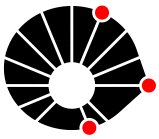
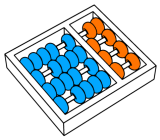




Anais do XII WTD

Equipe do XII WTD - 2017

Technical Report - IC-17-17 - Relatório Técnico
August - 2017 - Agosto

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Anais do XII Workshop de Teses, Dissertações e Trabalhos de Iniciação Científica

Profa. Ariadne Maria Brito Rizzoni Profa. Esther Luna Colombini
Profa. Juliana Freitag Borin Allan da Silva Pinto
Amanda Cristina Davi Resende Carlos Alberto Petry Eliana Alves Moreira
Elisangela Silva dos Santos Ícaro Cavalcante Dourado
Joana Esther Gonzales Malaverri Lucas Augusto Carvalho
Márcio de Carvalho Saraiva Priscila Aparecida de Moraes Ioris
Sergio Zumpano Arnosti

Resumo

Este relatório técnico contém os resumos de 07 trabalhos autorizados a serem publicados de um total de 34 trabalhos apresentados no XII Workshop de Teses, Dissertações e Trabalhos de Iniciação Científica (WTD), do Instituto de Computação (IC) da Universidade Estadual de Campinas (Unicamp) do ano de 2017. O Workshop ocorreu entre os dias 31 de Julho e 2 de Agosto de 2017 e contou com cerca de 210 participantes, entre ouvintes, apresentadores de trabalhos e organizadores. Na ocasião foram realizadas 28 apresentações orais e uma sessão de Pôsteres com 06 trabalhos. Aos alunos foi dada a possibilidade de escolher a forma de apresentação (oral, pôster), bem como escolher se desejasse publicar o seu trabalho nos anais do evento. A publicação dos resumos, sob forma de relatório técnico, tem por objetivo divulgar os trabalhos em andamento e registrar, de forma sucinta, o estado da arte da pesquisa do Instituto de Computação no ano de 2017.

1 Apresentação

Este relatório técnico contém resumos de 07 trabalhos cujos artigos foram autorizados a serem publicados no *XII Workshop* de Teses, Dissertações e Trabalhos de Iniciação Científica (WTD), do Instituto de Computação da Universidade Estadual de Campinas (Unicamp), edição 2017.

O *XII Workshop* ocorreu entre os dias 31 de Julho e 2 de Agosto de 2017, contando com cerca de 210 participantes, entre ouvintes, apresentadores de trabalhos e organizadores. Durante o WTD foram realizadas 28 apresentações orais e uma sessão pôster com 06 trabalhos. Aos alunos foi dada a possibilidade de escolher a forma de apresentação (oral ou pôster), bem como autorizar a publicação ou não de seu trabalho nos anais do WTD. A publicação dos resumos, sob forma de relatório técnico, tem por objetivo divulgar os trabalhos em andamento e concluídos além de registrar, de forma sucinta, o estado da arte da pesquisa do Instituto de Computação no ano de 2017.

Durante o XII WTD ocoreram os seguintes eventos:

- no dia 31 de Agosto a palestra de abertura do evento no Centro de Convenções da Unicamp, proferida pelo Professor Doutor André Fujita, do Instituto de Matemática e Estatística (IME) USP;

- no dia 01 de Agosto as seguintes atividades:
  - minicurso intitulado "Pesquisa Bibliográfica e Análise de dados", ministrado pelo Professor Doutor Jacques Wainer, do Instituto de Computação da Unicamp;
  - apresentações de trabalhos de iniciação científica;
  - apresentação da dissertação de mestrado e da tese de doutorado que venceram o Concurso de Teses e Dissertações (CTD) do Instituto de Computação da Unicamp, edição 2017;
- no dia 2 de Agosto as apresentações orais de trabalhos e sessão de posterês, realizadas nas salas 351, 352 e 353 do Instituto de Computação da Unicamp.

Agradecimentos.

Aos alunos que participaram do evento, em particular àqueles que se dispuseram a apresentar seus trabalhos, tanto de forma oral como na forma de pôsteres, bem como aos orientadores que os incentivaram a fazê-lo.

Aos professores e alunos de mestrado e doutorado do Instituto de Computação da Unicamp que compuseram as bancas de avaliação dos trabalhos.

Ao Professor Doutor Rodolfo Jardim de Azevedo, diretor do Instituto de Computação da Unicamp, e ao Professor Doutor Guilherme Pimentel Telles, coordenador do programa de Pós-Graduação, pelo incentivo, apoio e patrocínio providos ao evento.

Finalmente, aos alunos do programa de Pós-Graduação do Instituto de Computação da Unicamp que efetivamente organizaram e promoveram este evento, aos quais dedicamos o XII *Workshop* de Teses, Dissertações e Trabalhos de Iniciação Científica do Instituto de Computação da Unicamp:

• Allan da Silva Pinto, Amanda Cristina Davi Resende, Carlos Alberto Petry, Eliana Alves Moreira, Elisangela Silva dos Santos, Ícaro Cavalcante Dourado, Joana Esther Gonzales Malaverri, Lucas Augusto Carvalho, Márcio de Carvalho Saraiva, Priscila Aparecida de Moraes Ioris e Sergio Zumpano Arnosti.

Dra. Ariadne Maria Brito Rizzoni Carvalho
Dra. Esther Luna Colombini
Dra. Juliana Freitag Borin
Coordenadoras do XII WTD
Professoras do Instituto de Computação da Unicamp

Sumário

2 Programação

> "Viver é enfrentar um problema atrás do outro. O modo como você o encara é que faz a diferença."
>
> Benjamin Franklin

Neste capítulo, apresentamos a programação e alguns dados estatísticos do *XII Workshop de Teses, Dissertações e Trabalhos de Iniciação Científica* (WTD) do Instituto de Computação da Unicamp.

As tabelas a seguir aperesentam a programação completa ocorrida no XII WTD, disponível também em `http://ic.unicamp.br/wtd/2017/`.

Na edição deste ano, ocorreu no dia 31 de Julho a palestra de abertura no Centro de Convenções da Unicamp, proferida pelo Professor Doutor André Fujita do IME-USP, Tabela 1.

Tabela 1: Programação do dia 31 de Julho do XII WTD, realizada no Auditório do IC e Centro de Convenções da Unicamp.

Dia 31 de Julho de 2017		
Horário	**Local**	**Atividade**
13:00-15:30	Auditório do IC	Palestra aos ingressantes da pós-graduação do IC
17:00-18:00	Centro de Convenções	Palestra de abertura: Prof. Dr. André Fujita (IME-USP)

No dia 01 de Agosto, ocorreram as apresentações dos trabalhos de Iniciação Cinentífica e as apresentações da Tese de Doutrorado, da Dissertaão de Mestrado e do Trabalho de Iniciação Científica premiados no Concurso de Teses e Dissertações (CTD) de 2017 do Instituto de Computação da Unicamp. Ocorreu também o minicurso institulado *"Pesquisa Bibliográfica e Análise de dados"*, ministrado pelo Professor Doutor Jacques Wainer do Insituto de Cumputação, Unicamp, Tabela 2.

Tabela 2: Programação do dia 01 de Agosto do XII WTD, realizada no Auditório do IC.

Dia 01 de Agosto de 2017		
Horário	**Local**	**Atividade**
09:00-11:30	**Auditório do IC**	**Apresentação de trabalhos de iniciação científica** **Apresetnação de trabalhos premiados no CTD**
Aluno(a)	**Orientador(a)**	**Título do trabalho de Iniciação Científica**
Erik Perillo	Esther Colombini	Efficient Visual Attention with Deep Learning
Rodrigo Surita	Guido Araújo	Cylindrical Reconvergence Physical Unclonable Function
Luiz Fernando R. da Fonseca	Hélio Pedrini	Estabilização Digital de Vídeos
Nicholas Torres Okita	Edson Borin	Desenvolvimento de um critério de parada para o CRS Differential Evolution
Yuri Soares	Pedro Rezende	Computing Short Edge-Flipping Sequences Between Triangulations: a Heuristic Approach
Vinicius Balbino	Lehilton L. C. Pedrosa	Uma Aproximação para o Problema de Alocação de Terminais com Capacidade
Aluno(a) **Orientador(a)**	**Premiação**	**Título do trabalho premiado no CTD**
Allan Mariano de Souza Leandro Aparecido Villas	Dissertação de Mestrado	Um Sistema de Transporte Inteligente para Detecção e Controle de Congestionamento de Veículos Utilizando em Redes Veiculares
Anselmo Castelo Branco Ferreira Anderson Rocha	Tese de Doutorado	Multi-Analysis Techniques for Digital Image Forensics
14:00-17:00	**Minicurso**	**Título**
Prof. Dr. Jacques Wainer	Auditório do IC	Pesquisa Bibliográfica e Análise de dados

No dia 02 de Agosto ocorreram as apresentações orais de trabalhos nas salas 351, 352 e 353 e no Hall a de posterês nas dependência do IC 3.5 do Instituto de Computação, cuja programação está detalhada nas Tabelas 4, 5 e 6. Os trabalhos foram avaliados por bancas formadas por dois professores e um aluno de doutorado do Instituto de Computação, os alunos também tiveram participação como *chairs* das sessões.

O XII WTD contou com um total de 34 participações, sendo 28 apresentações orais e 06 participações na de pôsteres.

A Tabela 3 apresenta a nominata dos professores, pós-doutorandos e alunos mestrandos e doutorandos que compuseram as bancas de apresentação oral ocorridas nas salas 351, 352 e 353 do Instituto de Computação IC-3.5 no dia 02 de Agosto de 2017. A coluna mais a direita da tabela apresenta os nomes dos alunos mestrandos e doutorandos que participaram como Chairs das seções.

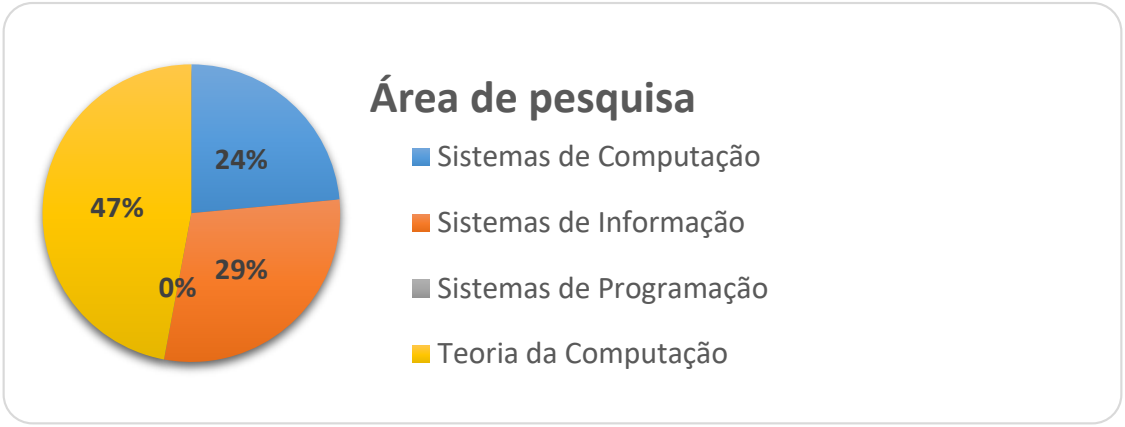
Tabela 3: Membros de bancas e Chairs das apresentações orais do XII WTD realizadas na salas 351, 352 e 353 do Instituto de Computação (IC-3.5), dia 02 de Agosto de 2017.

Membros de bancas e Chairs das apresentações orais		
Bancas: Professores e Pós-doutorandos	**Bancas: Alunos**	**Chairs: Alunos**
Alexandre Mello Ferreira	Allan Pinto	André Carvalho
Breno Bernard Nicolau de França	Armando Faz	Carla Negri Lintzmayer
Christiane Neme Campos	Breno Mendes	Celso Aimbire
Cid Carvalho de Souza	Elaine Hayashi	Elisângela da Silva
Diego de Freitas Aranha	Fagner Leal	Fernanda Andalo
Eliane Martins	Hayato Fujii	Helder May
Fábio Luiz Usberti	Juan Hernández	Juan Hernández
Flávio Keidi Miyazawa	Kleber Andrade	Marcelo Benedito
Gerberth Adín Ramírez Rivera	Luan Cardoso	Maycon Sambinelli
Guilherme Pimentel Telles	Lucas Melo	Paulo Finardi
Heiko Horst Hornung	Marcelo Palma	Rodrigo A. C. da Silva
Julio Cesar dos Reis	Natanael Ramos	
Leandro Aparecido Villas	Samuel Fadel	
Lehilton Lelis Chaves Pedrosa	Tiago Pedroso	
Orlando Lee	Yuri Soares	
Paulo Lício de Geus		
Rafael Crivellari Saliba Schouery		
Ricardo Caceffo		
Zanoni Dias		

A distribuição dos trabalhos com realação as quatro áreas de pesquisa do Instituto de Computação (Figura 2) estão assim distribuídas: 08 trabalhos para Sistemas de Computação, 10 para Sistemas de Informação e 16 para Teoria da Computação.

