Sipster: Settling IOU of Smart Meters Privately and Quickly (Multiple Benchmarks)

The Sipster paper can be access at
https://www.dropbox.com/s/8sh9t6yrnlggnc0/SMC-CPS.pdf

A. How the code matches with Sipster's algorithms?

The code contains seven functions that match with the critical procedures described in section 4.3. We use the blue text to mark the functions used in the code and explain how they fit with specific procedures as follows.

1. Setup Phase:
In this phase, the smart meters (SM), utility company (UC), and residential users (RU) will initialize their critical parameters. We complete these initializations in the main function of the Sipster.

2. Bill Issuing Phase:
SM updates its internal state τ and records the real-time consumption of RU and calculates the fine-grained expense. For each unit of expense, SM runs the stateful algorithm TokenGen (Algorithm 1). We use the SM_TokenGen function to describe these steps.

3. Bill Settlement Phase:
Once RU receives a token tk in the Bill Issuing Phase, RU can settle the bill for this particular token and receive receipts from UC (Algorithm 2). Receipts are then verified by RU to ensure their correctness. We use the UC_ReceiptGen function and RU_Verify function to achieve these procedures.

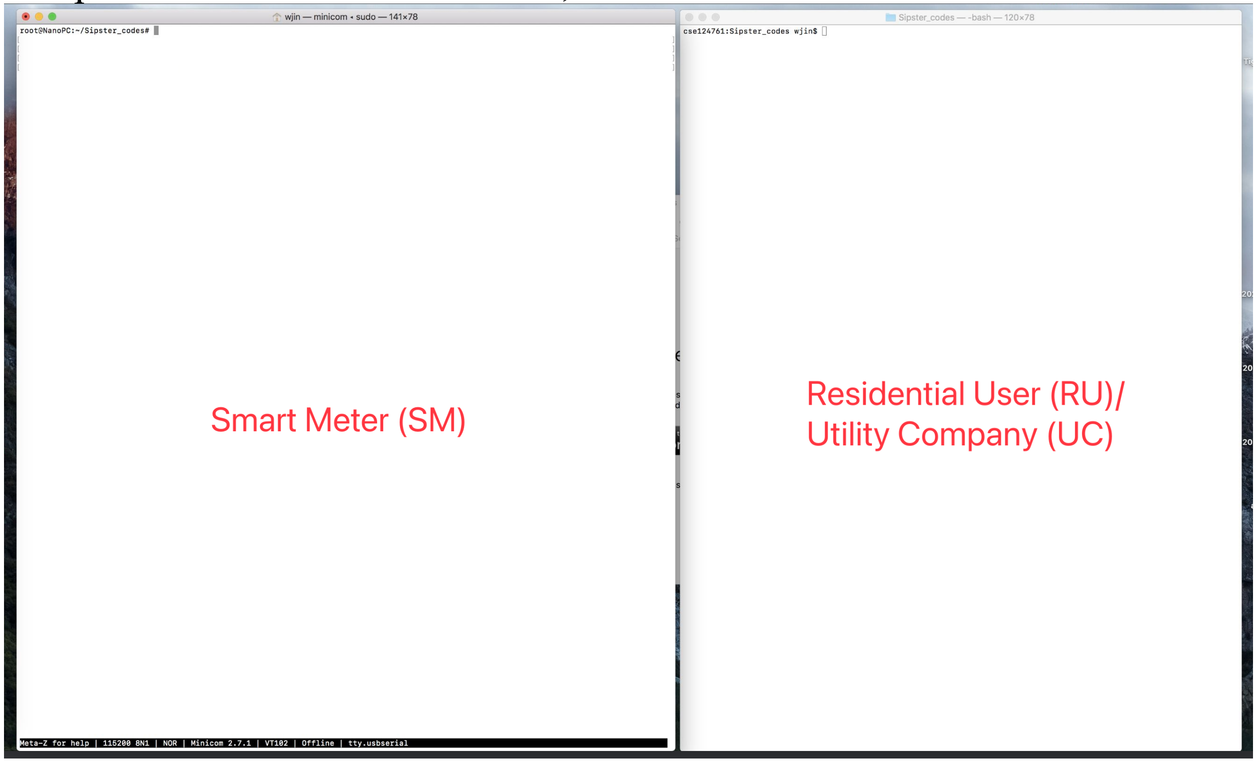
4. Bill Verification Phase:
In the billing period t, SM generates K tokens. Thus, RU collects K receipts from UC. To verify all bills are settled, SM first needs to prepare bill information (Algorithm 3) for RU. We use SM_BillGen function to generates the bill. Then, RU combines all his receipts and submits them to UC as proof of bill settlements (Algorithm 4). We use RU_CombineReceipt function to combine the receipts. And, UC_Bill_Verify function finally verifies the bill settlements.

