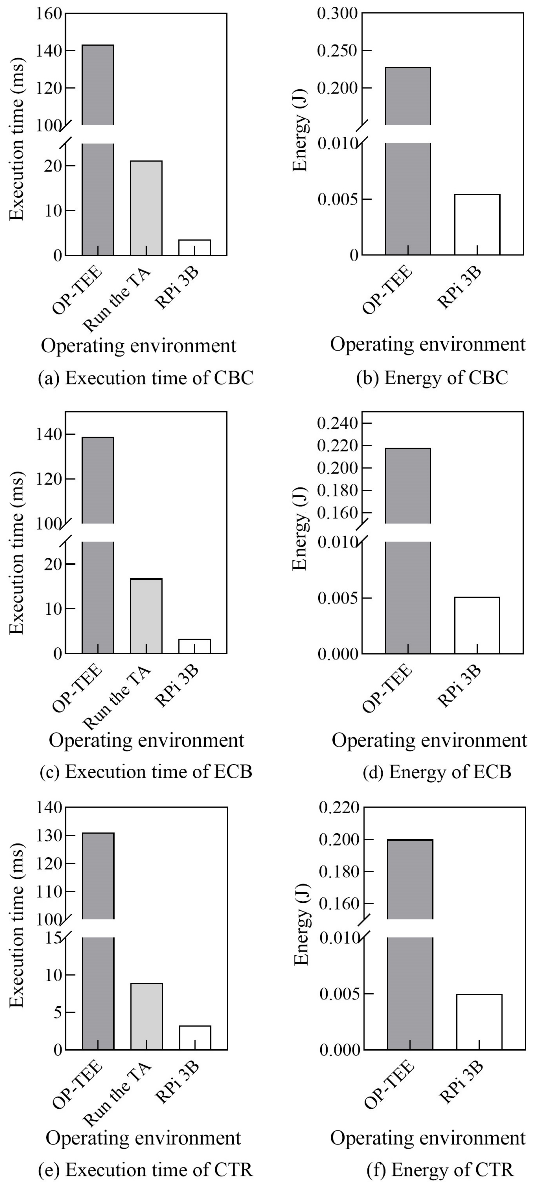


**Fig. 3 OP-TEE vs. RPi 3B for running the same encryption algorithm.**

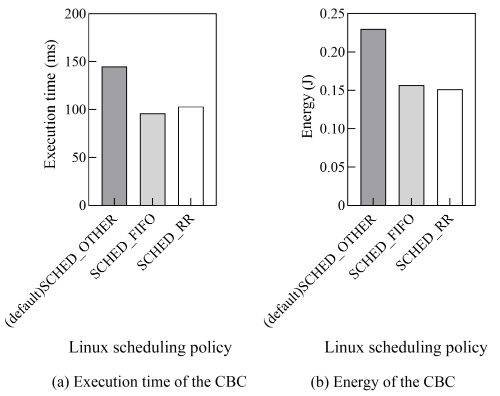


**Fig. 4 Execution time and energy of CBC in different scheduling modes.**

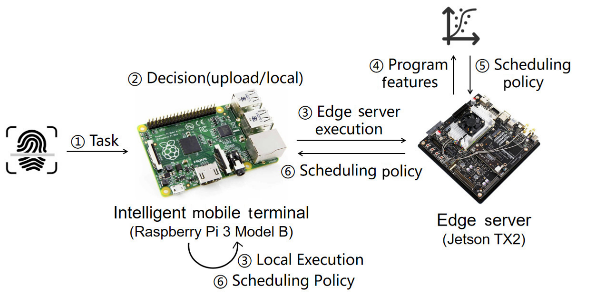


**Fig. 5 Fingerprint identification task scheduling strategy based on edge devices.**

overview of ETS-TEE.

• Mobile APP input fingerprint identification task.

• Extract mobile APP running task features and feed them into the local decision maker, which makes decisions on the task (which policy to use or unload) according to the hrtimer value of the task and the specified threshold, and schedule the classified tasks to the edge server or local.