Outra medida estatística se refere ao número de alunos considerando o nível acadêmico de cada um (Figura 2): 12 alunos de graduação, 16 de mestrado e 10 de doutorado.

Com relação ao tipo de participação 28 foram apresentações orais (Full Paper) e 06 foram pôsteres (Poster + Short Paper) (Figura 2).



Houve a participação de cerca de 210 pessoas no evento ao longo de todo o *Workshop*, entre participações de trabalhos e ouvintes. A organização do XII WTD contou com 14 pessoas que atuaram como membros do comitê de organização do evento. Os certificados de participação das apresentações orais e pôsteres foram confeccionados em formato eletrônico e disponibilizados a todos os participantes.

Tabela 4: Programação das apresentações orais do XII WTD realizadas na sala 351 do Instituto de Computação (IC-3.5), dia 02 de Agosto de 2017.

Dia 02 de Agosto de 2017 - Sala 351			
Horário	**Aluno(a) Orientador(a)**	**Trabalhos: apresentações orais**	**Área**
09:00-09:30	Luan Cardoso dos Santos Julio Lopez	Pipeline Oriented Implementation of NORX AEAD for ARM Processors	Teoria da Computação
09:30-10:00	Armando Faz Hernández Julio Lopez	An Introduction to Isogeny-based Cryptography	Teoria da Computação
10:00-10:30	Tiago Reis Julio Lopez	PRESENT runs fast: efficient and secure implementation in software	Teoria da Computação
11:00-11:30	André Carvalho Silva Orlando Lee	Grafos com número máximo um cruzamento	Teoria da Computação
11:30-12:00	Maycon Sambinelli Orlando Lee	Advances in Aharoni-Hartman-Hoffman's Conjecture for Split digraphs	Teoria da Computação
12:00-12:30	Murilo Santos de Lima Orlando Lee	On Generalizations of the Parking Permit Problem and Network Leasing Problems	Teoria da Computação
14:00-14:30	Hugo Kooki K. Rosado Lehilton L. C. Pedrosa	Árvore de Steiner Métrica com Peso nos Vértices	Teoria da Computação
14:30-15:00	Ulysses Rocha Flavio Keidi Miyazawa	Abordagens Heurísticas para o p-Cabo-Trincheira com Localização de Instalações	Teoria da Computação
15:00-15:30	Francisco J. M. da Silva Flavio Keidi Miyazawa	Game-Theoretic Analysis of Transportation Problems	Teoria da Computação
16:30-17:00	Celso Aimbiré W. Santos Christiane N. Campos e Rafael S. C. Schouery	Tight bounds for gap-labellings	Teoria da Computação
17:00-17:30	Natanael Ramos Cid Carvalho de Souza	Uma Matheurística para o Problema do Brigadista em Grafos	Teoria da Computação

Tabela 5: Programação das apresentações orais do XII WTD realizadas na sala 352 do Instituto de Computação (IC-3.5), dia 02 de Agosto de 2017.

Dia 02 de Agosto de 2017 - Sala 352			
Horário	**Aluno(a)** **Orientador(a)**	**Trabalhos: apresentações orais**	**Área**
09:00-09:30	João Paulo da Silva Ricardo Dahab	Criptografia baseada em Isogenias	Teoria da Computação
09:30-10:00	Luiz Claudio Navarro Ricardo Dahab	Using ontologies and machine-learning techniques for malware identification in Android environment	Sistemas de Informação
11:00-11:30	Hayato Fujii Diego de Freitas Aranha	Curve25519 for the Cortex-M4 and beyond	Teoria da Computação
11:30-12:00	Amanda C. D. Resende Diego de Freitas Aranha	Unbalanced Approximate Private Set Intersection	Teoria da Computação
12:00-12:30	Arthur Costa Lopes Diego de Freitas Aranha	Formal verification of constant-time execution	Teoria da Computação
14:00-14:30	Luis Forquesato Juliana Borin	Determinando aprendizado de pensamento computacional por análise de estatísticas de uso	Sistemas de Computação
14:30-15:00	Priscila A. de M. Ioris Rodolfo J. de Azevedo	Adding Custom Instructions into SuperScalar Processor	Sistemas de Computação
16:30-17:00	Cristiano Borges Cardoso Leandro Villas	Uma Solução Híbrida para os Problemas de Localização 3D e Sincronização em RSSFs	Sistemas de Computação
17:00-17:30	Esdras R. do Carmo Paulo Lício de Geus	Malware Behavior Analysis on the Deep Web	Sistemas de Computação

Tabela 6: Programação das apresentações orais do XII WTD realizadas na sala 353 do Instituto de Computação (IC-3.5), dia 02 de Agosto de 2017.

Dia 02 de Agosto de 2017 - Sala 353			
Horário	**Aluno(a) Orientador(a)**	**Trabalhos: apresentações orais**	**Área**
09:30-10:00	Wellington Lucas Moura Eliane Martins	Avaliação de Robustez do Docker Engine	Sistemas de Informação
10:00-10:30	Narcísio José Mula Eliane Martins	Testes de segurança baseado em modelos e sua aplicação ao Moodle Quizz	Sistemas de Informação
11:00-11:30	Luiz Alberto F. Gomes Mario Lúcio Côrtes	Machine Learning Based Prediction of CR Severity Level in FLOSS: Experimental Results	Sistemas de Informação
11:30-12:00	João Luis V. de Oliveira Mario Lúcio Côrtes	Using Machine Learning to promote knowledge Management in Agile Projects	Sistemas de Informação
12:00-12:30	Fernando Vieira da Silva Ariadne M. B. R. Carvalho	Anotação de emoções em tweets de investidores	Sistemas de Informação
14:00-14:30	Jaudete Daltio Claudia M. B. Medeiros	Graph-Kaleidoscope: A Framework to Handle Multiple Perspectives in Graph Databases	Sistemas de Informação
14:30-15:00	Márcio de C. Saraiva Claudia M. B. Medeiros	Finding out topics in educational materials using their components	Sistemas de Informação
15:00-15:30	Lucas Faloni Ferreira Cecília Mary F. Rubira	Prontuário Eletrônico de Paciente apoiado em técnicas de Linhas de Produtos de Software	Sistemas de Informação
16:30-17:00	Fabrício M. Gonçalves Cecília Baranauskas	Sistema para Gerenciamento de Projetos de Pesquisa Sociotécnica	Sistemas de Computação

Tabela 7: Programação das apresentações da de pôsteres do XII WTD realizadas dia 02 de Agsto de 2017 no *Hall* do IC3.5.

Dia 02 de Agosto de 2017 - Hall do IC 3.5			
Aluno(a)	**Orientador(a)**	**Trabalhos: seções de pôsteres**	**Área**
Akari Ishikawa	Eric Rohmer FEEC	Human Machine Interface for Hand Prosthesis Based on Electromyography and Computer Vision	Sistemas de Informação
Vitor Falcão Esdras Rodrigues André Almeida Seiji Hirao	Paulo L. de Geus	TorBot: Protecting the Tor Network against Malicious Traffic	Sistemas de Computação
Luan Egidio Ferreira Diogo Hideki Shiraishi	Edson Borin	Estudo de arquiteturas modulares para smartphones	Sistemas de Computação
Maria Bolina Kersanach	Helio Pedrini	Estratégias In Silico para Bioimpressão de Órgãos	Sistemas de Computação
Leandro A. F. de Magalhães	Diego Aranha	Aprimoramento do método Diceware para geração de senhas seguras	Teoria da Computação
Vinicius Balbino de Souza	Lehilton Chaves	Uma Aproximação para o problema de Alocação de P Terminais	Teoria da Computação

A Figura 2 apresenta a representação visual em formato de *Tag Cloud* das palavras-chave relacionadas aos trabalhos apresentados.

3 Resumos Estendidos

"Na ciência, o crédito vai para o homem que convence o mundo de uma ideia, não para aquele que a teve primeiro".

William Osler

Adding Custom Instructions into SuperScalar Processor

Priscila A. de Moraes Ioris, Rodolfo Azevedo

[1] Instituto de Computação –
Universidade Estadual de Campinas (Unicamp) –
Campinas, SP – Brazil

{priscila.moraes,rodolfo}@ic.unicamp.br

Abstract. *In the last few decades, there has been a transition from a simplescalar processor to superscalar processor. However, such development is not an easy task. To make it easier, Choudhary has developed a tool called FabScalar that aims to facilitate the creation of single ISA multicore processors. This tool allows quickly develop and use of a large range of different processors. In this work, we improved FabScalar to include custom designed instructions.*

Resumo. *Nas últimas décadas houve uma transição entre os processadores escalares para os processadores superescalares. No entanto, o desenvolvimento dos mesmos não é uma tarefa fácil. No sentido de facilitar esse trabalho, Choudhary desenvolveu uma ferramenta chamada FabScalar que visa facilitar criação dos processadores single-ISA multi-núcleo. Essa ferramenta permite que uma vasta gama de processadores diferentes possam ser facilmente desenvolvidos e utilizados. Nesse trabalho nós melhoramos o FabScalar inserindo instruções personalisadas.*

1. Introduction

Superscalar processors can exploit Instruction Level Paralelism (ILP) transparently and efficiently [Patterson and Hennessy 2012]. The main characteristic of superscalar processors is fetching multiple instructions and simultaneously dispatch them to Functional Units (FUs). They must have many similar FUs (e.g.: integer, float-point, load/store) and execute multiple instructions at the same time. Also, those processors can provide out-of-order execution and use branch prediction to increase the number of instructions in the pipeline [Patterson and Hennessy 2012].

With the widespread use of the processors and given the difficulties encountered in their implementation, Choudhary [Choudhary 2012, Choudhary et al. 2012] developed FabScalar. The goal of FabScalar is to automate the creation of different processors that use the same Instruction Set Architecture (ISA). This tool allows computer architects to design different superscalar cores, alleviating the development and verification effort that currently impedes exploring different microarchitecture configurations [Okamoto et al. 2014]. FabScalar offers a core generator based on a global model where developers can choose the appropriate parameters of core design configuration [Choudhary 2012, Choudhary et al. 2012, Choudhary et al. 2011].

Custom instructions can be inserted to processors into execute a particular computation [Singh and Jain 2015]. In this work, we added to FabScalar three new customs instructions: Two for Arithmetic Logic Unit (ALU) and one for Load/Store Unit (LSU).

2. FabScalar

Fabscalar is a tool that creates different types of microarchitecture by automatically composing the synthesizable Register Transfer Level (RTL) designs of diverse cores within a superscalar template that defines the pipeline stages and interfaces among them (Figure 2). Canonical Pipeline Stage Library (CPSL) offers options for each pipeline stage. Each implementation can differ in superscalar width and depth of sub-pipelining. Fabscalar uses the CPSL and the template to generate an RTL of the entire core using the desired configuration. Figure 1 shows the flow to generate one core.

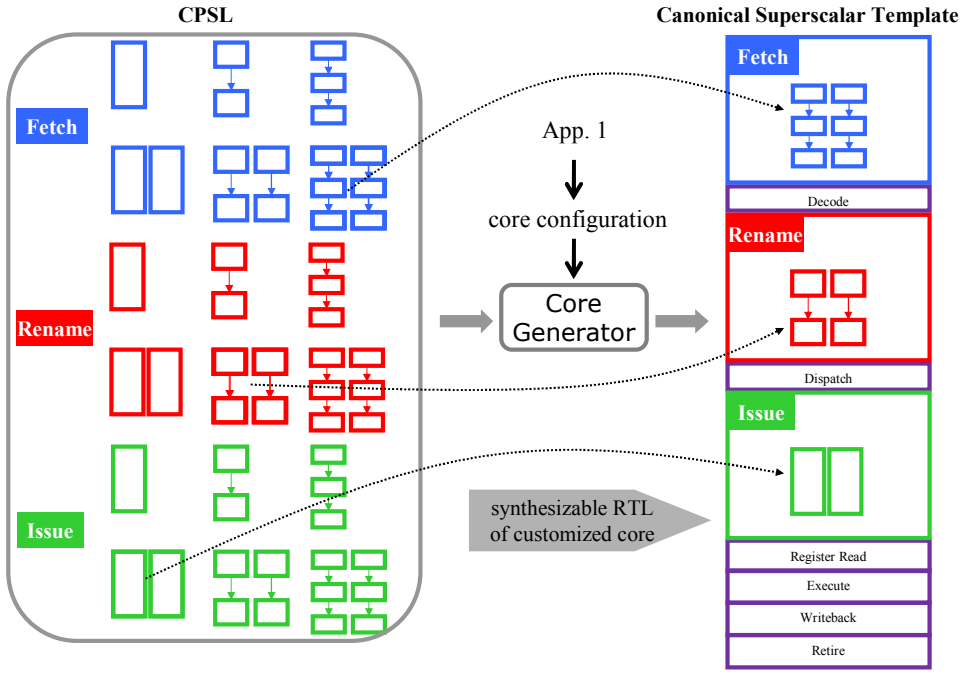


Figure 1. ***Generating a customized core with FabScalar.***

Fabscalar pipeline can be adapted to three major dimensions:

Superescalar Complexity: The superscalar complexity is determinate as a product of its superscalar width and the size of its related ILP- extracting structures (such as the issue queue, physical register file, and predictors). When the superscalar complexity is increased, users can extract more ILP from the program at the cost of increment logic delay through pipeline stages. This affects the overall performance, but the impact can only be analyzed with the next differentiating factor.

