Application of ICmetrics for Embedded System Security

Xiaojun Zhai, Kofi Appiah, Shoaib Ehsan, Huosheng Hu, Dongbing Gu, Klaus McDonald-Maier and Wah M Cheung
School of Computer Science & Electronic Engineering
University of Essex, Colchester, UK
{xzhai, kappiah, sehsan, hhu, dgu, kdm, wmcheu}@essex.ac.uk

Gareth Howells
School of Engineering and Digital Arts
University of Kent
Canterbury, UK
W.G.J.Howells@kent.ac.uk

***Abstract*—Integrated Circuit Metrics (ICmetrics) technology is concerned with the extraction of measurable features of an embedded system, capable of uniquely identifying the system's behaviour. Any changes in these identifiers (profiles) during consequent devices' operation would signal about a possible safety or security breach within the electronic system. This paper explores the combination of program counter (PC) and Cycles per Instructions (CPI) of a processor core as a potential ICmetrics source for embedded system security. The use of this combination exhibits that while isolated PC values may not always generate a stable identifier (profile) for a device that would distinguish the device from the rest in a considered set, the PC and CPI sequences and frequencies in the execution flow may serve as suitable ICmetrics features.**

Keywords—ICmetrics; embedded system security; program counter; Cycles per Instructions;

I. INTRODUCTION

ICmetrics (Integrated Circuit metrics) is a technology that can generate unique identifiers from the distinctive characteristics of the software and hardware associated with a particular electronic device [1]. Based on internal and external sensors, electronic devices sense different environmental condition, trigger the execution of different software, perform different tasks and even interact differently with different users. Various features can be extracted from digital devices' operation that may be integrated together to generate unique identifiers for each of the devices or create unique profiles that describe the devices' actual behaviour. The generated identifiers can then be used to create encryption keys for embedded devices enabling secure encrypted communication. This technology does not only ensure secure data storage and transmission, but also helps to detect failure, tampering and malicious exploitation of electronic devices [2].

Similar to features (biometrics) that is closely linked or can be characterized any particular human being, be it Iris, DNA or fingerprint; ICmetrics is the device equivalent for characterizing an electronic device. Compared to other popular techniques for electronic system, such as the Physical Unclonable Functions (PUF) technologies [3], hardware intrinsic security [4], biometrics [5], encryption and password [6], the ICmetrics technology can be considered as a hybrid approach as it exploits features derived which are extracted based on the interaction of the hardware with their users and/or environment.

In the previous works [7] and [8], the authors aimed to explore the possibility of applying the ICmetrics technology to generate stable encryption keys based on characteristics derived from embedded systems' operation. The program counter (PC) of a processor core was used as a source for ICmetrics features, where full PC profiles were analysed to find the distinct values from different programs. Kovalchuk et. al. [9] presents some interesting properties of a processor that can be used as ICmetrics. They explored samples of PC and used it as ICmetric to determine the effects on systems stability and performance. The number of occurrence of a particular PC value is used to address how different applications utilize the memory space. Their result shows that there are certain areas in the memory space, occupied by a single application.

Similar to [9], Kovalchuk et. al. [8] identified the PC as a potential ICmetric. The paper focused on possibilities of extracting the PC values with little or no effect on the actual program execution. They also suggested the PC may hold interesting features to complement other parameters for a robust ICmetric. Test conducted shows some level of identity between simple programs in terms of sequence and frequency of the PC values. Another interesting observation from the paper is the fact that PC values remain the same for a particular application when extracted by single stepping or sampling tracing.

Reiss and Renieris [10] present a means of visualizing and understanding the dynamic behaviour of large complex systems using data collected as the program runs. They first select subsets of the raw tracing data, which is normally voluminous and then apply run-length encoding in an attempt to infer its structure. Gniady et. al. [11] explores the viability of applying program counter-based prediction techniques to optimize buffer caching. Their technique allows the operating system to correlate the input and output operations with the program context in which they are triggered. This paper explores the combination of PC and Cycles per Instructions (CPI) as a potential source of ICmetric features. The paper is

organised as follows. Section II gives an overview of the ICmetric technology, followed by sections III and IV with detailed properties exhibited by the PC and CPI respectively. Section V, concludes with the potentials of combining PC and CPI as an ICmetric feature, with future directions.