• If executed locally, then use the system default scheduling policy (whose target is the minimum delay) or schedule tasks through the local model (whose target is the lowest energy loss). Otherwise, extract and send task features to the edge.

• Input task features into the edge server training model on the edge server.

• Edge server model outputs scheduling pattern labels for current tasks based on the task complexity.

• Return results from the local model or edge server model.

## 4 Predictive Model

### 4.1 Feature selection

According to the characteristics of the TA runtime in OP-TEE, multiple features were selected. The Pearson correlation coefficient was used to reflect the similarity between features, and the correlation coefficient matrix was constructed. The correlation coefficient value is between –1 and 1. The greater the absolute value of the correlation coefficient between features, the stronger the correlation; that is, the closer the correlation coefficient of the two features is to 1 or –1, the greater the correlation. Deleting one of the two features with a strong correlation will not affect the performance of classification results. In the depth model training process, the feature correlation is a strong linear correlation when it is greater than 0.75, so the feature with an absolute correlation greater than 0.75 is deleted.

A chi-square test was used to sort the features, and the top 10 features were retained for importance screening. The screening method is as follows: For the above ten features, one feature is deleted in each experiment, and the effect of missing this feature on the accuracy of the model is observed. As shown in Fig. 6, the influence of a single feature on the classification accuracy of the model is shown from large to small, and the first six features that have a great influence on the model accuracy are retained as the final feature input (as shown in Table 1).

At the same time, to avoid the masking of a feature with a small value due to a large eigenvalue and affect the prediction results of the model, we normalize the feature data through the discrete standardization method, as shown in Eq. (4).

$$x_{normalization} = \frac{x - Min}{Max - Min} \tag{4}$$

Because the TA is scheduled to the edge server or local device according to the prediction results of ETS-TEE, if better prediction results can be obtained with a low complexity feature set, then the prediction delay of the model will also be reduced to some extent. Thus, we combined the six features shown in Table 1 to construct feature set $A$, which contains only one feature

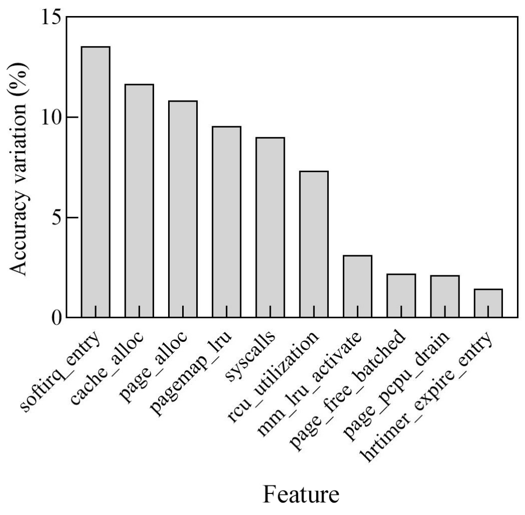


**Fig. 6 Influence of features on model accuracy.**

**Table 1 Model features.**

| Feature name | Description |
|---|---|
| softirq_entry | Number of software interrupts. |
| cache_alloc | Number of cache allocations while the program is running. |
| page_alloc | Number of pages allocated while the program is running. |
| pagemap_lru | Number of memory pages accessed by the program. |
| syscalls | Number of system calls. |
| rcu_utilization | RCU usage. |

of softirq_entry. Then, we built feature set $B$ and added cache_alloc feature on feature set $A$, which contains two features. We constructed feature set C and added the page_alloc feature to feature set $B$, which contains three features. By analogy, feature set $F$ is the most complex feature set that contains all the features in Table 1. Table 2 describes the feature sets with different complexities.

### 4.2 Model selection

The selection of a suitable machine learning algorithm in resource-constrained smart mobile terminals needs to consider three factors: fast execution time, low energy consumption, and high accuracy. In this study, the K-Nearest Neighbor (KNN), Decision Tree (DT), Naïve Bayesian (NB), and Support Vector Machine (SVM) are used to predict the optimal scheduling strategy for TAs.