Subpipeline: A pipeline stage usually lasts one clock cycle; however it could be subpipelined to achieve higher frequencies;

Stage-specific design choices: There are many choices of microarchitectural implementations such as speculation and recovery alternatives. Each of this possibilities

presents a range of costs and benefits; furthermore, the costs and benefits usually depend on specific instruction-level behaviors in the software.

CPSL offers two microarchitectures schedulers: MIPS R10K and Tomasulo-style. FabScalar have three possibilities of ISA: Portable Instruction Set Architecture (PISA) (derived from MIPS ISA [Price 1995, Burger et al. 1996]), MIPS32R2 and RISC-V [Chowdhury et al. 2015, Chowdhury et al. 2016].

Figure 2 shows the Fabscalar canonical superscalar template. It consists of nine canonical pipeline stages: fetch, decode, rename, dispatch, issue, register read, execute, writeback, and retire. Some of these stages can be configured using pre-defined parameters (as shown in the white boxes).

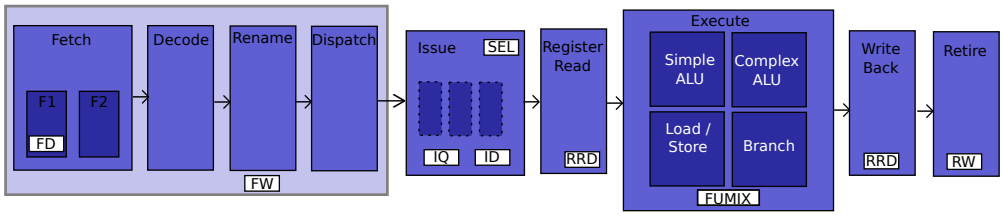


Figure 2. FabScalar pipeline with parameters

Some of the nine pipeline stages presented in Fig 2 can be subdivided: fetch can be divided in two and issue can provide up to three subdivisions. As a result, processor generated with FabScalar can have up to 12 pipeline stages. We added new custom instructions inside was inserted inside the existing FUs, it is important to mention that FUs do not require the same amount of cycles to execute distinct instructions.

3. Insertion of Custom Instruction

It is necessary to have a Hardware Description Language (HDL) implementation to add custom instructions at General Purpose Processors (GPPs). Custom instructions are usually found in Application-Specific Instruction Set Processors (ASIPs) [Singh and Jain 2015]. ASIP relies on programmable architecture that is designed to execute certain tasks optimally. This increased efficiency is associated with: faster performance, reduced production costs, simplified manufacturing process, and less power consumption [Qasim et al. 2012].

Processors have fixed ISA. Our work proposes to insert a custom instruction inside MIPS processor to fill the gap of this particular ISA. In the future, we will use FabScalar to generate new processors with different ISAs.

We started mapping the available opcodes of MIPS32R2 and select some unused ones. Each custom instruction must have a unique opcode. We changed FabScalar decoder to recognize the new instructions. Inside the decoder, we selected the source and destination registers, define de control signals to indicate the desired FU and remaining modules of the pipeline. We also mapped the opcode to the internal one to be used inside FabScalar (this FabScalar-MIPS32R2 maps all 6-bits MIPS opcodes to a new 12-bits internal opcode).

We used two versions of FabScalar to insert the new instructions: SimpleScalar and MIPS32R2. In the future, we will only work with the MIPS32R2 version. We made

an isolated Verilog module (Fig. 3) for each custom instruction to increase flexibility, scalability, and ease-of-use. The modules were inserted inside the proper FU as showed in Fig 4.

Figure 3. Custom instruction module.

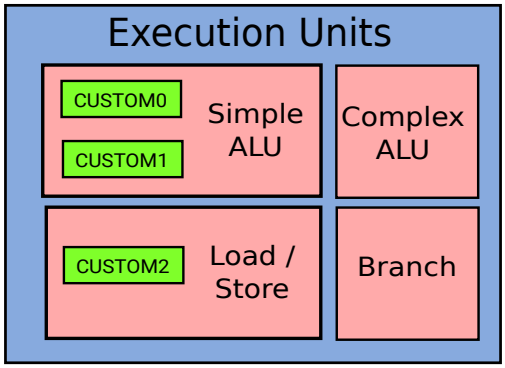


Figure 4. Custom instructions instantiate inside FUs

We designed three different kinds of custom instruction:

1. Two register inputs and one output - Simple ALU instruction;
2. One register, one immediate input, and one output - Simple ALU instruction;
3. One register, one immediate input, and one output - LSU instruction;

In this preliminary work we only reimplemented three existing instructions (add, addi and load) and assigned new opcodes to them. To verify the new instructions, we compare the behavior of the existing instructions to the newest ones by executing some of SPEC2000 benchmarks with the new instructions and compare the results.

We also changed the compiler back-end to support the custom instruction. This new instruction is encoded using inline assembly, Listing 1 shows a simple example of custom0 usage.

```
1 int main() {
2     unsigned int x;
3     asm ("li $t2, 20");
4     asm ("li $t3, 4");
5     asm ("custom0 $t5, $t2, $t3");
6     asm ("move %0, $t5" : "=r"(x));
7     printf ("R = %d\n", x);
8     exit(0);}
```

Listing 1. Simple C code using custom0 instruction.

In the future, we want to implement more complex instructions, and our target will be instructions dedicated to cryptography applications.

4. Conclusion

Custom instructions can be very useful to accelerate some algorithms. However, it is necessary to be careful with their implementation to not interfere with the frequency of generated processor. If the developer creates an instruction that has a delay higher than the clock processor, the final hardware will be slower than the original one. In this case, there must be an evaluation of the performance gain to analysis if the code execution using the processor with the custom instruction is faster than the execution without it.

References

Burger, D., Austin, T. M., and Bennett, S. (1996). Evaluating future microprocessors: the simplescalar tool set. Technical report, University of Wisconsin-Madison and Intel.

Choudhary, N., Wadhavkar, S., Shah, T., Mayukh, H., Gandhi, J., Dwiel, B., Navada, S., Najaf-abadi, H., and Rotenberg, E. (2011). Fabscalar: Composing synthesizable rtl designs of arbitrary cores within a canonical superscalar template. In *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pages 11–22.

Choudhary, N., Wadhavkar, S., Shah, T., Mayukh, H., Gandhi, J., Dwiel, B., Navada, S., Najaf-abadi, H., and Rotenberg, E. (2012). Fabscalar: Automating superscalar core design. *Micro, IEEE*, 32(3):48–59.

Choudhary, N. K. (2012). *FabScalar: Automating the Design of Superscalar Processors*. PhD thesis, Faculty of North Carolina State University, Raleigh, North Carolina.

Chowdhury, R. B. R., Kannepalli, A. K., Ku, S., and Rotenberg, E. (2016). Anycore: A synthesizable rtl model for exploring and fabricating adaptive superscalar cores. In *ISPASS*.

Chowdhury, R. B. R., Kannepalli, A. K., and Rotenberg, E. (2015). Fabscalar risc-v. In *2nd RISC-V Workshop Proceedings*.

Okamoto, T., Nakabayashi, T., Sasaki, T., and Kondo, T. (2014). Detail design and evaluation of fab cache. In *Computing and Networking (CANDAR), 2014 Second International Symposium on*, pages 591–595.

Patterson, D. A. and Hennessy, J. L. (2012). *Computer Architecture: A Quantitative Approach*. The Morgan Kaufmann Series in Computer Architecture and Design. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA Â©2011.

Price, C. (1995). Mips iv instruction set. Technical report, MIPS Technologies, Inc. Revision 3.2.

Qasim, Y., andPradyumna Janga, Kumar, S., and Alesaimi, H. (2012). Application specif processors. Cs 570 - project final report, Oregon State University.

Singh, M. P. and Jain, M. K. (2015). Isa customization for application specific instruction set processors. In *Pervasive Computing (ICPC), 2015 International Conference on*, pages 1–4.

Criptografia Baseada em Isogenias

João Paulo da Silva[1], **Ricardo Dahab**[1]

[1]Instituto de Computação – Universidade de Campinas (Unicamp)
1251 – 13083-852 – Campinas – SP – Brasil

jsilva@lasca.ic.unicamp.br,rdahab@ic.unicamp.br

Abstract. *Modern cryptography has as its pillar the difficulty in solving certain problems arising from several areas of knowledge. With the possible emergence of large-scale quantum computers several of these problems would be efficiently solved. In order to circumvent this fact, some problems that would remain difficult arise as successors and with them the area of study called Post-Quantum Cryptography. Cryptosystems based on the problem of calculating isogenies between supersingular elliptic curves were recently proposed as strong candidates. We introduce the concepts and algorithms in relation to them and point out our future research perspectives on such subject.*

Resumo. *A criptografia moderna tem como pilar a dificuldade em se resolver certos problemas advindos de várias áreas do conhecimento. Com o possível surgimento de computadores quânticos de larga escala vários desses problemas seriam eficientemente resolvidos. De forma a contornar tal fato, alguns problemas que permaneceriam difíceis surgem como sucessores e com eles a área de estudo denominada Criptografia Pós-Quântica. Criptossistemas baseados no problema de se calcular isogenias entre curvas elípticas supersingulares foram propostos recentemente como fortes candidatos. Introduzimos os conceitos e algoritmos em relação a eles e apontamos nossas perspectivas futuras de pesquisa em tal tema.*

1. Contexto

Desde o advento do algoritmo de Shor[Boneh and Lipton 1995] e a possibilidade de um computador quântico em larga escala a comunidade criptográfica se empenha de forma a se precaver. Tal algoritmo torna capaz de se determinar a ordem e estrutura de grupos abelianos finitos, assim como o cálculo do logaritmo discreto em grupos cíclicos, em tempo polinomial. Devido ao fato dos sistemas mais comumente empregados atualmente, como o RSA, serem vulneráveis ao algoritmo de Shor torna-se necessário a busca por novas opções. A partir dessa necessidade surge a área da criptografia chamada Criptografia Pós-Quântica. Vários candidatos foram apresentados desde então para suprir a demanda por criptossistemas resistentes a computadores quânticos: criptossistemas baseados em reticulados, criptossistemas baseados em códigos, criptossistemas baseados em hash. Mais recentemente foi proposta a Criptografia baseada em Isogenias. Em relação aos demais candidatos os esquemas conseguidos com a criptografia baseada em isogenias possuem chaves notoriamente menores.

1.1. Organização

Este trabalho está dividido da seguinte forma: A Seção 2 apresenta os conceitos básicos de curvas elípticas; a Seção 3 apresenta alguns conceitos relativos à isogenias; a Seção 4

aborda os algoritmos de troca de chaves e encriptação baseados em isogenias de curvas supersingulaares; na Seção 6, listamos os objetivos a serem alcançados durante o tempo de realização desta pesquisa; enfim, na Seção 7 discutimos os resultados esperados e seus impactos.

2. Curvas Elípticas

2.1. Definição

Uma curva elíptica E sobre um corpo K é uma curva algébrica de *genus* 1 a qual pode ser representada na forma de Weierstrass

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

com $a_1, a_2, ..., a_6 \in K$. Se tomarmos K de forma que a característica do corpo K seja diferente de 2 ou 3, com algumas mudanças de variáveis obtemos

$$y^2 = x^3 + Ax + B$$

com $A, B \in K$. Para que a equação acima, chamada Equação de Weierstrass Reduzida, não possua raízes múltiplas assumimos $4A^3 + 27B^2 \neq 0$. Denominamos por $E(K)$ o conjunto de pontos da curva elíptica E sobre o corpo K. A esse conjunto é adicionado um ponto auxiliar, induzido geometricamente, no infinito denotado por $\infty = (\infty, \infty)$.