B. How to run the codes?

The program is designed to evaluate the computation time of different algorithm phases across three Sipster devices, i.e., SM, RU, and UC. The two variables that impact the computation time are the number of RU (i.e., N) and the number of tokens (i.e., K).

We provide interactive shell commands for the reviewers to benchmark the Sipster.

The compiled Sipster codes output a runnable file "Sipster.run". By typing "./Sipster.run" in the shell terminal, we can enter the interactive interference.

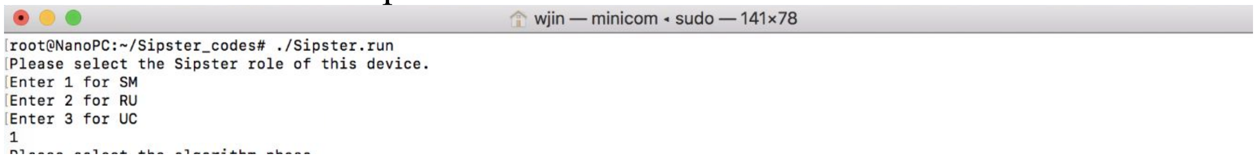

wjin — minicom ◂ sudo — 141×78
Sipster_codes — -bash — 120×78
Smart Meter (SM)
Residential User (RU)/
Utility Company (UC)

```
root@NanoPC:~/Sipster_codes# 
```

```
Meta-Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | tty.usbserial
```

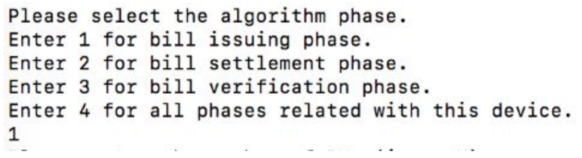
```
cse124761:Sipster_codes wjin$ 
```

The program will first ask the Sipster role of this device, i.e., SM, RU, or UC. We set up the SM device using an ARM board, and its shell terminal is shown as the left-side window. The right-side terminal window is designed to benchmark the RU/UC performances.



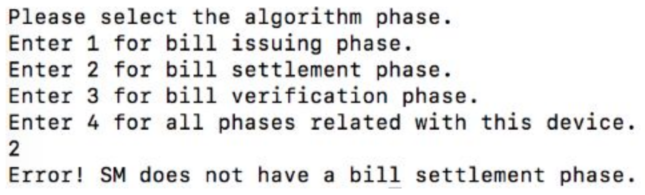
```
root@NanoPC:~/Sipster_codes# ./Sipster.run
Please select the Sipster role of this device.
Enter 1 for SM
Enter 2 for RU
Enter 3 for UC
1
```

Then, the program requires the user to specify which algorithm phase(s) she/he wants to test. Sipster contains three different phases, including the bill issuing phase, bill settlement phase, and bill verification phase. By entering “1”, the program will benchmark the computation time of SM during the bill issuing phase.



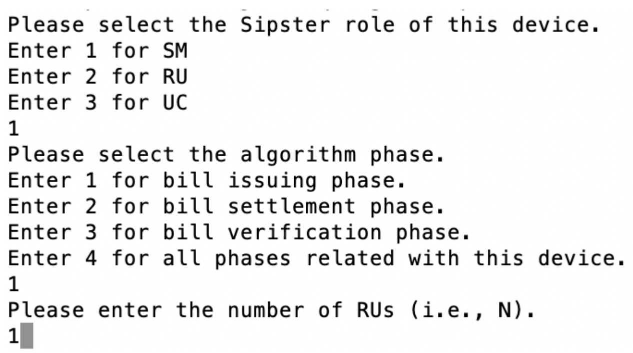
```
Please select the algorithm phase.
Enter 1 for bill issuing phase.
Enter 2 for bill settlement phase.
Enter 3 for bill verification phase.
Enter 4 for all phases related with this device.
1
```

Noteworthy, SM does not participate in the bill settlement phase. If the user wrongly entered “2”, the program will stop and throw out errors.