Among them, the selection of the $K$ value in the KNN model will have an impact on the classification process. For example, a too-small $K$ value means that the overall model becomes complex and is prone to overfitting, whereas a too-large $K$ value will lead to simplification and loss of significance of the model. The DT model makes decisions based on the structure of the tree. When making decisions, the optimal partition attribute is selected, and the dataset is segmented by the optimal attribute. Each subset of the segmentation recursively calls this method until the termination condition (e.g., prepruning condition) is reached. NB is a classification method based on the Bayesian theorem and independent assumption of feature condition. For a given training dataset, the joint probability distribution of input/output is first learned based on the independent assumption of feature conditions. Then, based on this model, for a given input $x$, the output $y$ with the maximum posterior probability is obtained using the Bayesian theorem. NB commonly uses three models: the Gaussian model, polynomial model, and Bernoulli model. Among them, the Gaussian model is used to deal with continuous feature variables. When this model is used, the feature is assumed to be a Gaussian distribution, and then the mean and standard deviations of the feature are calculated based on the training samples to obtain the prior probability of each attribute value under this feature. Contrary to the Gaussian distribution, the polynomial model is mainly applicable to the prior probability calculation of discrete features. The Bernoulli model is a random variable distribution model with only "yes" or "no" results. When the model is used, the value of all features becomes 0 or 1. If the feature itself is not 0 or 1, then a threshold is automatically set, which is set to 0 below the threshold and 1 above the threshold, so the feature is $0-1$. Then, the probability of $0-1$ is calculated in all training samples as the conditional probability of the feature. SVM is a binary classifier. When dealing with multiclass problems, it is necessary to construct a suitable multiclass classifier. At present, two main methods are used for constructing an SVM multiclass classifier: OvO SVM and OvR SVM. OvO is one versus one. A classifier is built between any two types of samples, and $N \times (N-1)/2$ classifiers are needed for $N$ categories. OvR is one versus the rest, which classifies a class of samples into one class, and the remaining samples into another class. For $N$ classes, only $N$ classifiers are needed. According to the characteristics of different algorithms, we adjust the parameters of the four algorithms and select the parameters with the best model effect for training.

**Table 2 Sets of model feature groups.**

| Set name | Feature groups within set |
|---|---|
| $A$ | softirq_entry |
| $B$ | $A$ + cache_alloc |
| $C$ | $B$ + page_alloc |
| $D$ | $C$ + pagemap_lru |
| $E$ | $D$ + syscalls |
| $F$ | $E$ + rcu_utilization |

### 4.3 Dataset

We built a cross-compiler environment on Ubuntu 16.04 and recompiled the Perf performance test tool with the same cross compiler as when compiling OP-TEE so that it can run in a TEE. We used the recompiled Perf tool to record OP-TEE built-in Xtest set running as a dataset. The dataset contains the operation characteristics of 101 TAs, including 75 TAs in the training set and 26 TAs in the test set. In this study, we evaluated the model classification effect via 10-fold cross-validation. Specifically, after the dataset was randomly disrupted, 10% of the data were randomly selected as the test set, and the remaining dataset was used as the training set. This process was repeated ten times, and the classification algorithm was evaluated according to the average accuracy.

### 4.4 Metric

We took the latency and energy consumption as the starting points. The execution time of TA $x$ is denoted as $T_x$, which is defined as Eq. (5).

$$T_x = T_x^{conn} + T_x^{exec} + T_x^{upload} + T_x^{model} \tag{5}$$

where $T_x^{conn}$ is the interaction time of establishing a connection between the REE and TEE, $T_x^{exec}$ is the execution time of TA, $T_x^{upload}$ is the transmission time of the edge server of the task upload, and $T_x^{model}$ is the inference time of the latency model. Specifically, the value of $T_x^{upload}$ is defined as Eq. (6).

$$T_x^{upload} = \frac{S_x}{Bw} \tag{6}$$

where $S_x$ is the relevant data size of TA $x$ and $Bw$ is the network bandwidth between the mobile device and edge server. At the same time, the operating energy consumption of the TA $x$ is expressed as $E_x$, which is defined as Eq. (7).

$$E_x = E_x^{conn} + E_x^{exec} + E_x^{model} \tag{7}$$

where $E_x^{conn}$ is the energy consumption of establishing a connection between the REE and TEE, $E_x^{exec}$ is the execution energy consumption of the TA, and $E_x^{model}$ is the inference energy consumption of the energy model.