Quando temos $K = \mathbb{F}_q$, onde $q = p^m, m \in \mathbb{N}$, um corpo finito, o Teorema de Hasse nos diz que $\#E(\mathbb{F}_q) = q + 1 - t$, onde $|t| \leq 2\sqrt{q}$. Uma curva elíptica é dita ser *supersingular* se, e somente se, $t \equiv 0 \; mod \; p$, caso contrário ela é dita *ordinária*.

2.2. j-Invariante de uma Curva Elíptica

Seja E uma curva elíptica definida pela equação $y^2 = x^3 + Ax + B$, com $A, B \in K$ e característica de K diferente de 2 ou 3. Definimos o *j-invariante* de E como

$$j = j(E) = 1728\frac{4A^3}{4A^3 + 27B^2}.$$

Teorema 1 *Sejam $y_1^2 = x_1^3 + A_1x_1 + B_1$ e $y_2^2 = x_2^3 + A_2x_2 + B_2$ curvas elípticas definidas sobre um corpo K com j-invariantes j_1 e j_2, respectivamente. Se $j_1 = j_2$, então existe $\mu \in \overline{K}^\times$ tal que*

$$A_2 = \mu^4 A_1, \quad B_2 = \mu^6 B_1.$$

A transformação

$$x_2 = \mu^2 x_1, \quad y_2 = \mu^3 y_1$$

leva uma curva na outra.

2.3. Pontos de Torção

Os pontos de torção de uma curva elíptica são pontos cuja ordem é finita. Abaixo segue uma definição dos grupos dos pontos de torção de ordem n.

Definição 2 *Seja E uma curva elíptica definida sobre um corpo K. Seja n um inteiro positivo. Definimos o conjunto formado pelos pontos de torção de ordem n como*

$$E[n] = \{P \in E(\overline{K}) \mid nP = \infty\}.$$

O teorema abaixo caracteriza os conjuntos dos pontos de torção através de isomorfismos de grupos.

Teorema 2 *Seja E uma curva elíptica definida sobre um corpo K. Seja n um inteiro positivo. Se a característica de K não divide n, ou é 0, então*

$$E[n] \simeq \mathbb{Z}_n \bigoplus \mathbb{Z}_n.$$

Se a característica de K é $p \geq 0$ e $p|n$, escreva $n = p^r n'$ com $p \nmid n'$. Então

$$E[n] \simeq \mathbb{Z}_{n'} \bigoplus \mathbb{Z}_{n'}$$

ou

$$E[n] \simeq \mathbb{Z}_n \bigoplus \mathbb{Z}_{n'}.$$

3. Isogenias

3.1. Definição

Fixados um primo p e uma potência $q = p^k$, sejam E_1 e E_2 curvas elípticas sobre $K = \mathbb{F}_q$. Uma isogenia $\phi : E_1 \to E_2$ é um morfismo algébrico não constante

$$\phi(x, y) = \left(\frac{f_1(x, y)}{g_1(x, y)}, \frac{f_2(x, y)}{g_2(x, y)} \right)$$

com $\phi(\infty) = \infty$ e $f_i, g_i, i \in \{1, 2\}$, polinômios.

Mais especificamente, ϕ é um homomorfismo de grupos entre $E_1(\mathbb{F}_q)$ em $E_2(\mathbb{F}_q)$, ou seja, dados $P, Q \in E_1(\mathbb{F}_q)$ vale

$$\phi(P + Q) = \phi(P) + \phi(Q)$$

sendo a operação $+$ a soma de pontos elípticos.

O teorema abaixo nos fornece uma condição necessária e suficiente para termos duas curvas isógenas.

Teorema 3 *(Tate) Sejam E_1 e E_2 curvas elípticas definidas sobre $\mathbb{F}_q$, $q = p^k$. E_1 é isógena a E_2 se, e somente se, $\#E_1 = \#E_2$.*

3.2. Fórmula de Vélu

Afim de computarmos, de forma explícita, os polinômios que compõem a isogenia da forma definida anteriormente fazemos uso de uma fórmula devida a Vélu[Vélu 1971].

Teorema 4 *(Fórmula de Vélu) Considere o corpo base K tal que a característica de K seja diferente de 2 ou 3. Seja $E : y^2 = x^3 + Ax + B$ uma curva elíptica na forma de Weierstrass simplificada. Seja G um subgrupo de E de ordem l, l primo. Seja S o conjunto dos representantes de $G/\sim$, onde é tal que $P \sim Q \iff P = \pm Q$. Então, existe uma isogenia $\phi : E \to E'$ onde $ker\phi = G$, dada por*

$$\phi_x(x,y) = x + \sum_{Q \in S} \left[\frac{t_Q}{x - x_Q} + \frac{\mu_Q}{(x - x_Q)^2} \right]$$

$$\phi_y(x,y) = y - \sum_{Q \in S} \left[\mu_Q \frac{2y}{(x - x_Q)^3} + t_Q \frac{y - y_Q}{(x - x_Q)^2} - \frac{g_Q^x g_Q^y}{(x - x_Q)^2} \right]$$

onde

$$Q = (x_Q, y_Q), \quad \mu_Q = (g_Q^y)^2,$$
$$t_Q = \begin{cases} g_Q^x & \text{, se } Q = -Q \\ -2g_Q^x & \text{, se } Q \neq -Q \end{cases}$$
$$g_Q^x = 3x_Q^2 + A, \quad g_Q^y = -2y_Q.$$

Observa-se que o cômputo é feito através de somas em pontos do subgrupo G. Sua complexidade é dada por $\mathcal{O}(|G|)$.

4. Algoritmos

Fixaremos $\mathbb{F}_q = \mathbb{F}_{p^2}$, p primo da forma $l_A^{e_A} l_B^{e_B} f \pm 1$, l_A, l_B primos pequenos e f um cofator, de forma que p seja primo. Conseguimos, assim, uma curva E sobre $\mathbb{F}_{p^2}$ tal que $\#E(\mathbb{F}_{p^2}) = (l_A^{e_A} l_B^{e_B} f)^2$.

Apresentaremos a seguir um algoritmo para troca de chaves semelhante ao Diffie-Hellman para curvas elípticas supersingulares (SIDH) e um algoritmo de encriptação que pode ser facilmente derivado do SIDH[Feo et al. 2011]. A troca de informação é feita entre dois indivíduos hipotéticos denominados Alice e Bob.

4.1. SIDH

- Parâmetros Públicos: Fixamos uma curva elíptica supersingular E_0 definida sobre $\mathbb{F}_{p^2}$ e bases $\{P_A, Q_A\}$ e $\{P_B, Q_B\}$ para os grupos de pontos de torção $E_0[l_A^{e_A}]$ e $E_0[l_B^{e_B}]$, respectivamente.
- Alice escolhe $m_A, n_A \in_R \mathbb{Z}_{l_A^{e_A}}$, tal que não sejam os dois divisíveis por l_A e calcula $\phi_A : E_0 \to E_A$ (fórmula de Vélu) com *kernel* $K_A := \langle [m_A]P_A + [n_A]Q_A \rangle$. Alice computa também o conjunto $\{\phi_A(P_B), \phi_A(Q_B)\} \subset E_A$. Alice então envia para Bob a tupla
$$(E_A, \phi_A(P_B), \phi_A(Q_B)).$$
- Bob faz cálculos análogos aos de Alice e envia para ela a tupla
$$(E_B, \phi_B(P_A), \phi_B(Q_A)).$$
- Ao receber os dados de Bob, Alice computa $\phi'_A : E_B \to E_{AB}$ com o *kernel* sendo o subgrupo $\langle [m_A]\phi_B(P_A) + [n_A]\phi_B(Q_A) \rangle$. Bob procede de forma análoga.
- A chave secreta compartilhada será dada pelo j-invariante da curva comum a eles (isomorfas)

$$E_{AB} = \phi'_B(\phi_A(E_0)) = \phi'_A(\phi_B(E_0)) = E_0 / \langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B \rangle$$

4.2. Encriptação

A partir do algoritmo de troca de chaves visto anteriormente se deduz um algoritmo para encriptação.

- Toma-se $p = l_A^{e_A} l_B^{e_B} f \pm 1$, E_0, $\{P_A, Q_A\}$ e $\{P_B, Q_B\}$ como no algoritmo anterior. Seja $\mathcal{H} = \{H_k : k \in K\}$ uma família de funções de *hash* indexadas por um conjunto finito K, sendo H_K uma função de $\mathbb{F}_{p^2}$ no espaço de mensagens $\{0,1\}^w$.
- **Geração de Chaves**: Escolha $m_A, n_A \in_R \mathbb{Z}_{l_A^{e_A}}$, não ambos divisíveis por l_A. Compute $(E_A, \phi_A(P_B), \phi_A(Q_B))$ e escolha $k \in_R K$.
  - **Chave Pública**: $(E_A, \phi_A(P_B), \phi_A(Q_B), k)$.
  - **Chave Privada**: (m_A, n_A, k).
- **Encriptação**: Dadas a chave pública e a mensagem $m \in \{0,1\}^w$, escolha dois elementos $m_B, n_B \in_R \mathbb{Z}_{l_B^{e_B}}$, não ambos divisíveis por l_B, e calcule

$$h = H_k(j(E_{AB})),$$

$$c = h \bigoplus m.$$

  O texto cifrado é a tupla $(E_B, \phi_B(P_A), \phi_B(Q_A), c)$.
- **Decriptação**: Dado o texto cifrado e a chave privada (m_A, n_A, k), calcule o j-invariante $j(E_AB)$ e faça

$$h = H_k(j(E_{AB})),$$

$$m = h \bigoplus c.$$

  O texto claro é m.

5. Problemas Difíceis em Isogenia

Abaixo listaremos alguns problemas definidos em [Feo et al. 2011] para isogenias entre curvas elípticas supersingulares, os quais se supõe serem difíceis. A dificuldade suposta é em relação ao fato de que para todo algoritmo polinomial que resolva os problemas, a vantagem de tal algoritmo é uma função negligenciável no parâmetro de segurança $\log p$[Feo et al. 2011].

- **Decisional Supersingular Isogeny (DSSI) Problem**: Seja E_0. Dada E_A outra curva elíptica supersingular definida sobre $\mathbb{F}_{p^2}$, decida se E_A é $l_A^{e_A}$-isógena a E_0.
- **Computational Supersingular Isogeny (CSSI) Problem**: Seja $\phi_A : E_0 \to E_A$ uma isogenia cujo *kernel* é $\langle [m_A]P_A + [n_A]Q_A \rangle$, com $m_A, n_A \in_R \mathbb{Z}_{l_A^{e_A}}$ e não ambos divisíveis por l_A. Dada E_A e os valores $\phi_A(P_B), \phi_A(Q_B)$, encontre um gerador de $\langle [m_A]P_A + [n_A]Q_A \rangle$.
- **Supersingular Computational Diffie-Hellman (SSCDH) Problem**: Seja $\phi_A : E_0 \to E_A$ uma isogenia cujo *kernel* é $\langle [m_A]P_A + [n_A]Q_A \rangle$, e seja $\phi_B : E_0 \to E_B$ uma isogenia cujo *kernel* é $\langle [m_B]P_B + [n_B]Q_B \rangle$, com $m_A, n_A \in_R \mathbb{Z}_{l_A^{e_A}}$ (respectivamente $m_B, n_B \in_R \mathbb{Z}_{l_B^{e_B}}$) e não ambos divisíveis por l_A (respectivamente l_B). Dadas as curvas E_A e E_B e os pontos $\phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A)$, encontre o j-invariante de $E_0/\langle [m_A]P_A + [n_A]Q_A, [m_B]P_B + [n_B]Q_B \rangle$

6. Objetivos

Os objetivos gerais deste trabalho resumem-se a:

- Realização de um estudo dos conceitos e algoritmos empregados na construção de primitivas criptográficas baseadas em isogenias;
- Propor um novo esquema de geração de chave compartilhada que não necessite da divulgação de pontos auxiliares e assim possivelmente comprometer a segurança do esquema;
- Estudo e implementação de fórmulas análogas à de Vélu para o cálculo e avaliação de isogenias para curvas nos modelos de Edwards e Huff.

7. Resultados Esperados

Através da realização deste trabalho buscamos estender o estado da arte no âmbito da criptografia baseada em isogenias. A apresentação de um algoritmo de troca de chaves que não revelasse informação adicional tornaria mais seguro o protocolo contra ataques como o apresentado em [Petit 2017].