II. ICMETRICS TECHNOLOGY BASED EMBEDDED SYSTEM SECURITY SOLUTION

The ICmetrics technology is mainly based on measurement of potential features that are derived from characteristics of an embedded system under different circumstances. By analysing and employing the features, a unique identifier can be generated and used to describe or determine the embedded system. Typically, the circumstances of an embedded system can change by running different software or interacting with different users.

In order to calibrate an embedded system for a specific circumstance, a calibration phases is applied once while an application is running for the first time, where the features of this application are extracted and recorded by the ICmetrics. Those recorded unique features can then be integrated as a unique identifier for the purpose of encryption. In this case, while different applications or circumstances are employed in this embedded system, the different features should then be generated, which means based on these new features, a distinct encryption key should be finally created. As the keys are mismatched, the system will stop working to protect the user data. Based on the above idea, two phases of the ICmetrics system are required: 1) calibration phase, where features and characteristics of an embedded system are recoded and analysed. 2) Operational phase, where unique identifier is generated for the key generation.

The overall ICmetrics technology based embedded system diagram is shown below in Fig. 1.

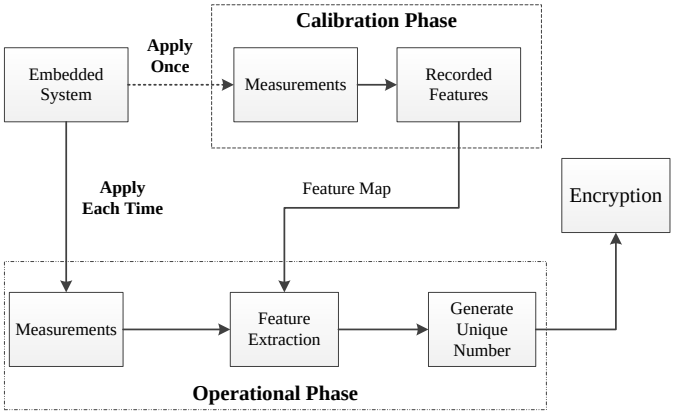


Fig. 1. A block diagram of a typical ICmetrics based embedded system.

In the current ICmetrics based embedded system, identifying the devices' characteristics and generating suitable features for the ICmetrics system are the main challenges in this domain [7]. The suitable features must represent the characteristics of the embedded system, and the feature extraction and analysing processes should not significantly affect the performance of the embedded system. Considering the above two points, the PC and CPI are introduced as possible ICmetrics features in the following sections.

III. PROPOTIES OF PC

Depending on the platform and size of memory used, the PC values of a program can be logged during its execution. The PC values can also be significantly huge. A typical execution of the angle conversion program [8] on an ARM9 Cortex - M3 processor can have PC values ranging from *134218632* to *134223590*, using an intrusive tracing method. For a single execution, the PC profile of an application can range from ten to hundred thousand in a second. The profile also exhibits some characteristics distinct to the very application in execution. The features may be more dominant over time or space; access to specific memory location. The spatial and temporal features exhibited by four algorithms chosen from the automotive package of the MiBench suite of benchmark algorithms [8], namely: angle conversion (AC); bit count (BC); cubic function (CF); and square roots (SR) are shown in figures 2 and 3.

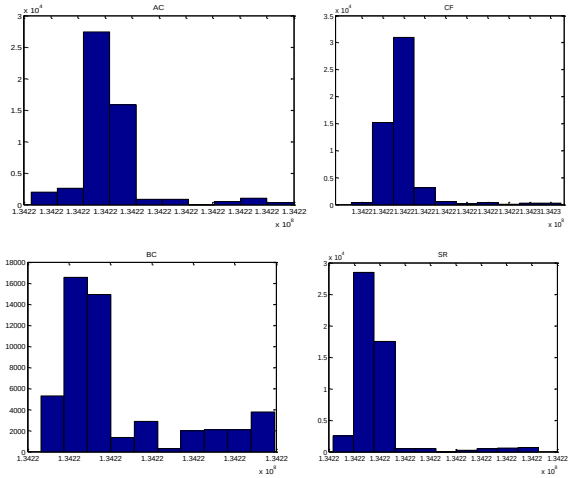


Fig. 2. Spartial PC distribution for the four different applications.