Therefore, the prediction model was built with the highest response speed (i.e., $Min(T_x)$) and lowest operating energy consumption (i.e., $Min(E_x)$) as the objectives. Optimal scheduling strategies are recommended for TAs under different metrics.

## 5 Experimental Evaluation

### 5.1 Setup

**HW platform.** We used the NVIDIA Jetson TX2 embedded platform as the edge server, which is equipped with dual-core Denver2 and four-core ARM Cortex-A57. The mobile terminal uses a Raspberry Pi 3 Model B embedded platform, which is equipped with a 64-bit four-core A53 processor and 1 GB of RAM. At the same time, the experiment used the external HW measurement tool Power-Z KM001 device to measure the power consumption of RPi 3B in real time.

**Software platform.** Jetson TX2 runs Ubuntu 16.04. Python scikit-learn builds a prediction model. RPi 3B runs OP-TEE compiled by aarch64-linux-gnu cross compiler.

### 5.2 Trusted application scheduling policy for low latency

To speed up the TA response and reduce energy consumption, ETS-TEE is proposed, which specifies the task scheduling process. Among them, a low delay trusted application scheduling strategy based on Edge server (LD-TA-Edge based) is proposed for the minimum latency target of ETS-TEE. Figure 7 presents its workflow. The strategy uploads the runtime

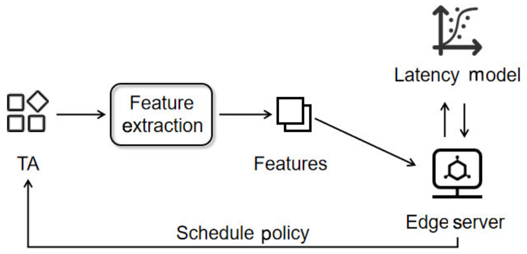


**Fig. 7 LD-TA-Edge based execution process.**

characteristics of the program to the edge server, which is deployed through the Wi-Fi network. The machine learning model deployed in the edge server will use the feature set to reason and output the reasoning result; that is, the scheduling mode label with the lowest running time, and the trusted task will be executed according to the label.

#### 5.2.1 Latency model

The low-delay prediction model takes the optimal delay as the goal, inputs the program feature set with different complexities, and outputs the scheduling pattern label with the lowest running time.

As shown in Fig. 8, KNN, DT, NB, and SVM are used to construct low-delay prediction models, and the classification effects of these algorithms are compared. Among them, the NB classifier uses three different models[23], and Fig. 8c shows that the NB classification with a Gaussian distribution as the prior has the best effect. SVM has two strategies for multiclass classification[24]: OvO, which is to design an SVM between any two classes of samples, and OvR, which is to classify a class of samples in turn during training and the remaining samples into another class. For the two strategies, a grid search algorithm was used to optimize the penalty factor and kernel function parameters of

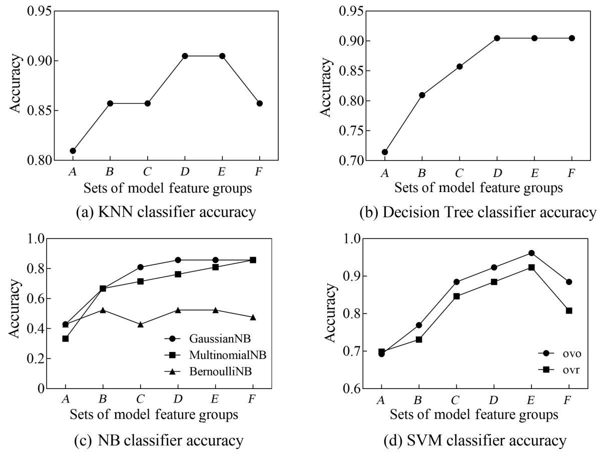


**Fig. 8 Classification accuracy of different classifiers on time feature sets.**

SVM[25]. Figure 8d shows the classification accuracy of the two strategies, which shows that SVM-OvO is better than SVM-OvR.