Os autores em [Moody and Shumow 2011] propõe novas fórmulas para o cálculo e avaliação de isogenias. Eles afirmam serem elas mais interessantes pelo fato de serem derivadas de fórmulas de adição e duplicação de pontos elípticos mais amigáveis e eficientes que a tradicionalmente utilizada para curvas no modelo de Weierstrass. Obter implementações eficientes de tais fórmulas possibilitaria tornar as primitivas criptográficas baseadas em isogenias ainda mais fortes concorrentes para um possível padrão em Criptografia Pós-Quântica.

Referências

Boneh, D. and Lipton, R. J. (1995). Quantum cryptanalysis of hidden linear functions (extended abstract). In *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, volume 963 of *Lecture Notes in Computer Science*, pages 424–437. Springer.

Feo, L. D., Jao, D., and Plût, J. (2011). Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Cryptology ePrint Archive, Report 2011/506.

Moody, D. and Shumow, D. (2011). Analogues of velu's formulas for isogenies on alternate models of elliptic curves. Cryptology ePrint Archive, Report 2011/430.

Petit, C. (2017). Faster algorithms for isogeny problems using torsion point images. Cryptology ePrint Archive, Report 2017/571.

Vélu, J. (1971). Isogénies entre courbes elliptiques. *C.R. Acad. Sc. Paris*, Série A(273):238–241.

Formal verification of constant-time execution

Arthur Costa Lopes[1], **Diego de Freitas Aranha**[1]

[1]Institute of Computing – University of Campinas (UNICAMP)

Abstract. *Developing secure implementations of cryptography, in particular those protected against side-channel attacks is a challenging problem. In the case of timing attacks, the task can be facilitated by employing verification tools to check constant-time behavior. Given this concern, this work explores different verification tools and their distinct forms of analysis with application to cryptographic libraries. A simple syntax for describing implementations was used and a benchmarking database was constructed to validate such tools, indicating that the combination of static and dynamic is required to fully verify the timing behavior of a cryptographic implementation.*

Resumo. *Desenvolver implementações seguras de criptografia, em particular aquelas protegias contra ataques de canal lateral é um problema desafiador. No caso de ataques de temporização, a tarefa pode ser facilitada ao se empregar ferramentas de verificação para determinar execução em tempo constante. Dado esse problema, este trabalho expora diferentes ferramentas de verificação e suas formas distintas de análise, com aplicação em bibliotecas criptográficas. Uma sintaxe simples para descrever as implementações foi utilizada e uma base de implementações foi construída para comparar a validar tais ferramentas, indicando que uma combinação de análise dinâmica e estática é requisito para verificar completamente o comportamento de uma implementação do ponto de vista de tempo de execução.*

1. Introduction

In the past years many timing attacks against cryptographic implementations came up in the literature, compromising entire cryptosystems and exposing sensitive data by using timing characteristics. Those attacks consist in the measure of the executing/response time of an insecure system, which will be different for distinct inputs, since some part of the code will rely on some properties of the input, such as length, partial correctness and some others. Such implementations are called *time variant*, since the running time may vary. On the other hand, if a system is secure against timing attacks, it is called *constant time* because it produces the answer with the same delay independent of the inputs.

There are attacks against implementations of AES that were able to recover the full key almost in real time just by observing encryption of a few kilobytes of data [Gullasch et al. 2011], or by analyzing network response time [Bernstein 2005]. Some others were able to recover Diffie-Hellman exponents and even factor RSA keys by measuring precisely the time required to perform some operations like encryption and multiplications [Kocher 1996], also using the Chinese Remainder Theorem (CRT) and Montgomery reduction together with a timing attack to factor RSA[Schindler 2000].

Timing attacks can be extremely complex as shown above using several techniques and observing network traffic or power consumption, but it can be very simple. For example, suppose an algorithm that checks if a given string matches a certain secret password.

It can be easily implemented by comparing each character from both phrases and, if it founds a difference, it stops the application immediately, or if there are no differences it outputs that the string is correct. Now, given this specific implementation, consider an attack in which the opponent sends a random word, but with the first letter being 'a' and measures the response time. If the first letter of the password is in fact 'a', then the execution time would be greater than the case in which the first letter was wrong, since the code will try to match second letter. Now the attacker has a simple way to check each letter separately, reducing the complexity of the brute force attack from exponential to linear.

2. Related work

Given the power of timing attacks and the volume of articles in the literature, rises the importance of the study of analysis tools to prevent such attacks. Many approaches tried to prevent such vulnerabilities by creating tools to be used by the programmer to improve the code for critical applications. The main types of analysis are the Dynamic and Static analysis, both presented in the sections below.

2.1. Dynamic Analysis

One method to detect if an implementation is vulnerable is to analyze its behavior while executing given some input. This kind of methods are called Dynamic since they rely on an experiment using the compiled version of the code, and not just the source code itself. A recent tool called dudect [Reparaz et al. 2017] was created using the concept. It tests several inputs, trying to find a different behaviour that yields a variant running time. To detect such behaviour it uses some statistical tests, such as the Welch's t-test for detecting variations in the samples collected. An advange of dynamic analysis is detecting timing variances introduced by the instructions, such as early-abort multipliers available in the ARM architecture.
The main limitation of these tools is that they can't check if a given implementation is in fact constant-time, since in this case every sample would have the same value. Furthermore, unlikely inputs carefully crafted by an attacker to produce variable time behavior may not be explored.

2.2. Static Analysis

Another type of analysis, known as static analysis, is widely applied to find vulnerabilities involving sensitive variables used in a cryptosystem. This kind of analysis receives the name static because it works based on an intermediate representation of the code, instead of the execution of a compiled binary.
The static analysis tools can detect several types of problems within an implementation without compiling and executing the software itself. To perform this kind of analysis the user simply scan the source code, sometimes with some special commands. The user will usually received several warnings describing how the secret information flows within the application. Some recent tools like FlowTracker[Rodrigues et al. 2016] and ct-verif[Almeida et al. 2016] uses techniques for analyzing intermediate representation like Boogie or LLVM IR to detect branch instructions using secret information as one of the parameters, therefore being vulnerable since the running time will depend on those parameters.

2.2.1. FlowTracker

FlowTracker [Rodrigues et al. 2016] is a static tool develop to detect and analyze the implicit flow of information within an implementation. Such tool can be used to keep track of sensitive input, such as cryptography keys, and determine if its value may have some influence in the running time, for example by being the parameter for a branch instruction.

3. Representation

Another contribution of this project was to create a standard representation for the usage of the analysis tools. Since each tool has some specific characteristics, the input XML contains a compilation of the main properties that the user must give in order to perform the analysis, for example which parameter is sensitive and the name of the function that is being analyzed. With this general file the user can easily analyze the code being write by various tools, expanding the verification and therefore being able to find a larger number of vulnerabilities.

The XML created to describe an implementations consists of a man tag representing each function to be tested, including the name and all parameters, separated in two sections: public and secret. The user can use comments to specify the meaning of each parameter.

4. Results

The FlowTracker tool was used to analyze several implementations from some cryptography libraries such as BearSSL[1] and NaCl[2]. We checked several implementations from those libraries, including symmetric cryptography, hash functions and MACs, trying to conclude if they are in fact constant-time and secure against timing attacks.

At first, the dudect tool was used, which as expected output "problably constant-time implementation", since every sample taken had the same mean, producing an inconclusive result. After using a dynamic analysis tool we changed to the static one (FlowTracker). Considering how it works, the tool provided the full graph for the implementation and that it does not have any vulnerabilities, confirming that the code is in fact constant-time.

After testing all the constant-time codes we started analyzing the output from the tools for variant-time implementations. As expected, both tools found vulnerabilities for variable-time implementations. FlowTracker was able to show the subgraph indicating where the secret information is being used, along with branch instructions. The tool dudect was able to find samples with different means, concluding that the code does not run in constant time regarding the input.

In total the benchmark database has about 70 implementations of several cryptographic algorithms. From all those implementations there are about 20 that we verified using the FlowTracker that are indeed constant-time since no vulnerability was found. Regarding the rest, FlowTracker was able to find dependencies between secret input specified in the representation file and some branch instructions in the code.

5. Conclusions and future work

Given the amount of code analyzed from the libraries, those tools can indeed be used by a programmer to improve the code and mitigate timing attacks against it. The usage of

[1]BearSSL - A smaller SSL/TLS library - https://www.bearssl.org

[2]NaCl: Networking and Cryptography library - https://nacl.cr.yp.to

those tools is a practice that can be easily applied by developers to help them create more secure implementations, avoiding the leakage of sensitive and crucial data. We observed that a combination of static and dynamic analysis must be used to fully characterize the timing behavior of a cryptographic implementation.

We plan to extend the benchmark database to include more libraries and examples to be used by several tools, analyzing their performance, effectiveness and the correctness.
Also, we will study how those analysis tools could be integrated to commonly used development tools, in order to facilitate the usage by users that are not aware with side-channels attacks against insecure implementations.

6. Acknowledgements

We thank Intel and FAPESP (São Paulo Research Foundation) for financial support though process 14/50704-7.

References

Almeida, J. B., Barbosa, M., Barthe, G., Dupressoir, F., and Emmi, M. (2016). Verifying constant-time implementations. In Holz, T. and Savage, S., editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 53–70. USENIX Association.

Bernstein, D. J. (2005). Cache-timing attacks on aes.

Gullasch, D., Bangerter, E., and Krenn, S. (2011). Cache games - bringing access-based cache attacks on AES to practice. In *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*, pages 490–505. IEEE Computer Society.

Kocher, P. C. (1996). Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Koblitz, N., editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer.

Reparaz, O., Balasch, J., and Verbauwhede, I. (2017). Dude, is my code constant time? In Atienza, D. and Natale, G. D., editors, *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, March 27-31, 2017*, pages 1697–1702. IEEE.

Rodrigues, B., Pereira, F. M. Q., and Aranha, D. F. (2016). Sparse representation of implicit flows with applications to side-channel detection. In Zaks, A. and Hermenegildo, M. V., editors, *Proceedings of the 25th International Conference on Compiler Construction, CC 2016, Barcelona, Spain, March 12-18, 2016*, pages 110–120. ACM.

Schindler, W. (2000). A timing attack against RSA with the chinese remainder theorem. In Koç, Ç. K. and Paar, C., editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 109–124. Springer.

Graph-Kaleidoscope: A Framework to Handle Multiple Perspectives in Graph Databases

Jaudete Daltio[1], **Claudia M. Bauzer Medeiros**[1]

[1]Institute of Computing - UNICAMP - 13083-820 - Campinas/SP, Brazil

{jaudete,cmbm}@ic.unicamp.br

Abstract. *Scientific research has become data-intensive and data-dependent, with the collaboration of multidisciplinary teams. Advances in information technology are continuously being proposed to support researchers in dealing with the underlying large and heterogeneous datasets, such as NoSQL and graph databases. One important requirement not solved yet by these technologies is to support multiple perspectives. This research presents Graph-Kaleidoscope, a framework that allows users to build multiple perspectives over graph databases. Perspectives are defined through an adaptation of the concept of views in relational databases.*

Resumo. *A pesquisa científica tornou-se intensamente usuária de dados e fortemente dependente de dados, sendo cada vez mais presente a colaboração de equipes multidisciplinares. Novas tecnologias são continuamente propostas para apoiar pesquisadores no gerenciamento e na análise de grandes volumes de dados heterogêneos, como bancos de dados NoSQL e bancos de grafos. Uma importante funcionalidade ainda não explorada por essas tecnologias é a criação de múltiplas perspectivas sobre dados. Esta pesquisa apresenta Graph-Kaleidoscope, um framework que permite a criação de múltiplas perspectivas sobre bancos de dados grafos. As perspectivas são definidas através de uma adaptação do conceito de visões de bancos de dados relacionais.*

1. Introduction and Motivation

Increasingly, the world of science is being changed. Data is being produced and collected at an unprecedented scale and outpaces the speed with which it can be analyzed and understood. *Data-intensive science* emerged as a new paradigm for scientific exploration [Hey et al. 2009]. Computer science has become a key element in scientific research in many areas, such as bioinformatics, social network sciences and health.

NoSQL and graph databases are partial solutions to data management challenges for this new scenario, implementing different data models to solve specific data management issues on large and heterogeneous datasets. One important requirement of data-intensive science not solved yet by these technologies is to support multiple perspectives on large and complex datasets. Many research scenarios require dealing with a subset of data of interest, under multiple aggregation/generalization levels, for given perspectives and a specific vocabulary [Santanche et al. 2014].

Motivated by this scenario, the goal of our research is to allow users to build multiple perspectives over graph databases. Perspectives are defined through an adaptation of the concept of views in relational databases – i.e., each perspective is handled as a view in

the graph database. As such, our framework Graph-Kaleidoscope extends the concept of view from relational databases to graph databases – for which, so far, little has been done in terms of views.

