Computation Integrity in Mobile Cloud: Running Probabilistically Checkable Proof on Android

Clinton Walker
Southern Polytechnic State University
Marietta, GA 30060
cwalker6@spsu.edu

Qijun Gu
Texas State University
San Marcos, TX 78666
qijun@txstate.edu

Terry Penner
LeTourneau University
Longview, TX 75607
TerryPenner@letu.edu

***Abstract*—Mobile cloud computing is a viable collaborative computing paradigm for the Internet of Things (IOT) to strengthen the synergy of mobile devices. The security of such collaborative computing needs much attention. This paper explores how to use Probabilistically Checkable Proof (PCP) on Android devices to verify if offloaded computation is correctly executed by remote devices. The paper shows that PCP could be viable for use in a mobile cloud environment. However, PCP may incur a significant overhead due to the low data throughput of the wireless network used by Android devices. The cost of overhead may outweigh any gains from verifying computation on remote Android devices. We conducted various experiments utilizing PCP in a mobile network of several Android devices and showed where PCP is beneficial and where improvements could be made to create an efficient and practical system.**

***Index Terms*—Probabilistically Checkable Proof, Android, Mobile Cloud**

I. Introduction

As the availability and capabilities of mobile devices continue to grow, mobile devices are offering rich computing capability with context and social awareness to the Internet of Things (IOT). Many recent works have studied various ways to create mobile cloud among mobile devices to harness their computing power [1], [2], [3]. The cloud provides a collaborative mobile computing environment by enabling partitioning, offloading and execution of applications among mobile devices. This new computing paradigm is expected to reduce overall power consumption and execution time of mobile devices.

The integrity of offloaded computation is a main security concern in mobile cloud. Malicious mobile devices may not execute the offloaded code completely or correctly and may forge arbitrary results. Nevertheless, it is a very challenging problem to validate that remote devices have performed the offloaded computation correctly. The validating process is in general modeled as an interaction between a pair of devices called verifier and prover. The verifier challenges the prover with some randomly picked data sets. The prover computes a response based on the data sets and sends the response back. If the response matches the expectation of the verifier, the prover and the offloaded computation are integral.

Recently, Probabilistically Checkable Proof (PCP) [4], [5] was proposed as a practical solution to validating offloaded computation. Different from other methods, such as remote attestation [6], [7], [8], PCP directly verifies computation instead of code. Hence, PCP better ensures the integrity of computation and is promising to be applied to secure computation in cloud. A main merit of PCP is that it can significantly reduce the computation of verification to make verification practical. *It is critical that any verification must take less computation time than performing the original computation locally. Otherwise, it is not worth performing verification remotely.*

However, to the best of our knowledge, PCP is still backed by high-end clusters in earlier studies [4], [5], even though its computation is already greatly reduced. It is questionable if PCP is feasible to mobile devices that have less computing power. Therefore, the main purpose of this work is to investigate how PCP can be utilized in mobile cloud. This work contributes in the following aspects. (i) The core of PCP was implemented in Android devices to carry out preliminary experiments. The implementation has the complete sequence of offloading, execution, and verification. (ii) Various experiments were conducted to collect the data involved in offloaded computation and verification. It helps to show the main factors that influence the overall performance of PCP.

In the remainder of the paper, we will first describe how PCP is implemented and executed on Android devices in Section II. Then, we will discuss the experiments and findings of running PCP on Android devices in Section III. The related work on computation integrity is summarized in Section IV. The paper is concluded in Section V.