The models with good results are compared together, and the results are shown in Fig. 9. In Fig. 9, the SVM algorithm has the highest accuracy (96.15%) when the feature set $E$ is inputted. Therefore, SVM based on the grid search algorithm is selected as the core algorithm of the LD-TA-Edge based execution process.

### 5.2.2 Inference latency and energy consumption

Figure 10a shows the comparison of the time overhead of the LD-TA-Edge based execution process and the reasoning time of SVM + $E$ executed locally. The graph shows that the time cost of the LD-TA-Edge based execution process (1.71 s) is 3.6x higher than that of SVM + $E$ (6.21 s). Figure 10b shows the comparison of the energy consumption between the two strategies. Figure 10b also shows that the energy consumption of the LD-TA-Edge based execution process is only 2.68 J, which is 73.2% less than that of the local SVM + $E$ (10.01 J).

The average execution delay of the LD-TA-Edge based execution process is 37.4% lower than that of TAs. As described in the motivational experiment in Section 2.3 of this paper, the average time consumptions of the fully running CBC (143.28 ms), ECB (138.92 ms), and CTR (131.04 ms) encryption programs in the TEE are 39.91, 41.35, and 39.95 times higher than those in the REE, respectively. By using the LD-TA-Edge based strategy, the average time consumption of the above encryption algorithms (CBC: 90.41 ms, ECB: 86.69 ms, and CTR: 85.18 ms) is reduced by 36.9%, 37.6%, and 35.0%, respectively, compared with that in the ordinary environment (25.18, 25.80, and 25.97 times), which is 14.73, 15.55, and 13.98, respectively.

### 5.2.3 Overhead

The proposed low-delay prediction model only needs offline training and deployment once, without repeated training. Therefore, the proposed LD-TA-Edge based scheduling strategy has a low running overhead, which only includes the overhead of the task upload. The overhead in different network environments is shown in Table 3.

## 5.3 Trusted application scheduling policy for low energy consumption

To speed up the TA response and reduce energy consumption, ETS-TEE is proposed, which specifies the task scheduling process. Among them, a local low-energy TA (LE-TA) scheduling strategy is proposed for the low-energy-consumption target of ETS-TEE. Figure 11 presents its workflow.

This strategy uses the machine learning model trained with the optimal energy consumption as the goal to complete the reasoning of the optimal energy consumption operation mode of the trusted task through the task feature set locally and specify the trusted task to run according to this mode to consume the lowest energy.

### 5.3.1 Energy model

The low-energy-consumption prediction model takes

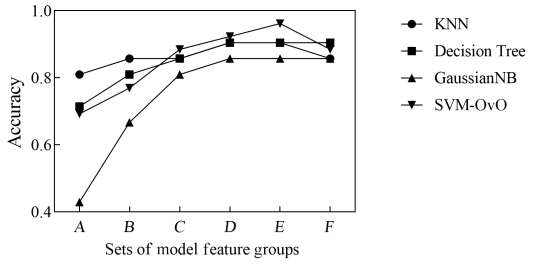


**Fig. 9 Summary of classification model accuracy on time feature sets.**

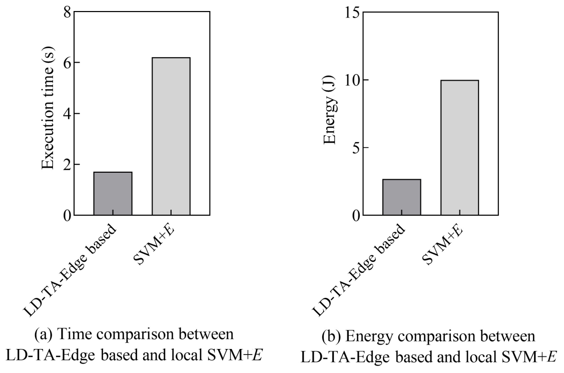


**Fig. 10 LD-TA-Edge based versus SVM+$E$.**

**Table 3 Energy consumption of LD-TA-Edge based in different network environments.**

| Network environment | Upload limit (bps) | Upload energy (J) |
|---|---|---|
| WIFI | $1.9\times10^6$ | 0.000 37 |
| Good 4G | $1.0\times10^6$ | 0.000 71 |
| Regular 4G | $1.28\times10^5$ | 0.005 66 |