This paper presents two main contributions of our research. The first is the formalization of the operators for graph data that underpins our framework. As a result, we define PGDM – a property graph data model – together with its operators. Our second contribution lies in the definition of the framework itself, based on the use of PGDM, and for which we have developed a first prototype.

2. Theoretical Foundations

2.1. The Graph Data Management Paradigm

The graph data management paradigm is defined by the use of graphs as data models and the use of graph-based operators to express data manipulation [Angles and Gutierrez 2008]. The graph data model is relationship driven, as opposed to the relational data model that requires the use of foreign keys and joins to infer connections between data items. Graph databases are usually adopted to represent data sets where relations among data and the data itself are at the same importance level.

In this model, queries are performed through graph traversals, graph pattern matching or graph algorithms. The formal foundation of graph data models is based on variations on the mathematical definition of a graph. On top of the basic layer, several graph data structures were proposed by the database community, attempting to improve expressiveness, representing data in a better (and less ambiguous) way. Each graph data structure has its own data manipulation commands, which can be only invoked at the application level, such as via an API, or at a user-friendly level, such as via a query language.

While graph systems are being increasingly adopted, graph database systems are at the same overall level of maturity as object-oriented database systems were in the early to mid-90's. To that effect, we explicitly paraphrased two sentences of the classical "Object-Oriented Manifesto" [Atkinson et al. 1992], concerning the state of the art of OO databases in 1989, when that paper first appeared, and which we repeat here. *"Three points characterize the field at this stage: (i) the lack of a common data model, (ii) the lack of formal foundations and (iii) strong experimental activity. Whereas Codd's original paper [Codd 1980] gave a clear specification of a relational database system (data model and query language), no such specification exists for object-oriented database systems"* (Manifesto page 2). If we now replace the term "object-oriented" by "graph", the entire sentence holds.

2.2. Database Views

As emphasized in [Santanche et al. 2014], though a database view was originally defined to be the result of a query, its definition has evolved with time to designate a portion of the data that is of interest to a specific group of users. Under this premise, views are no longer "mere" queries, but an adequate means to support multiple, interdisciplinary perspectives of a given database. Thus, the concept of views adopted in our research extends views in relational databases. For us, a view can be used to achieve different purposes: (i) restriction: to simplify the use of data – to hide unnecessary details or to provide data protection, blocking the access to sensitive data; (ii) scale: to create virtual units derived

from aggregations of database objects; or (iii) restructure: to create virtual units derived from rearrangements of database objects.

The notion of view, as a particular perspective built on the underlying data, is independent from the database model. View mechanisms, however, are strongly dependent on the underlying model. Views mechanisms are consolidated in relational systems. The same does not occur in graph databases, mostly due to the absence of a consensual model.

3. Graph-Kaleidoscope Framework

The first challenge addressed by our research is to formalize the graph data model that underpins our framework in terms of Codd´s principles [Codd 1980]. That means that we have to determine a graph data structure, integrity rules and the elementary operations that can be combined/composed in the view generating function. This resulted in our Property Graph Data Model (PGDM). The second challenge addressed by our research is the definition of the framework itself – even after the graph operators are defined, there is still much to done to deal with views on graphs.

3.1. Property Graph Data Model (PGDM)

PDGM is based on property graphs data structure. A graph G is expressed as: $(G_S, \mathbb{G}_S)$ where G_S is G schema and $\mathbb{G}_S$ is G state. The *"schema"* of a graph is defined by the data design process, which describes the semantic organization of all modelled information and consists of a non empty set of vertex types, V_S, and a set of edge types, E_S. A state of a given graph schema G_S, denoted by $\mathbb{G}_S$, is determined by occurrences of vertex types and edge types at a given time.

A graph *integrity constraint* defines a restriction on its possible states. The goal of a graph integrity constraint is to ensure data integrity and consistency over a graph's life-cycle. A constraint is checked only when an operation that changes the state $\mathbb{G}_S$ is performed, i.e. creation/update/delete of vertices or edges. PGDM has three types of integrity constraints: (i) entity integrity, that adapts the concept of a "primary key", where entities are vertices and edges; (ii) referential integrity, that adapts the concept of a "foreign key" of relational theory; and (iii) domain integrity.

The manipulation of property graphs is based on elementary operators. Some operators uses an intermediate *property graph path*. A property graph path GP is a temporary structure created over the graph database to evaluate an operator or to specify the output graph schema of an operator. A GP is the representation of a linear path connecting vertices and the edges involved in these connections. Table 1 presents an overview of the elementary operators of PGDM. As can be seen, PGDM has seven unary operators and three binary operators. These operators can be progressively composed to create complex operators.

3.2. Architecture

A core idea of our architecture is to separate view definition (namely, the specification of the view generating function) from actual view creation (namely, the execution of the function). Figure 1 gives an outline of this concept, in which these two main functionalities are respectively called "View Builder" and "View Factory". The interface, at the top of this figure, receives requests from users.

	Name	Goal	Input	Parameters	GP	Output		Notation
						Schema	Occurrences	
Unary	Restriction (σ)	Select a subset of data that satisfy a given predicate	G	φ: predicate	Yes	GP graph schema	Subset of $\mathbb{G}_S$ in GP whose vertex/edge attributes satisfy the predicate	$\boldsymbol{\sigma}\ (\varphi)\ G.GP$
	Projection (Π)	Create a new vertex type (l,$\{A\}$)	G	l: label $\{A\}$: set of attributes	Yes	The new vertex type	Subset of vertex/edge attributes of $\mathbb{G}_S$ in GP	$\Pi\ (l, \{A\})$ $G.GP$
	Rename (ρ)	Alter an attribute name in a vertex/edge type	G	V_i: vertex type or E_j: edge type $A_i \vert B_i$: attribute name C: new attribute name	No	G_S changed	$\mathbb{G}_S$	$\boldsymbol{\rho}\ (V_i, A_i, C)\ G$ $\boldsymbol{\rho}\ (E_j, B_i, C)$ G
	Edge Creation (ε)	Connect vertices that satisfy a predicate	G	E_j: edge type φ: predicate to connect vertices	Yes	GP graph schema and the new edge type	$\mathbb{G}_S$ plus new edge instances connecting the vertices whose attributes satisfy the predicate	$\boldsymbol{\varepsilon}\ (E_j, \varphi))$ $G.GP$
	Group (γ)	Create a new vertex type by exposing a common attribute value	G	l: label A_i: attribute for grouping	Yes	The new vertex type	Subset of distinct values of a_i in $\mathbb{G}_S$	$\boldsymbol{\gamma}\ (l, A_i)\ G.GP$
	Attribute Creation (α)	Alter a vertex/edge type by adding a new attribute	G	V_i: vertex type or E_j: edge type C: attribute name f: function or literal	Yes	GP graph schema and new attribute C	$\mathbb{G}_S$ and the new attribute values	$\boldsymbol{\alpha}\ (V_i, C, f)$ $G.GP$ $\boldsymbol{\alpha}\ (E_j, C, f)$ $G.GP$
	Conditional Traversal (τ)	Select a subset of data visited in a traversal	G	v_i: initial vertex E_j: edge type φ: predicate sc: stop condition	No	Subset of G_S	Subset of $\mathbb{G}_S$ visited	$\boldsymbol{\tau}$ $(v_i, E_j, \varphi\ , sc)$
Binary	Union ($\cup$)	Select all data of both inputs	G_1, G_2	None	Yes	G_{S1} plus G_{S2}	$\mathbb{G}_{S1}$ plus $\mathbb{G}_{S2}$	$\bigcup$ $G_1.GP, G_2.GP$
	Difference ($\setminus$)	Select a disjoint subset of data	G_1, G_2	$\{I\}$: set of compatible attributes	Yes	G_{S1}	Subset of $\mathbb{G}_{S1}$ not present in $\mathbb{G}_{S2}$	$\setminus\ (\{I\})$ $G_1.GP, G_2.GP$
	Intersection ($\cap$)	Select a common subset of data	G_1, G_2	$\{I\}$: set of compatible attributes	Yes	G_{S1}	Subset of $\mathbb{G}_{S1}$ common to $\mathbb{G}_{S2}$	$\bigcap\ (\{I\})$ $G_1.GP, G_2.GP$

Table 1: Elementary Operators

Users can either *define a view* (i.e., specifying the view generating function) (through "View Builder") or *construct a view* (computing the view generating function) through "View Factory". The "View Factory" is composed by four modules: **Parse**, **Process**, **Check** and **Create**. The persistence layer of the framework is composed of a graph database management system; it provides management mechanisms to store graph data and view generating functions. Views can eventually be materialized, thereby becoming part of the graph database.

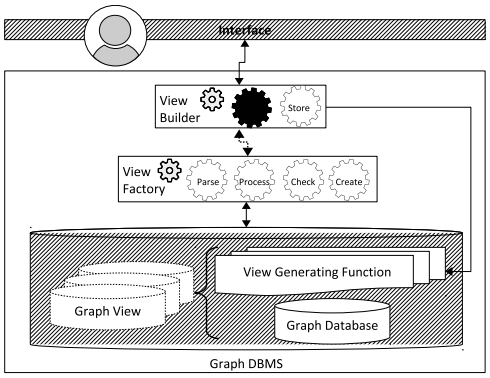


Figure 1. Graph-Kaleidoscope Architecture

4. Related Work

The first proposal for views in graphs appeared in the 80's, together with the first graph data models. Kunii's research [Kunii 1983] very briefly approaches graph views in her thesis, proposing a view definition composed by a list of vertex type definitions and list of link type definitions. With the arrival of graph database management systems, some view mechanisms have started to appear in the last years. An example of construction of views in graph databases appears in [Fan et al. 2014]. The research concerns the development of efficient algorithms to answer graph queries using a set of graph views. Here, a graph view is a graph pattern query. Similar to us, [Ghrab et al. 2015] proposes a graph data model. The authors define a graph data structure, integrity rules and a set operators based on GraphQL and UML builders, based on the idea of bind graph patterns with graph templates. However it does not properly define or formalize the operators. The notion of graph views is also proposed for domains in which relationships are analyzed under the so-called *complex networks*, which are sometimes materialized in graph databases (e.g., [Lysenko et al. 2016]). In this context, view-like mechanisms are proposed to extract a portion of a network for analysis, but the construction of such views is based on invoking queries using the database language.

As can be seen, the general idea of views in graphs is present in all approaches – they all deal with graph data and adopt some kind of view definition mechanism/paradigm to describe how to derive graph data from other data graphs. The main difference is the approach adopted: many employ graph patterns to formulate graph queries. We, instead, define generic operators that can be used to construct views over generic graph databases. Our operators allow to achieve all purposes of a view, mainly aggregation and restructuring not cover by other works. Therefore, a view definition is also considered as graph query over a set of base graphs and/or view graphs for deriving new view graphs.

5. Conclusions and Ongoing Work

This paper presented Graph-Kaleidoscope, a framework that supports management of views in graph databases. Views, here, are provided to let experts exploit multiple perspectives from the underlying data. Our specification is general enough and can be used in many research contexts, in which data present the same kind of characteristics (e.g., for domains that deal with complex multi-scale networks, such as health or biodiversity).

This research contributes therefore towards solving problems of multi-perspective research in applications that are characterized by inter-disciplinarity (and thus multiple ways of analyzing a problem). Graph-Kaleidoscope is characterized by: (1) use of graph databases to store and analyze datasets of highly connected data; (2) adapting the concept of views from relational databases to represent the idea of focus; and (3) specification and implementation of a graph view framework to support views over graph databases – views that are themselves graphs.

Ongoing work covers both theoretical and implementation issues. Implementation efforts require improvement in the framework's code, in particular considering the underlying data catalogue, and generating function storage and indexing. Another direction involves designing and developing a user-friendly interactive interface, to help users define and explore views. From a theoretical point of view, we might think of considering other operators, e.g., defined as a combination of our elementary operators.

References

Angles, R. and Gutierrez, C. (2008). Survey of graph database models. *ACM Comput. Surv.*, 40(1):1:1–1:39.

Atkinson, M., DeWitt, D., Maier, D., Bancilhon, F., Dittrich, K., and Zdonik, S. (1992). Building an object-oriented database system. chapter The Object-oriented Database System Manifesto, pages 1–20. San Francisco, CA, USA.

Codd, E. F. (1980). Data models in database management. *SIGMOD Rec.*, 11(2):112–114.

Fan, W., Wang, W., and Wu, Y. (2014). Answering Graph Pattern Queries Using Views. In *Proc. 30th International Conference Data Engineering - ICDE*, pages 167–176.

Ghrab, A., Romero, O., Skhiri, S., Vaisman, A. A., and Zimányi, E. (2015). Grad: On graph database modeling. *CoRR*, abs/1602.00503.