II. Running PCP on Android Devices

A. Background on PCP

Fig. 1 shows a general diagram of computation offloading and computation verification in mobile cloud. The complete process is modeled as four steps. A pair of devices are involved in the process: verifier and prover. The first two steps are for performing mobile cloud computing. The verifier offloads its computation C and the associated data x to the prover. Then, the prover executes the computation $C(x)$ and returns the result y. In the next two steps, the verifier wants to verify if the computation is correctly executed by the prover. The verifier sends a random query γ, which is computed based on the offloaded computation $\{C, x, y\}$, to the prover. The prover

978-1-936968-92-3 © 2013 ICST
DOI 10.4108/icst.collaboratecom.2013.254081

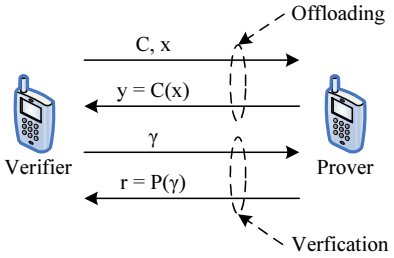

Fig. 1. Diagram of Offloading and Verification

computes a response $r = P(\gamma)$ and sends it back. The verifier verifies the result of the offloaded computation by evaluating the correctness of the response r.

PCP models the offloaded computation C as an arithmetic circuit and maps the signals on the circuit to the data used in computation. Then, it creates a set of constraints over the circuit. It verifies the results of the offloaded computation by evaluating if the constraints are satisfied by the signals. Following the study in [4], we use the same offloaded computation, i.e. matrix multiplication, where the data x is of two matrices A and B and the computation C is $C = A \cdot B$. The constraints for the arithmetic circuit of matrix multiplication are below, where i, j are the subscript indices of matrix elements and z is a vector of circuit signals satisfying all constraints.

$$Q^a_{i,j} = z^a_{i,j} - A_{i,j} = 0$$
$$Q^b_{i,j} = z^b_{i,j} - B_{i,j} = 0$$
$$Q^c_{i,j} = C_{i,j} - \sum_{k=1}^{m} z^a_{i,k} z^b_{k,j} = 0$$

The satisfiability of the circuit is expressed as below, where v is a random vector selected by the verifier, and $\gamma =< \gamma_0, \gamma_1, \gamma_2 >$.

$$\begin{aligned} P(\gamma) &= \sum_{i,j} v^a_{i,j} Q^a_{i,j} + \sum_{i,j} v^b_{i,j} Q^b_{i,j} + \sum_{i,j} v^c_{i,j} Q^c_{i,j} \\ &= \gamma_0 + \gamma_1 \cdot z + \gamma_2 \cdot z \otimes z \end{aligned}$$

The prover holds z and $z \otimes z$, which are determined by A and B. The verifier randomly generates γ and then sends γ to the prover for computing $P(\gamma)$. Upon a matching response, the computation is accepted as valid. Readers can find more detailed information about PCP in [4].

B. Implementation with Android

The implementation of PCP in an Android environment uses two Nexus 7 tablets. One acts as the verifier and the other acts as the prover. The Wi-Fi Direct service in Android is used to create a peer-to-peer network for the two tablets. An Android application is developed to transmit data and control over the network between the two tablets as well as perform offloaded computation and verification. A testing computer is used to capture and consolidate data from the events using the Android SDK and Eclipse. The summary of the implementation configuration is in Table I.

TABLE I
IMPLEMENTATION CONFIGURATION

Android:	Nexus 7 Tablets
Testing computer:	Macbook Pro
Android IDE:	Eclipse Android SDK
Java Testing:	NetBeans7.3.1 IDE

Both tablets were loaded with testing programs that were designed to time stamp the major events in the offloaded computation and the PCP verification. These measurements are taken in both the verifier and the prover. The complete process includes the following eight steps.

- Step 1, *Data Transmission*: The verifier transfers matrices A and B to the prover.
- Step 2, *Matrix Multiplication*: The prover performs the matrix multiplication.
- Step 3, *Result Transmission*: The prover sends the result back to the verifier.
- Step 4, *Query Generation*: The verifier computes γ randomly.
- Step 5, *Query Transmission*: The verifier sends γ to the prover.
- Step 6, *Proof Computation*: The prover computes the proof $P(\gamma)$.
- Step 7, *Proof Transmission*: The prover sends the proof back to the verifier.
- Step 8, *Proof Check*: The verifier checks the proof against the satisfiability.