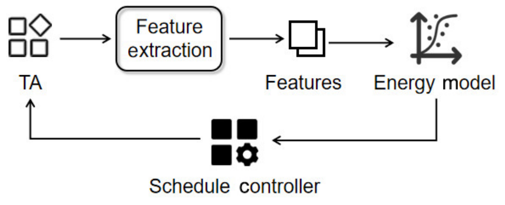


**Fig. 11 LE-TA execution process.**

the optimal energy consumption as the goal, inputs the program feature set with different complexities, and outputs the scheduling mode label with the lowest running energy consumption. The experimental results are shown in Fig. 12.

As shown in Fig. 12, KNN, DT, NB, and SVM are used to construct the low-energy-consumption prediction models, and the classification effects of different algorithms are compared. Among them, Fig. 12c shows that the Naïve Bayesian classification with a Gaussian distribution as the prior is the best. Figure 12d shows that the SVM-OvO classification is generally the best. The models with good results are compared, and the results are shown in Fig. 13. As shown in Fig. 13, the KNN algorithm has the highest accuracy (95.24%) when the feature set F is inputted. Therefore, the KNN is selected as the core algorithm of LE-TA in this study.

### 5.3.2 Inference time and energy consumption

LE-TA has a low time cost (16 ms) and low energy consumption (0.024 J), so choosing a local LE-TA strategy can achieve a good performance.

The experiment shows that the energy consumption of LE-TA is reduced by 38.0% compared with the default execution of the TA. As described in the motivation experiment in Section 2.3 of this paper, the average energy consumptions of the complete operation of the CBC, ECB, and CTR encryption programs in the TEE are 0.2279 J, 0.2178 J, and 0.2000 J, respectively. Through the scheduling of the LE-TA strategy, the average energy consumption of the above encryption algorithms (CBC: 0.1431 J, ECB: 0.1348 J, and CTR: 0.1262 J) is reduced by 37.2%, 38.1%, and 36.9%, respectively.