Hey, T., Tansley, S., and Tolle, K., editors (2009). *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, Washington.

Kunii, H. (1983). *Graph data language : a high level access-path oriented language*. PhD thesis, University of Texas at Austin.

Lysenko, A., RoznovăŢ, I. A., Saqi, M., Mazein, A., Rawlings, C. J., and Auffray, C. (2016). Representing and querying disease networks using graph databases. *BioData Mining*, 9(1):23.

Santanche, A., Longo, J., Jomier, G., Zam, M., and Medeiros, C. B. (2014). Multi-focus research and geospatial data - Anthropocentric concerns. *JIDM*, 5(2):146–160.

Prontuário Eletrônico de Paciente apoiado em técnicas de Linhas de Produtos de Software

Lucas F. Ferreira[1], **Cecília M. F. Rubira**[1], **Sheila K. Venero**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Av Albert Einstein – Campinas – SP – Brazil

Abstract. *The Electronic Medical Record is an important technological tool that brings benefits to the healthcare industry. Although there are several systems and initiatives for adopting these kinds of technologies, they often do not meet the specific needs of different medical specialities that use standards and metrics to perform their activities with quality. This paper aims to conduct a Software Products Product Line approach combined with Component Based Development to support the development of Electronic Medical Records. This approach will be assessed by the development of a system called OncoPEP that will satisfy the criteria of the Oncologic Quality standard.*

Resumo. *O Prontuário Eletrônico do Paciente é uma importante ferramenta tecnológica que traz benefícios para a área da saúde. Apesar de existirem diversos sistemas e iniciativas para a adoção desses sistemas, muitas vezes eles não satisfazem as necessidades de ambulatórios de especialidades que usam padrões e métricas para realizar suas atividades com qualidade. Este trabalho tem como objetivo realizar uma abordagem de Linhas de Produtos de Software juntamente com Desenvolvimento Baseado em Componentes para apoiar o desenvolvimento de Prontuários Eletrônicos de Pacientes. A abordagem será concretizada por meio do desenvolvimento de um sistema nomeado OncoPEP que atende critérios do padrão de qualidade oncológica.*

1. Introdução

O uso da Tecnologia da Informação na área da saúde não só permite agilizar os diversos processos dentro de um centro de saúde, economizar tempo, recursos e custos, mas também aumentar a qualidade do atendimento aos pacientes e a produtividade dos profissionais da saúde. Uma das ferramentas mais importantes que visam trazer muitos benefícios à área da saúde é o Prontuário Eletrônico do Paciente (PEP), definido como um conjunto de informações computadorizadas referentes à saúde de um sujeito, com o objetivo de assegurar que serviços de saúde estejam integrados de modo contínuo, com eficiência e privacidade de dados, permitindo o acesso a essas informações somente a usuários autorizados [ISO 2005]. Quando bem projetado, o PEP traz benefícios como a diminuição do volume de papéis, facilidade na recuperação de informação, acesso simultâneo à informação, entre outros.

No Brasil, as Unidades Básicas de Saúde têm obrigação, por ordem do ministério de saúde (PORTARIA Nº 1.412 de 2013 [Ministério da Saúde 2013]), de implantar o e-SUS Atenção Básica (e-SUS AB), composto por dois sistemas (Coleta de Dados Simplificado (CDS) e Prontuário Eletrônico do Cidadão (PEC)). Apenas aproximadamente 24%

das unidades básicas de saúde já adotaram um sistema, sendo que mais de 70% dos sistemas utilizados são sistemas próprios e privados [Portal Brasil 2016]. Ainda exitem muitos problemas básicos para a implantação do e-SUS AB, como a falta dinheiro para comprar computadores, o pagamento da internet, a digitalização das informações dos pacientes, o treinamento dos usuários, etc.

Além disso, mesmo em ambulatórios que possuem uma infraestrutura de computadores e acesso à internet, o PEP fornecido pelo governo não atende às necessidades das diferentes especialidades da saúde, como por exemplo, o uso de padrões e métricas para manter a qualidade das atividades. Esse é o caso da Clínica de Oncologia do Hospital das Clínicas (HC) da Unicamp, onde os processos realizados no ambulatório seguem os critérios do programa *Quality Oncology Practice Initiative* (QOPI) [ASCO 2017] de qualidade de atendimento (vide Seção 3). Os sistemas pagos são ainda mais inviáveis devido à falta de orçamento e o fato de possivelmente estarem em língua estrangeira, que dificulta ainda mais o uso desses sistemas por usuários brasileiros.

O trabalho em andamento tem como objetivo desenvolver um PEP nomeado de OncoPEP para a Clínica Oncológica do HC da Unicamp, que atenda os critérios QOPI. Será adotada uma abordagem de linha de produto de software, apoiada pelo desenvolvimento baseado em componentes, com o intuito de facilitar o reúso e modificação de código para necessidades específicas de outros ambulatórios.

Espera-se que o OncoPEP possa ser reutilizado em outras clínicas. Na Seção 2 temos a contextualização do problema. Na Seção 3 é apresentada a abordagem que está sendo realizada para o desenvolvimento do sistema. Por fim, na Seção 4 é apresentado um breve resumo da abordagem proposta e o que vêm a seguir.

2. Contextualização do Problema

O prontuário do paciente físico, isto é, pasta com papéis é a forma mais tradicional de manter informações de um paciente, e o seu uso resulta em alguns problemas que comumente são resolvidos com a utilização do PEP. Nos prontuários físicos, a recuperação de informação é um processo de busca manual, e como muitas vezes o prontuário é utilizado por vários profissionais, as folhas são inseridas em ordem aleatória, dificultando ainda mais a obtenção dos dados. Em alguns casos, folhas podem não estar na pasta ou não serem encontradas, resultando na reexecução de exames, aumentando o custo e o tempo de tratamento do paciente. Outra questão é o fato das anotações serem realizadas manualmente, o que muitas vezes gera falta de compreensão por outros profissionais, ou em casos extremos pelo mesmo profissional. Além disso, esses prontuários devem ser armazenados por 20 anos mesmo que não estejam sendo utilizados, gerando um grande volume de papéis e resultando na reserva de grandes espaços para seu armazenamento. Na Clínica Oncológica do HC da Unicamp, diversas informações são preenchidas repetidamente em formulários, cada vez que o paciente desenvolve uma atividade na clínica. [Galvão and Ricarte 2012][Perjons et al. 2005]

O PEP resolve parte dessas dificuldades apresentadas, devido às suas características inerentes de ser um sistema computadorizado, que pode armazenar grande quantidade de informação em pequeno espaço físico, recuperar informação rapidamente, apresentar dados sem se preocupar com a ortografia, acessar simultaneamente as informações, compartilhar informações com outros profissionais, evitar a reinserção de informações já

armazenadas em formulários, entre outros.

3. Abordagem Proposta

O OncoPEP irá atender os critérios QOPI (*Quality Oncology Practice Initiative* [ASCO 2017]) com o objetivo de melhorar o acompanhamento realizado por médicos, nutricionistas, enfermeiros, assistentes sociais e psicologistas aos pacientes da Clínica Oncológica do HC da Unicamp. QOPI é um programa de avaliação de qualidade, desenvolvido em 2002, por profissionais da *American Society of Clinical Oncology* (ASCO) dedicados à oncologia, para promover a excelência no tratamento do câncer, ajudando as práticas de tratamento a criar uma cultura de auto-exame e aperfeiçoamento. O programa atualmente é guiado por oncologistas, enfermeiros, pesquisadores de serviços de saúde e especialistas em qualidade.

O OncoPEP será desenvolvido com uma abordagem de linha de produto de software (LPS) [Clements and Northrop 2002] apoiada pelo desenvolvimento baseado em componentes (DBC) [Bachmann et al. 2000]. LPS é uma abordagem para reúso de artefatos que são comuns para uma família de sistemas que permite gerenciar a variabilidade desses sistemas (possibilidade de um artefato ou software ser modificado para ser utilizado em diferentes contextos), resultando em sistemas com características específicas. O DBC visa reduzir o custo e o tempo de desenvolvimento de sistemas através da união de componentes pré-fabricados. O DBC reduz o acoplamento separando a especificação de componente de sua implementação, favorecendo a reutilização.

Quatro PEPs existentes foram explorados, a fim de extrair funcionalidades comuns alternativas. Os sistemas explorados foram: GNU Health [GNU Health 2017], OpenEMR [OpenEMR 2017], OpenVista [Medsphere 2017] e GEMED Oncologia [GEMED 2017]. Os sistemas GNU Health e OpenEMR, além de ter disponível sua documentação, também foi possível executar suas versões de demonstrações. Já nos sistemas OpenVista e GEMED Oncologia, suas funcionalidades foram extraídas de suas respectivas documentações. Na Tabela 1 é apresentado uma comparação entre a existência das funcionalidades dos quatro sistemas. As funcionalidades presentes nos quatro sistemas podem ser consideradas como importantes, contudo, não necessariamente todas estarão presentes no OncoPEP.

Um levantamento de processos de negócio foi realizado, permitindo identificar os principais processos desenvolvidos na clínica de oncologia. Na clinica oncológica o paciente possui duas entradas possíveis: para consulta ou direto para quimioterapia. A entrada para consulta é composta por: Caso Novo, Seguimento (o paciente já não tem a doença mas precisa de um seguimento contínuo da evolução), Suporte (paciente terminal com tratamento paliativo) e Quimioterapia (o paciente realiza a consulta e passa para a infusão da quimioterapia). Em consultas do tipo Caso Novo, o paciente passará obrigatoriamente por avaliação do serviço social, nutrição, psicologia além da consulta médica. Já nos outros casos, os serviços são prescritos de acordo com a necessidade.

Na entrada do paciente direto para a quimioterapia ele se dirige diretamente para a equipe de enfermagem que realizará uma avaliação de sintomas e sinais vitais básicos, como também a avaliação de exames de rotina pelo médico antes da infusão.

Após o reconhecimento dos processos de negócios foi feita a modelagem dos requisitos que contou com a ajuda de um grupo da disciplina MO620B lecionada no pro-

Tabela 1. Funcionalidades de sistemas existentes. 1 GNU Health, 2 OpenEMR, 3 OpenVista, 4 GEMED Oncologia

Administração do Hospital	**1**	**2**	**3**	**4**
Gestão de Usuários do sistema	●	●	●	●
Gestão de Doenças e Padrões de Procedimentos	●	●	●	●
Contabilidade	●	●	●	●
Agendamento/Programação	●	●	●	●
Gestão de Suprimentos	●		●	●
Gestão de Fornecedores	●			●
Gestão de Colaboradores	●		●	
Recepção do Hospital				
Triagem Enfermagem			●	●
Cadastro do Paciente	●	●	●	●
Procedimento Médico				
Avaliação e Consulta Médica de Pacientes	●	●	●	●
Histórico do Paciente	●	●	●	●
Prescrição Médica	●	●	●	●
Armazenamento de Exames	●	●	●	●
Suporte de Riscos Hereditários	●			
Gestão de Cirurgias/Quimioterapia	●		●	●
Assistência Social				
Avaliação do Serviço Social	●		●	
Nutricionista				
Consulta Nutricionista			●	●
Suporte ao Estilo de Vida	●			
Psicóloga(o)				
Consulta Psicológica			●	●
Hotelaria				
Gestão de Internações	●		●	
Comunicação com o Paciente				
Portal do paciente		●	●	●
Prevenção				
Suporte a ações preventivas para a mulher e pré-natal	●			

grama de pós-graduação do Instituto de Computação. O modelo conceitual do OncoPEP é o prontuário, que possui diagnósticos e é acessado por pacientes, médicos, enfermeiros, nutricionistas, psicólogos e assistentes sociais de acordo com a necessidade cada um. Um diagnostico é composto por dores, queixas, exames e prescrição medicamentosa. As partes acima possuem modelos de diagrama UML (classe e sequência) que define mais especificamente cada uma das partes e a interação entre elas. Uma visão geral e simplificada da modelagem do OncoPEP pode ser vista na Figura 1.

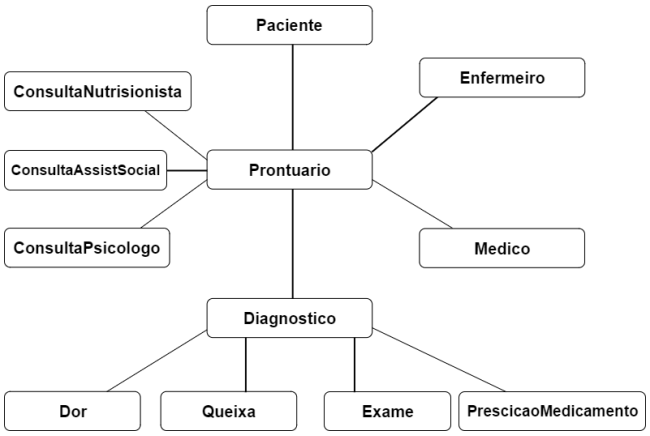


Figura 1. Modelo Conceitual simplificado do OncoPEP.