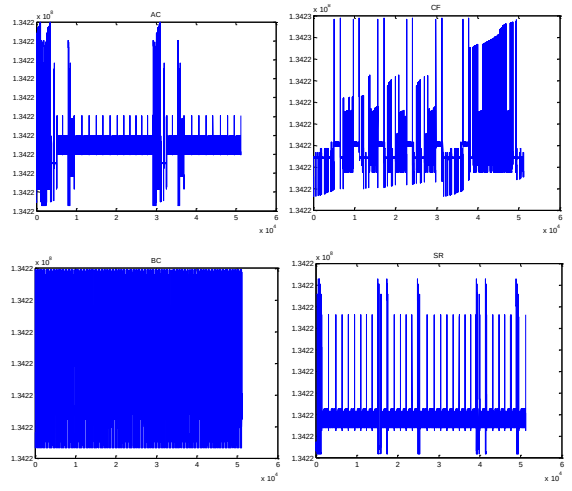


Fig. 3. Temporal PC distribution for the four different applications.

The spatial distribution shows how similar application may have common PC values. The only distinguishing feature is the frequency of use of a particular address space, which can only be estimated after the program, has run to completion. In the contrary, the temporal distribution shows the pattern of use of address space over a period of time, and can further be refined to generate ICmetric features during the program execution.

IV. Propoties of CPI

In embedded system, processing throughput is one aspect of a processor's important parameter. Typically, CPI or its multiplicative inverse of Instructions per Cycle (IPC) is used to represent a processor's performance [12]. Most computers or embedded systems run synchronously at a constant clock rate, the CPU clock rate depends on the specific CPU architecture and its hardware implementation technology used. A program is comprised of a number of micro operations which depend on the instruction sets and the exact CPU architecture that are used in the embedded system. For example, the well-known CPU design strategy: Reduced Instruction Set Computing (RISC) normally has 5 stages:

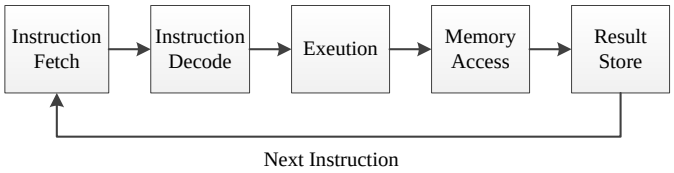


Fig. 4. A block diagram of a typical RISC processing steps.

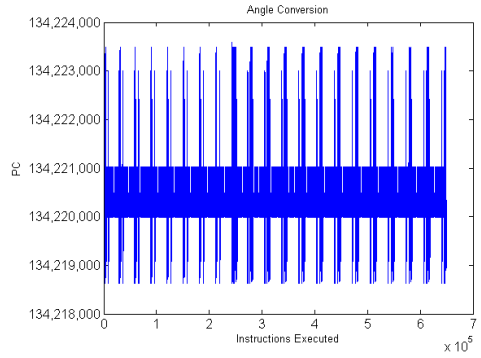
In Fig. 4, each stage needs one clock cycle for passing an instruction through the stages sequentially. The number of CPU clock cycles for each instruction varies in the used hardware architectures. For example, without pipeline strategy, a new instruction is fetched in the first stage at least 5 clock cycles, but a new instruction may be fetched every cycle if the pipeline strategy is used. In addition, some of instructions require multi-cycle to be executed as they need access to memory during the processing (e.g. Load, Store and Jump). Therefore, the number of CPU cycles needed for a single instruction to be executed in a computer system is termed as CPI. A program normally consists of varying instructions, which means the program should have varying CPI values during execution. Thus, average CPI value that can be used to describe the system performance varies in a specified period of execution. The average CPI of an embedded processor can be calculated as given in (1) [13]:

