Sipster: Settling IOU of Smart Meters Privately and Quickly

A. How to run the code?

It only requires three steps to run the Sipster. First, we need to install two tool kits, i.e., PBL and OpenSSL. We use ECDSA from OpenSSL. Its elliptic curve is over a prime field of n = 256 bits. We use the Pairing based Cryptography Library (PBL) with a "Type A" elliptic curve generated for 256-bit group order and 512-bit base field. Second, we need to download the source file from the code repository and compile the source files in the folder “Codes” using the GCC compiler. It gives a runnable output file "Sipster.run". Third, run the "./Sipster.run" to test the algorithm.

PBL: http://crypto.stanford.edu/pbc
OpenSSL:https://www.openssl.org/

We also include demo videos of running the code and verifying the results on GitHub. https://github.com/MobiSec-CSE-UTA/Sipster

B. 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.

C. How to remotely access our test environment?

Fig. 1 shows the experimental setups of Sipster: SM is built on a testbed with 1.5GHz ARM Cortex-A9 processor and 1GB RAM, reflecting the relatively weak computation power of a typical smart meter.

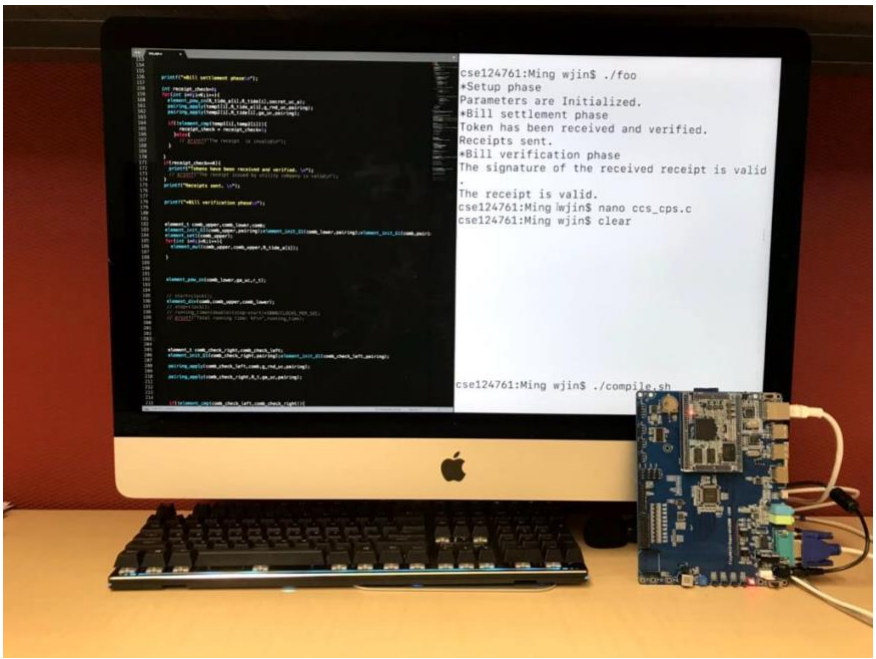
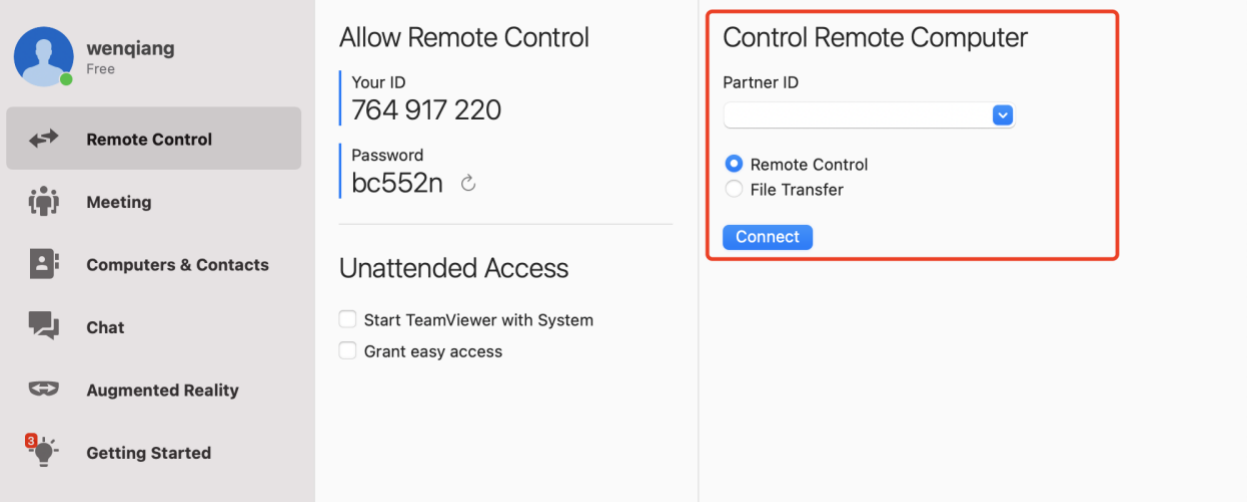


Fig. 1 An illustration of the experimental setup

We provide a remote access our test environment using the TeamViewer. The detailed instructions are included as follows.

After installing TeamViewer, the reviewer can enter the marked ID number in the “Partner ID” form-field as “1 090 974 46” and press “Connect”.



Then, the dialog box will pops up. The password needs to be entered as “g579zi”.

After login, the reviewer can observe two shells that control the SM and RU/UC respectively.

D. How to verify the results?

We provide interactive shell commands for the reviewers to verify our results. Taking the results of Figure 2(a) of section 5, the reviewer can verify the computation time of SM at different algorithm phases, i.e., bill issuing phase and bill verification phase, by simply following the instructions prompted by the program.

The program will first ask which algorithm phase the reviewer wants to test. By entering "1", the program will run the bill issuing phase and automatically generate the computation time, while insertion of "2" indicates that the bill verification phase will be tested. The program will also ask the number of tokens, i.e., K, for this SM. By varying the value of K from 10 to 100, we can get the results in Figure 2(a) of section 5.

Fig.2 and Fig.3 show how the program runs to verify computation results.

```
root@NanoPC:~/SM# ./Sipster.run
Please select the algorithm phase.
Enter 1 for bill issuing phase
Enter 2 for bill verification phase
1
Please enter the number of tokens (i.e., K) for this SM.
10
############### Algorithm begins ###############
1. Setup phase completed.
2. Bill issuing phase.
>>>>>SM: Token generated.<<<<<.
Computation time of bill issuing phase: 330.827000 ms
```

Fig. 2 Computation time of bill issuing phase on SM

```
root@NanoPC:~/SM# ./Sipster.run
Please select the algorithm phase.
Enter 1 for bill issuing phase
Enter 2 for bill verification phase
2
Please enter the number of tokens (i.e., K) for this SM.
10
############### Algorithm begins ###############
1. Setup phase completed.
2. Bill issuing phase.
>>>>>SM: Token generated.<<<<<.
3. Bill settlement phase.
>>>>>UC: Token signature verified.<<<<<
>>>>>UC: Receipt generated.<<<<<
>>>>>RU: Receipt verified.<<<<<
4. Bill verification phase.
>>>>>SM: bill generated.<<<<<
Computation time of bill verification phase: 4.703000 ms
root@NanoPC:~/SM#
```

Fig. 3 Computation time of bill verification phase on SM