Note that the first three steps complete the process of computation offloading in Fig. 1. A subtle difference from a mobile cloud is that the code of the offloaded computation is already loaded as an Android application in the two tablets in our implementation. The reason of having this implementation is that the transmission of the code does not contribute observable overhead to the whole process. The remaining five steps represent the process of PCP verification in Fig. 1. Because steps 7 and 8 take a small constant amount of time, they are not studied in later experiments.

III. EXPERIMENTS AND FINDINGS

We conducted experiments on offloaded matrix multiplication and PCP verification in the mobile computing setting. The analysis on the experiment results will help us to identify (1) what are the most time-consuming steps in offloading and PCP, (2) what is the difference between the time of PCP and the time of executing the original matrix multiplication locally, and (3) which processes need to be improved.

A. Experimental Settings

In experiments, the matrix multiplication was executed over square matrices of 100×100, 300×300, and 600×600. Two sets of data were collected to study the overhead and performance. One is the *computation time* for matrix multiplication (step 2), query generation (step 4), and proof computation (step

TABLE II
COMPLEXITY ANALYSIS

Steps	Computation	Transmission
1. Data Transmission		$O(n^2)$
2. Matrix Multiplication	$O(n^3)$	
3. Result Transmission		$O(n^2)$
4. Query Generation	$O(n^2)$	
5. Query Transmission		$O(n^2)$
6. Proof Computation	$O(n^2)$	

$n \times n$ is the matrix size.

6). The other is the *transmission time* for data transmission (step 1), result transmission (step 3), and query transmission (step 5). The results described in this section are data collected from 25 runs of matrix multiplication for each matrix.

B. Analysis of Results

1) Complexity Analysis: Before discussing the experimental results, we first show the expected complexity in Table II regarding the two metrics: computation time and transmission time. It is noted that the computation time of matrix multiplication is one order of magnitude higher than other computation times, which justifies that PCP takes less computation time than performing the original computation. However, besides computation time, performing PCP also requires to transmit a large amount of data over the Wi-Fi network and store the data in Android devices. In mobile cloud, the wireless bandwidth is typically limited. Hence, it is necessary to find out how the network factor may affect the overall performance of PCP.

2) Decomposed Time Analysis: Fig. 2 shows the decomposed time, including both computation and transmission, for the first six steps of offloading and PCP. The itemized analysis on the time of these steps is detailed below.

a) Step 1, Data Transmission: This step involves transmitting two $n \times n$ matrices and results in $O(n^2)$ transmission time. The transmission of a 100×100-element matrix takes 150 milliseconds in average, while the transmission of a 600×600-element matrix needs 5.9 seconds.

b) Step 2, Matrix Multiplication: The prover performs the computation in this step, which needs $O(n^3)$ multiplications. This significant amount of computation is the reason that a device would want to offload the computation to another device in the cloud. In Fig. 2, we can observe that the greater the size of the matrix, the greater the gain obtain from offloading the computation.

c) Step 3, Result Transmission: The result of matrix multiplication is a single $n \times n$ matrix. Being half as large as the data set in step 1, the transmission time of this step was seen to be approximately half of the time in step 1.

d) Step 4, Query Generation: In this step, the verifier generates a random query. The size of the query is proportional to the size of the matrices. The generation time is thus $O(n^2)$. For 600×600-element matrix multiplication, the generation needs only 224 milliseconds. This is a prime indication that, if considering computation time alone, PCP could be a viable verification solution for mobile devices in mobile cloud.