$$CPI = \frac{T \times f_{\max}}{I} \tag{1}$$

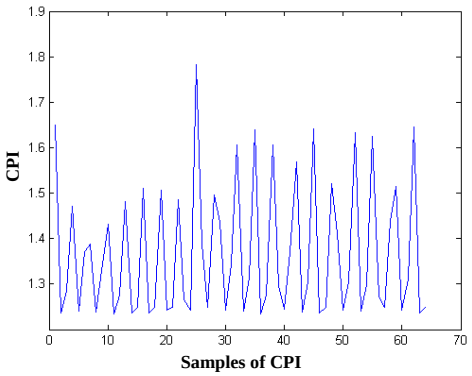
where I is the total number of instructions, T is time consumption while executing the total number of instructions, and $f_{\max}$ is the maximum clock frequency of an embedded processor.

Although PC can exhibit some properties of a program, a large number of data need to be processed for the feature extraction. In this case, if all the PC data are analysed, the system performance may degrade. Therefore, according to the discussion of the CPI properties, CPI can be an alternative feature for the ICmetrics system.

Fig. 5 shows an example of PC and CPI diagrams of a program executed in ARM based embedded system.



(a)



(b)

Fig. 5. Example of PC and CPI diagrams. (a) PC. (b) CPI.

In Fig. 5, each point on graph (a) represents an executed PC value, and each point on graph (b) represents average CPI value taken over every 10,000 instructions executed. From the point of view of processing data volume, (using the two graphs), the CPI graph only contains 0.01% data volume of the PC graph. This could significantly reduce the processing time for the analysis of ICmetrics features. From the point of view of feature extraction, although the CPI graph has much less information compare to the PC graph, the main features of the executed program is remained. For example, the peaks PC values normally indicate where the branch jump instructions occur, which means certain function or condition are called or triggered. Therefore, it is possible to obtain enough information to identify the executed program from these points. As can be seen from Fig.5 (b), the CPI graph still remains the main characteristics of the program, especially, the relationship of these peaks are retained.

In Fig.5 (b), every single average CPI value is taken over every 10,000 instructions per interval. However, this interval number can be chosen differently. If fewer instructions are taken, more details of the program can be obtained. Fig. 6 illustrates several average CPI graphs for the same program when using different intervals.

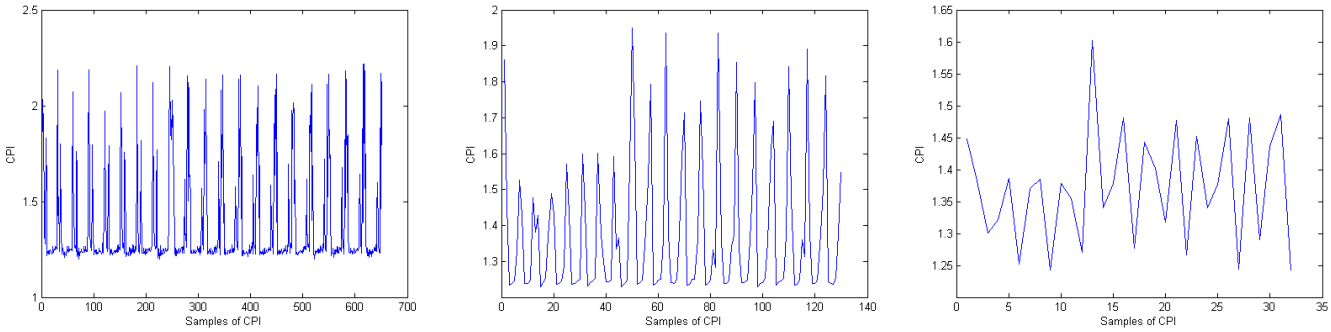


Fig. 6. Average CPI graphs for the same program with different intervals. (a) interval set to 1,000. (b) interval set to 5,000. (c) interval set to 20,000.

As can be seen from Fig. 6, while using different intervals, the average CPI graphs for the same program are significantly different. Generally, more details of the program's behaviour can be obtained if less interval value is used, for example, those branch jump points are not present in Fig. 6 (b) and (c) but can be detected in Fig. 6 (a). According to this property, we could explore the sizes of the intervals to obtain the best suitable value for a specific program. Once a fixed interval is found, the feature extraction for the ICmetrics system would benefit from the balance of the system accuracy and performance.