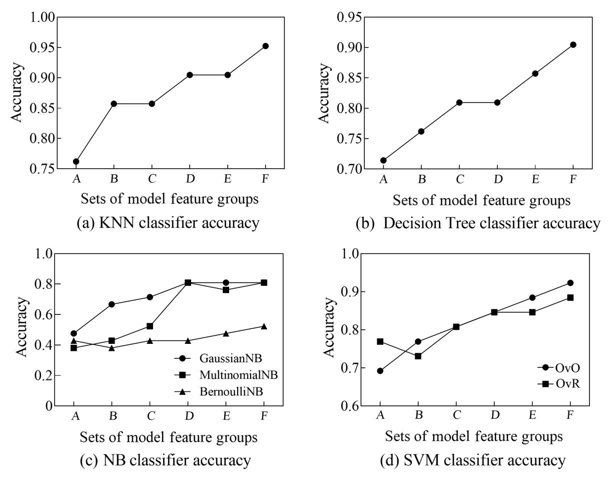


**Fig. 12 Classification accuracy of different classifiers on energy feature sets.**

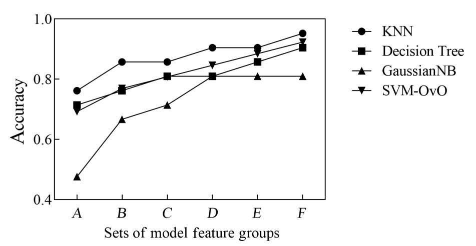


**Fig. 13 Summary of classification model accuracy on energy feature sets.**

## 6 Conclusion

In this work, the proposed ETS-TEE (energy-efficient task scheduling strategy) effectively reduced the overhead of executing tasks in TEE. In detail, according to the expectation of time and energy consumption, we propose the LD-TA-Edge based strategy (low-delay TA scheduling strategy based on the edge server) and LE-TA scheduling strategy deployed on NVIDIA Jetson TX2 and Raspberry Pi 3B platforms. The experimental results show that the low-delay trusted task scheduling strategy based on the edge server is 3.6 times faster than the local running speed, and the energy consumption is reduced by 73.2%. Compared with the default execution of TAs, the proposed LE-TA strategy saves 38.0% energy and the LD-TA-Edge based strategy speeds up by 37.4%.

## Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61902229) and Fundamental Research Funds for the Central Universities (No. GK202103084) .

## References

[1] China Internet Network Information Center, The 47th statistical report on internet development in China, http://www.cnnic.cn/hlwfzyj/hlwxzbg/hlwtjbg/202102/t20210203 71361.htm, 2021.

[2] IBM Security, Data breach costs report for 2020, https://www.ibm.com/security/data-breach, 2020.

[3] ARM, Introducing arm trustzone, https://developer.arm.com/technologies/trustzone, 2021.

[4] J. Ren, X. M. Wang, J. B. Fang, Y. S. Feng, D. X. Zhu, Z. C. Luo, J. Zheng, and Z. Wang, Proteus: Network-aware web browsing on heterogeneous mobile systems, in *Proc. 14$^{th}$ ACM International Conference on Emerging Networking Experiments and Technologies* (*CoNEXT*), Heraklion, Greece, 2018, pp. 379–392.

[5] X. M. Wang, L. Li, J. Li, and Z. Q. Li, A turbo Q-learning (TQL) for energy efficiency optimization in heterogeneous networks, *Entropy*, vol. 22, no. 9, p. 957, 2020.