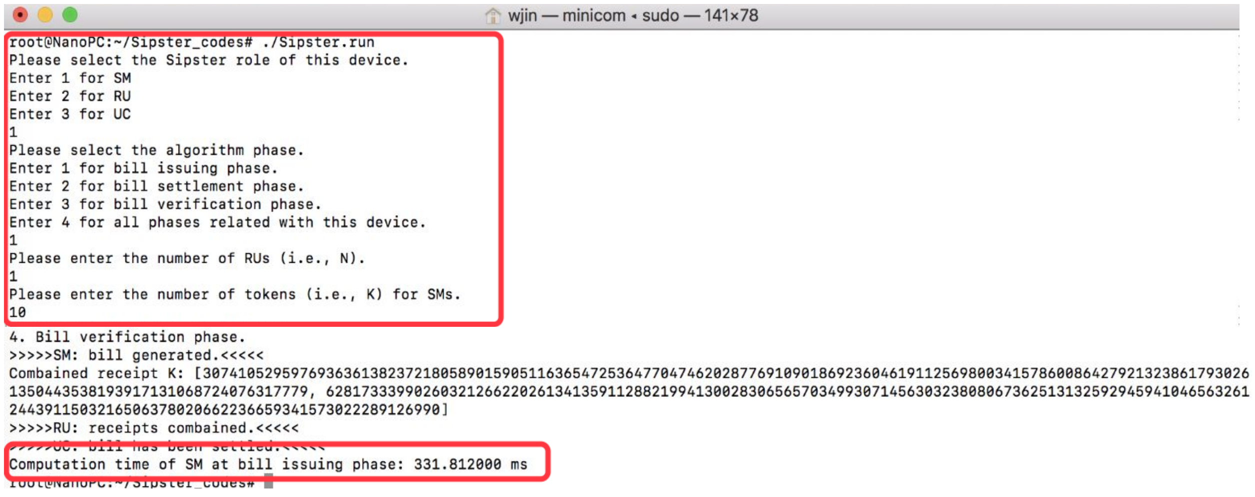
```
Please select the algorithm phase.
Enter 1 for bill issuing phase.
Enter 2 for bill settlement phase.
Enter 3 for bill verification phase.
Enter 4 for all phases related with this device.
2
Error! SM does not have a bill settlement phase.
```

The program will also ask the number of RUs, i.e., N, related to the current device. Since one smart meter (SM) is only associated with one RU, we input “1” for this case. However, UC devices can process the bills from multiple RUs.



```
Please select the Sipster role of this device.
Enter 1 for SM
Enter 2 for RU
Enter 3 for UC
1
Please select the algorithm phase.
Enter 1 for bill issuing phase.
Enter 2 for bill settlement phase.
Enter 3 for bill verification phase.
Enter 4 for all phases related with this device.
1
Please enter the number of RUs (i.e., N).
1
```

Finally, by entering the number of tokens, i.e., K, used in the algorithm, the program will run with the previous inputted parameters and measure the computation time.



```
root@NanoPC:~/Sipster_codes# ./Sipster.run
Please select the Sipster role of this device.
Enter 1 for SM
Enter 2 for RU
Enter 3 for UC
1
Please select the algorithm phase.
Enter 1 for bill issuing phase.
Enter 2 for bill settlement phase.
Enter 3 for bill verification phase.
Enter 4 for all phases related with this device.
1
Please enter the number of RUs (i.e., N).
1
Please enter the number of tokens (i.e., K) for SMs.
10
4. Bill verification phase.
>>>>>SM: bill generated.<<<<<
Combained receipt K: [307410529597693636138237218058901590511636547253647704746202877691090186923604619112569800341578600864279213238617930 26
13504435381939171310687240763177 79, 628173339902603212662202613413591128821994130028306565703499307145630323808067362513132592945941046563261
244391150321650637802066223665934157302228912699 0]
>>>>>RU: receipts combained.<<<<<
>>>>>UC: bill has been settled.<<<<<
Computation time of SM at bill issuing phase: 331.812000 ms
root@NanoPC:~/Sipster_codes#
```

In summary, by varying the inputs' combinations, the program benchmarks the Sipster from different aspects. For example,

1. By using different combinations of Sipster role (device) and algorithm phase, the program gives the computation time of each algorithm phase running on the selected device, i.e., SM, RU, or UC.

2. By entering "4" in the algorithm phase selection, the program gives the computation time of all phases executed by the device.

3. We can also compare the results across different inputs of token number (i.e., K.) and the number of RUs (i.e., N).