V. Conclusion

The ICmetrics technology can automatically extract and analyse the features derived from an embedded system, and use these features to generate unique basic number for the purpose of encryption. The embedded system could benefit from it by automatically detecting frauds and protecting data communication during the system run time.

PC alone is not enough to segment the individual phases of a program in execution. The peaks extracted from the CPI maps directly into the PC profile, which identifies a particular phase within a program. The phase is an interval within a program that has similar behaviour. By analysing similar phases, a unique basic number can be generated and subsequently used as encryption key in the ICmetrics security system. If an automated approach can be developed capable of identifying these phases, it would make it possible to extract information about how a program is changing in a way that could generalise to all embedded systems.

Acknowledgment

The authors gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council under grant EP/K004638/1 and the EU Interreg IV A 2 Mers Seas Zeeën Cross-border Cooperation Programme – SYSIASS project: Autonomous and Intelligent Healthcare System (project's website http://www.sysiass.eu/).

References

[1] Y. Kovalchuk, K. D. McDonald-Maier, and G. Howells, "Overview of ICmetrics technology-security infrastructure for autonomous and intelligent healthcare system," International Journal of u- and e- Sevice, Science and Technology, vol. 4, pp. 49-60, 2011.

[2] M.Rahmatian, H. Kooti, I. G. Harris, and E. Bozorgzadeh, "Hardware Assisted Detection of Malicious Software in Embedded Systems," IEEE Embedded Systems Letters, vol. 4, pp. 94-97, 2012.

[3] G. E. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in 44th ACM/IEEE Design Automation Conference, 2007, pp. 9-14.

[4] H. Handschuh, G.-J. Schrijen, and P. Tuyls, "Hardware Intrinsic Security from Physically Unclonable Functions," in Towards Hardware-Intrinsic Security, A.-R. Sadeghi and D. Naccache, Eds., ed: Springer Berlin Heidelberg, 2010, pp. 39-53.

[5] A. K. Jain, P. Flynn, and A. Ross, Handbook of Biometrics: Springer US, 2008.

[6] W. Sheng, G. Howells, M. C. Fairhurst, F. Deravi, and K. Harmer, "Consensus Fingerprint Matching with Genetically Optimised Approach," Pattern Recognition, vol. 42, pp. 1399-1407, 2009.

[7] Y. Kovalchuk, W. G. J. Howells, H. Hu, D. Gu, and K. D. McDonald-Maier, "A practical proposal for ensuring the provenance of hardware devices and their safe operation," in 7th IET International Conference on System Safety, incorporating the Cyber Security Conference, 2012, pp. 1-6.

[8] Y. Kovalchuk, W. G. J. Howells, H. Hu, D. Gu, and K. D. McDonald-Maier, "ICmetrics for low resource embedded systems," in the 3rd International Conference on Emerging Security Technologies, 2012, pp. 121 - 126.

[9] Y. Kovalchuk, H. Huosheng, G. Dongbing, K. McDonald-Maier, D. Newman, S. Kelly, et al., "Investigation of Properties of ICmetrics Features," in the 3rd International Conference on Emerging Security Technologies (EST) 2012, pp. 115-120.

[10] S. P. Reiss and M. Renieris, "Encoding program executions," in the 23rd International Conference on Software Engineering, 2001, pp. 221-230.

[11] C. Gniady, A. R. Butt, and Y. C. Hu, "Program-counter-based pattern classification in buffer caching," in *the 6th conference on Symposium on Opearting Systems Design & Implementation*, 2004, pp. 27.

[12] K. Hwang and N. Jotwani, Advanced Computer Architecture: Tata McGraw-Hill Education, 2010.

[13] M. Annavaram, R. Rakvic, M. Polito, J. Bouguet, R. Hankins, and B. Davies, "The fuzzy correlation between code and performance predictability," in the 37th International Symposium on Microarchitecture (MICRO), 2004, pp. 93-104.