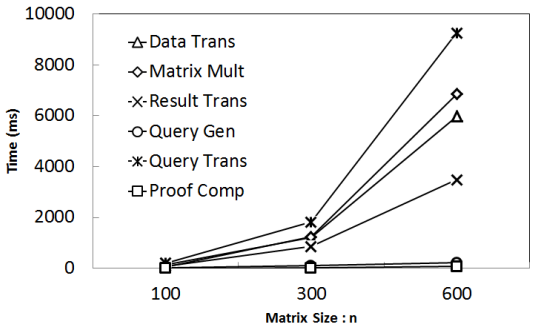

Fig. 2. Decomposed Time for Offloading and PCP

However, the computation time of PCP is significantly smaller than the transmission time in other steps.

e) Step 5, Query Transmission: According to the fundamental theory of PCP, the size of γ is the same to the number of signals z in the arithmetic circuit. For matrix multiplication, the set of signals are made of the two matrices A and B. Hence, the number of signals is proportional to the size of the matrices, and thus the size of γ and the transmission time of γ are $O(n^2)$. Observing Fig. 2, it is easy to see that transmission of the query is the most time-consuming step. Therefore, the transmission of the large amount of data in queries becomes the dominant factor in determining the overall performance of PCP.

f) Step 6, Proof Computation: This step executes the dot product of γ and z. So, the computation time is $O(n^2)$. Similar to step 4, this step of PCP does not incur much overhead. For 600×600-element matrix multiplication, this step only needs 73 milliseconds.

3) Findings: The results of the collected data show where the strength and the weakness are in PCP. The strength lies in the reduced amount of computation time of generating the query and computing the proof. However, the overhead of transmitting the query creates a situation where PCP may be too intensive to warrant verifying computation with remote devices.

a) Advantage of PCP: Fig. 3 shows the PCP's computation time for query generation and proof computation in comparison to the time of matrix multiplication. Obviously, PCP uses much less computation time than executing the matrix multiplication locally. In addition, the complexity analysis in Table II also shows that PCP's complexity is one order of magnitude smaller than matrix multiplication. Hence, the desirable computational feature of PCP is well captured by the analysis and the experimental results.

b) Lags in Mobile Network: Unfortunately, Fig. 3 shows that the query transmission time is significantly greater than the computation time of PCP, and dominates the overall performance of PCP. For 600×600-element matrix, it needs 9.55 seconds for completing the PCP process, but only 6.85 seconds for matrix multiplication. Hence, in our experimental setting, performing the original matrix multiplication locally to verify the result is better than using PCP. Such a significant

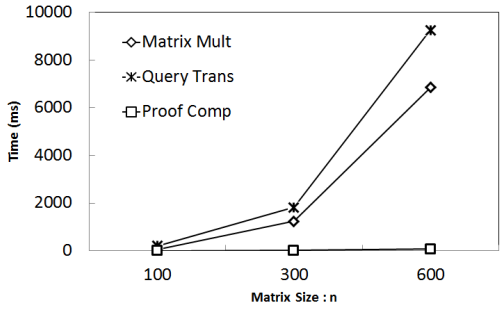

Fig. 3. Comparison of PCP and Matrix Multiplication

transmission overhead of PCP is due to the low throughput of the Wi-Fi connection, which was less than 4.7Mbps in our measurement.

c) Turning Points of PCP: Based on our experiments and analysis, PCP can be beneficial to mobile cloud in either of the two scenarios in practice. In the first scenario, the network throughput should be sufficiently high so that the transmission time is comparable to or smaller than the computation time. Based on our experiments, this will require the throughput to be greater than 100Mbps. Such a high throughput wireless network is only achievable with IEEE 802.11n currently.

In the second scenario, the computation time of matrix multiplication should be significantly larger than the PCP time. According to the complexity analysis in Table II, the complexity of matrix multiplication is one order of magnitude higher than PCP. If the size of matrix grows enough, the time of matrix multiplication will surpass PCP eventually. Using regression, we have the following estimation based on the experimental results. The time of matrix multiplication is approximate to $0.00003193n^3$, and the time of PCP is approximate to $0.0262n^2$. Therefore, in our experiments, if the matrix size n grows above 820, the time of matrix multiplication will be greater than the time of PCP. However, the computation on such a big matrix crashed in the device, because its memory need exceeded the heap limit in Android.