[6] Z. N. Mohammad, F. Farha, A. O. M. Abuassba, S. K. Yang, and F. Zhou, Access control and authorization in smart homes: A survey, *Tsinghua Science and Technology*, vol. 26, no. 6, pp. 906–917, 2021.
[7] Linaro, Open portable trusted execution environment, https://www.op-tee.org/, 2021.
[8] Y. Fan, S. L. Liu, G. Tan, and F. Qiao, Fine-grained access control based on trusted execution environment, *Future Generation Computer Systems*, vol. 109, pp. 551–561, 2020.
[9] U. Lee and C. Park, SofTEE: Software-based trusted execution environment for user applications, *IEEE Access*, vol. 8, no. 99, pp. 121874–121888, 2020.
[10] H. Oh, K. Nam, S. Jeon, Y. Cho, and Y. Paek, MeetGo: A trusted execution environment for remote applications on FPGA, *IEEE Access*, vol. 9, pp. 51313–51324, 2021.
[11] Y. Wang, J. Li, S. Zhao, and F. Yu, Hybridchain: A novel architecture for confidentiality-preserving and performant permissioned blockchain using trusted execution environment, *IEEE Access*, vol. 8, pp. 190652–190662, 2020.
[12] J. Chen, X. X. Qi, F. H. Wu, J. B. Fang, Y. Dong, Y. Yuan, Z. Wang, and K. Q. Li, More bang for your buck: Boosting performance with capped power consumption, *Tsinghua Science and Technology*, vol. 26, no. 3, pp. 370–383, 2021.
[13] K. Suzaki, K. Nakajima, T. Oi, and A. Tsukamoto, TS-Perf: General performance measurement of trusted execution environment and rich execution environment on Intel SGX, Arm TrustZone, and RISC-V keystone, *IEEE Access*, vol. 9, pp. 133520–133530, 2021.
[14] J. Amacher and V. Schiavoni, On the performance of ARM trustzone, *Distributed Applications and Interoperable Systems*, doi: 10.1007/978-3-030-22496-7_9.
[15] C. Xie, X. L. Chai, and X. You, Research on scheduling strategy of dual-operating system based on trustzone technology, *Microcontrollers & Embedded Systems*, vol. 20, no. 11, pp.71–74, 2020.
[16] L. Y. Ran, Research on efficient cloud task scheduling algorithm based on deep reinforcement learning, Master dissertation, Chongqing Institute of Green and Intelligent Technology, University of Chinese Academy of Sciences, China, 2020.
[17] J. Chen, J. Zhang, H. Z. Tian, C. X. Song, and W. Zou, A method for virtual machine energy consumption model building by LSTM cyclic neural network, *Transducer and Microsystem Technologies*, vol. 39, no. 9, pp. 16–19, 23, 2020.
[18] Y. He and S. X. Liu, The scheduling policy of linux, *Electronic Science and Technology*, no. 5, pp. 31–34, 38, 2004.
[19] Z. Zhou, Y. J. M. Yuan, and F. M. Li, Energy consumption modeling and quantitative calculation of servers in cloud data center, *Journal of Hunan University* (*Natural Science Edition*), vol. 48, no. 4, pp. 36–44, 2021.
[20] J. W. Li, Z. Liu, Z. J. Lai, and F. Yin, Study on smart phone power consumption test based on user experience, *Telecommunication Network Technology*, no. 10, pp. 62–65, 2014.
[21] J. Ren, L. Gao, J. L. Yu, and L. Yuan, Energy-efficient deep learning task scheduling strategy for edge device, *Chinese Journal of Computers*, vol. 43, no. 3, pp. 440–452, 2020.
[22] M. F. Su, G. J. Wang, and R. F. Li, Resource deployment with prediction and task scheduling optimization in edge cloud collaborative computing, *Journal of Computer Research and Development*, vol. 58, no. 11, pp. 2558–2570, 2021.
[23] Y. I. Kurniawan, T. Cahyono, Nofiyati, E. Maryanto, A. Fadli, and N. R. Indraswari, Preprocessing using correlation based features selection on naive bayes classification, *IOP Conference Series: Materials Science and Engineering*, vol. 982, no. 1, p. 012012, 2020.
[24] S. Mallika, An efficient machine leaning based gene expression cancer diagnosis, *Software Engineering*, vol. 5, no. 9, pp. 336–341, 2013.
[25] C. Du, J. H. Shao, W. Yang, Z. S. Wang, L. J. Deng, and H. J. Shen, Support vector machine indoor visible light positioning optimized by grid search method, (in Chinese), *Laser Journal*, vol. 42, no. 3, pp. 104–109, 2021.

**Hai Wang** received the BS, MS, and PhD degrees from Xi'an Jiaotong University, Xi'an, China, in 2000, 2004, and 2010, respectively. He is an associate professor at the School of Information Technology, Northwest University, Xi'an, China. He is a deputy director of Shaanxi New Network Security Guarantee and Service Engineering Laboratory, a member of China Computer Society, and special committee of Network and Data Communication. His current research interests include mobile network management, service computing, and semantic Web.

**Jie Ren** received the PhD degree in computer system architecture from Northwest University in 2017. He has been on a two-year academic visitor at the School of Computing and Communication, Lancaster University, UK. He is currently a lecturer at the School of Computer Science, Shaanxi Normal University, Xi'an, China. His current research interests include "performance-power" optimization for mobile devices, on-device deep learning, mobile augmented reality, and natural language processing.

**Lu Cai** received the BS degree in Internet of things engineering from Northwest University, Xi'an, China, in 2019. She is currently pursuing the MS degree in computer application technology at the School of Information Technology, Northwest University, Xi'an, China. Her current research interests include machine learning and the optimization of energy efficiency.

**Xuan Hao** received the BS degree in Internet of things engineering from Northwest University, Xi'an, China, in 2019. She is currently pursuing the MS degree in computer application technology at the School of Information Technology, Northwest University, Xi'an, China. Her current research interests include machine learning and the optimization of energy efficiency.

**Yuhui Ma** received the MS degree in electromagnetic field and microwave technology from Xidian University in 2018. She is currently a teacher at the Center for Experimentation and Instruction at the School of Information Technology, Northwest University, Xi'an, China. Her current research interests include microwave technology and antenna, information security, and intelligent information processing.