IV. Related Work

To ensure the integrity of remote computation, several ideas were studied in the past. One is remote attestation [6], [8], [7], in which the status of remote devices and the integrity of their code are verified. The core ideas are to randomly send time-bounded probes to the remote devices and request them to return the content of the probed code memory. If the remote devices can respond timely with correct memory content, they are considered intact. Such remote attestation can only work with pre-determined code memory space, and thus will not work in mobile clouds, because receptors allocate the storage of the offloaded code at run time. Another type of remote attestation relies on physical unclonable functions (PUF) in remote devices [9]. It computes PUF-based hardware signature. The verifier challenges the prover with a random computation. The prover embeds the signature with the result of the computation in response. The verifier can believe the result if the embedded signature matches the signature of PUF. This remote attestation is not applicable in cloud as well, because it asks the originator to have the signature of the physical unclonable functions of other devices in advance.

A recent development is PCP [4], [5]. Different from remote attestation, PCP verifies computation instead of code and does not rely on any special hardware feature. It is promising to apply PCP in mobile cloud, although PCP may be limited in a few aspects for mobile cloud. One is the computation overhead, which is determined by the complexity of the arithmetic circuit of the offloaded computation. When the computation is complicated, the corresponding circuit size will grow significantly, which results in a demanding overhead. Another limitation is its applicability to arithmetic operations. So far, PCP is still costly to inequality comparisons and branch [5]. Our study shows that beside its computation overhead, the network overhead shall be considered as well if using PCP in mobile cloud.

V. Conclusions

In this work, we implemented and experimented PCP on Android devices to test the feasibility and limitation of PCP in a mobile cloud setting. We found that, given the scale of computation that the Android devices can accommodate, the gain from the PCP's computation reduction may be offset by the overhead of transmitting the data (namely, query γ) used in PCP. Using PCP in mobile cloud will only be beneficial to mobile cloud if either the wireless network has a sufficiently high bandwidth to reduce the transmission time, or the device can accommodate more demanding computation to surpass the transmission overhead.

Acknowledgment

This research is funded by the NSF award #1156712 that is co-funded by the DoD.

References

[1] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *Proc. of ACM Workshop on Mobile Cloud Computing Services*, 2010, pp. 1–6.

[2] M. Satyanarayanan, "Mobile computing: the next decade," in *Proc. of ACM Workshop on Mobile Cloud Computing Services*, 2010.

[3] E. Miluzzo, R. Cáceres, and Y.-F. Chen, "Vision: mClouds - computing on clouds of mobile devices," in *Proc. of the ACM workshop on Mobile cloud computing and services*, 2012, pp. 9–14.

[4] S. Setty, R. McPherson, A. J. Blumberg, and M. Walfish, "Making argument systems for outsourced computation practical (sometimes)," in *Proc. of NDSS*, 2012.

[5] S. Setty, V. Vu, N. Panpalia, B. Braun, A. J. Blumberg, and M. Walfish, "Taking proof-based verified computation a few steps closer to practicality," in *Proc. of USENIX Security*, 2012.

[6] A. Seshadri, A. Perrig, L. v. Doorn, and P. Khosla, "SWATT: SoftWare-based ATTestation for Embedded Devices," in *Proc. of IEEE Symposium on Security and Privacy*, 2004, pp. 272–284.

[7] M. Shaneck, K. Mahadevan, V. Kher, and Y. Kim, "Remote Software-Based Attestation for Wireless Sensors," in *Security and Privacy in Ad-hoc and Sensor Networks in LNCS*. Springer, 2005, pp. 27–41.

[8] C. Castelluccia, A. Francillon, D. Perito, and C. Soriente, "On the difficulty of software-based attestation of embedded devices," in *Proc. of ACM CCS*, 2009, pp. 400–409.

[9] S. Schulz, A.-R. Sadeghi, and C. Wachsmann, "Lightweight remote attestation using physical functions," in *Proc. of ACM Wisec*, 2011, pp. 109–114.