4. Conclusões

O trabalho tem como objetivo utilizar uma abordagem de Linhas de Produto de Software apoiada pelo Desenvolvimento Baseado em Componentes e propor uma arquitetura que facilite o reúso de código para o desenvolvimento de Prontuários Eletrônicos de Pacientes. Um sistema chamado OncoPEP será desenvolvido para a Clínica Oncológica do Hospital das Clínicas da Unicamp com base na arquitetura que será proposta e que atenda os critérios de qualidades específicos de oncologia (QOPI).

Referências

ASCO (2017). Quality Oncology Practice initiative - QOPI. `http://www.instituteforquality.org/quality-oncology-practice-initiative-qopi`. Acesso em 11 de março de 2017.

Bachmann, F., Bass, L., Buhman, C., Dorda, S. C., Long, F., Robert, J., Seacord, R., and Wallnau, K. (2000). Technical concepts of component-based software engineering. *Technical Report CMU/SEI-2000-TR-008, Software Engineering Institute*.

Clements, P. and Northrop, L. (2002). Software product lines: practices and patterns. *Addison-Wesley*.

Galvão, M. C. B. and Ricarte, I. L. M. (2012). *Prontuário do Paciente*. Guanabara Koogan LTDA.

GEMED (2017). GEMED: A solução necessária para Gestão em Saúde, customizada para Organizar e Controlar a sua Clínica. `http://www.interprocess.com.br/gemed/`. Acesso em 7 de março de 2017.

GNU Health (2017). GNU Health: a Free/Libre project for health practitioners, health institutions and governments. http://health.gnu.org/. Acesso em 7 de março de 2017.

ISO (Geneva: 2005). International organization for standardization. health informatics: Electronic health record – definition, scope and context. *ISO/TR 20514*.

Medsphere (2017). OpenVista. http://www.medsphere.com/open-vista/enterprise-wide-integrated-solution. Acesso em 7 de março de 2017.

Ministério da Saúde (2013). PORTARIA Nº 1.412, DE 10 DE JULHO DE 2013. Institui o SISAB. http://bvsms.saude.gov.br/bvs/saudelegis/gm/2013/prt1412_10_07_2013.html. Acesso em 13 de março de 2017.

OpenEMR (2017). OpenEMR: BETTER SOFTWARE. BETTER OUTCOMES. BETTER HEALTHCARE. http://www.open-emr.org/. Acesso em 7 de março de 2017.

Perjons, E., Wangler, B., Wäyrynen, J., and Åhlfeldt, R. (2005). Introducing a process manager in healthcare: an experience report. *Health Informatics Journal*, 11.

Portal Brasil (2016). Postos de saúde devem justificar não adesão ao prontuário eletrônico. http://www.brasil.gov.br/saude/2016/11/postos-de-saude-devem-justificar-nao-adesao-ao-prontuario-eletronico. Acesso em 13 de março de 2017.

Testes de segurança baseado em modelos de estados e sua aplicação ao Moodle Quiz

Narcísio Mula, Eliane Martins

Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 15.064 – 91.501-970 – Campinas – SP – Brazil

narciomula@gmail.com, eliane@ic.unicamp.br

Abstract. *Security Testing aims to determine if a system meets your security requirements, and serves both to determine whether security features have been implemented properly and to identify vulnerabilities. Model-Based Security Testing (MBST) is based on behavioral models. It is a very useful for the industry, since in addition to systematizing the specification of security requirements, it also allows the (semi) automatic treatment of the generation of security test cases. In this work we present a method for security tests based on models to detect vulnerabilities in Moodle, specifically, in the Quiz module.*
Keywords: *security testing- model-based testing - attack patterns - XSS attack.*

Resumo. *Testes de Segurança têm por objetivo determinar se um sistema satisfaz os seus requisitos de segurança, e servem tanto para determinar se as funcionalidades de segurança foram implementadas devidamente, quanto para identificar vulnerabilidades. Teste de Segurança Baseado em Modelo (ou MBST, do inglês Model-Based Security Testing) se baseia em modelos de comportamento. É uma área muito útil para a indústria, pois além de sistematizar a especificação de requisitos de segurança, permite também tratar, de maneira (semi-)automática, a geração dos casos de teste de segurança. Neste trabalho apresentamos um método para testes de segurança baseado em modelos visando detetar vulnerabilidades no Moodle, especificamente, no módulo Quiz.*
Palavras-chave*: testes de segurança-testes baseados em modelos-padrões de ataques- ataque XSS.*

1. Introdução

Moodle, acrônimo de Modular Object-Oriented Dynamic Learning Environment (Ambiente Modular de Aprendizagem Dinâmica Orientada a Objetos), é uma plataforma de apoio à aprendizagem baseada em software livre, desenvolvida de forma colaborativa por uma comunidade virtual que engloba diversos perfis profissionais, desde professores a administradores e designers.

Disponível em mais de 100 idiomas, o Moodle é um sistema consagrado, usado em 232 países, com mais de 74 mil sites registrados (https://moodle.net/stats/). De acordo com o CVE (Common Vulnerabilities and Exposures), até 2016, foram relatadas 318 vulnerabilidades no Moodle, das quais 23,3% permitem ataques de XSS (Cross-site scripting), 22,6% permitem o vazamento de informação, e 14,5% permitem a usuários

violar alguma restrição (http://www.cvedetails.com/product/3590/?q=Moodle). Testar a segurança do Moodle é útil para os desenvolvedores melhorarem mecanismos de proteção. De acordo com estudos feitos em 2009, 90% dos incidentes de segurança são causados por exploração de vulnerabilidades existentes [Schieferdecker et al. 2012].

Testes de Segurança têm por objetivo determinar se um sistema satisfaz os seus requisitos de segurança, ou seja, se satisfaz a propriedades tais como confidencialidade, integridade e disponibilidade [Avizienis et al. 2004]. Existem diversas técnicas para testar a segurança. Uma técnica muito utilizada denomina-se fuzz testing, e consiste em introduzir entradas inválidas ao sistema em testes (SeT) e observar se o sistema aceita, ao invés de rejeitar, essas entradas. O objetivo é determinar se o fato de aceitar entradas inválidas pode ser explorado por um atacante, fazendo com que o sistema deixe de prestar o serviço esperado (crash ou abort), ou ainda, se passa a ter um comportamento não esperado. Esta técnica se tornou popular porque é fácil de usar e requer pouco conhecimento sobre o SeT, i.e., não necessita de análise de código e nem da especificação, pois é baseada nos dados da interface do SeT. Existem duas formas de se produzir as entradas inválidas: por mutação ou por geração [Miller et al. 2007]. Nas técnicas baseadas em mutação, as entradas inválidas são produzidas pela modificação de entradas válidas coletadas que podem ser geradas aleatoriamente ou usando uma heurística. As técnicas baseadas em geração, partem de uma especificação do formato do arquivo ou das mensagens e produz entradas que a violem .

As técnicas baseadas em dados não levam em conta o estado do sistema. Em consequência, para sistemas em que o efeito das entradas varia conforme o estado em que o sistema se encontra, é necessário produzir um grande volume de entradas para revelar falhas que dependem da combinação estado-entrada. Além de não garantir que todos os estados sejam atingidos, os testes não são reprodutíveis, o que torna difícil determinar a combinação estado-entrada que ocasionou um defeito (failure).Os testes baseados em modelos são mais adequados neste caso, pois se baseiam no comportamento do SeT e, portanto, são mais capazes de encontrar bugs que causem violações de requisitos. Essas técnicas são denominadas de model-based fuzzing [Schneider et al. 2012], mas no contexto deste trabalho usaremos o termo teste de segurança baseados em modelos ou MBST (do Inglês, Model Based Security Testing). Neste trabalho propomos um método de MBST, baseado em estados. O modelo é usado para representar o comportamento do atacante.

O texto contém as seguintes seções após a introdução, a seção 2 apresenta brevemente os fundamentos de MBST. A seção 3 descreve o estudo de caso, enquanto o método de teste utilizado é descrito na seção 4. A seção 5 conclui o trabalho.

2

2. Testes de Segurança Baseados em Modelo

Testes consistem em executar um sistema ou componente com o intuito de detectar diferenças entre o comportamento obtido e o esperado, de acordo com a definição do IEEE Standard Glossary of Software Engineering Terminology 610.12-1990. Para isso é necessário submeter o sistema em testes (SeT) a um conjunto finito de casos de teste, devidamente selecionados a partir de um domínio de entradas possíveis que é potencialmente infinito. A seleção é feita com o uso de critérios [Goodenough e Gerhart, 1975], que são condições que um conjunto de testes deve satisfazer. Nos testes de caixa branca, estas condições são determinadas em termos do código; um critério poderia ser, por exemplo: "cobrir todas as instruções do programa". Já nos testes caixa preta, as condições são determinadas em termos da especificação.

Os testes baseados em modelos (MBT), é uma técnica de testes em que os critérios são definidos em termos de um modelo que representa o comportamento do SeT e/ou de seu ambiente [Utting et al. 2012]. Os passos de um processo de testes usando MBT são: i) modelagem, em que um modelo é construído a partir dos requisitos. Este é designado como modelo de teste, pois é projetado para atender aos objetivos dos testes; ii) geração dos casos de teste, guiada por critérios de seleção. Esses critérios podem basear-se nas funcionalidades do sistema ou em elementos do modelo; por exemplo, cobertura de todas as funcionalidades, ou cobertura de todas as transições; iii) concretização; iv) execução dos casos de teste, em que as entradas de teste são fornecidas ao SeT e as saídas produzidas são coletadas; v) análise dos resultados, em que as saídas observadas são comparadas com as saídas esperadas e um veredicto é emitido: passou (√), falhou (×) ou inconclusivo (?).

A breve descrição acima foi baseada na taxonomia proposta por Utting et al. (2012). No caso de MBST, baseamo-nos na taxonomia introduzida por Felderer et al. (2016). De acordo com Tian-yang et al. (2010) [apud Felderer 2016] pode-se testar a segurança do ponto de vista da funcionalidade ou das vulnerabilidades. Testes de funcionalidades de segurança visam determinar se os mecanismos de segurança atendem às especificações. Já os testes de vulnerabilidades de segurança, abordam a descoberta de vulnerabilidades que são causadas por falhas de projeto dos mecanismos de segurança ou por defeitos de software. Os testes de vulnerabilidades de segurança simulam ataques em um sistema e requerem que o testador pense como um atacante e são mais difíceis de automatizar.

3

3. O Estudo de caso

Neste trabalho, vamos testar o Módulo Quiz da plataforma Moodle.

4. O Método de testes utilizado

De acordo com Schieferdecker et al. (2012), o uso de testes de segurança requer os passos descritos na seção 2. Sendo assim, adotamos os seguintes passos:

4.1. Identificação dos objetivos para os testes de segurança

Pretendemos levantar as possibilidades de um atacante explorar vulnerabilidades deste módulo para cometer fraudes do tipo: a) Um atacante pode ter acesso as provas dos alunos; b) O atacante pode se passar por um professor e ter acesso às questões; c) atacante pode se passar por um professor e ter acesso às notas, para modificá-las;

4.2. Modelagem

Inspirado em Bozic et al.(2014), o modelo de testes representa o comportamento do atacante e é obtido a partir de padrões de ataques, que descrevem: os objetivos do atacante; as pré-condições, as ações individuais que realizam o ataque e as pós-condições.

Objetivo: Realizar o ataque XSS, **Pre Condição:** O atacante tem de poder submeter dados ao sistema em teste. **Ações: 1**). O atacante conecta-se a aplicação através da sua URL, comunicando através de HTTP. Se a URL for válida retorna o código OK. E sinaliza o estado Conectado. 2). Encontrar pontos de entrada de dados (Ex: caixa de texto, urls etc). Se encontrados, são retornados ao atacante que os armazena em variáveis para uso na próxima ação. Sinaliza o estado Parsed. 3). O atacante introduz as entradas maliciosa (Strings de XSS) nos pontos de entrada de dados (variáveis string). Para tal tem de recorrer ao um arquivo de entradas XSS previamente preparada de acordo com o critério de teste escolhido. 4). Comparar saída com os valores esperados. **Pós Condição:** Caso a entrada maliciosa encontra um ponto de dados não sanitizado, sinaliza a existência de uma vulnerabilidade XSS

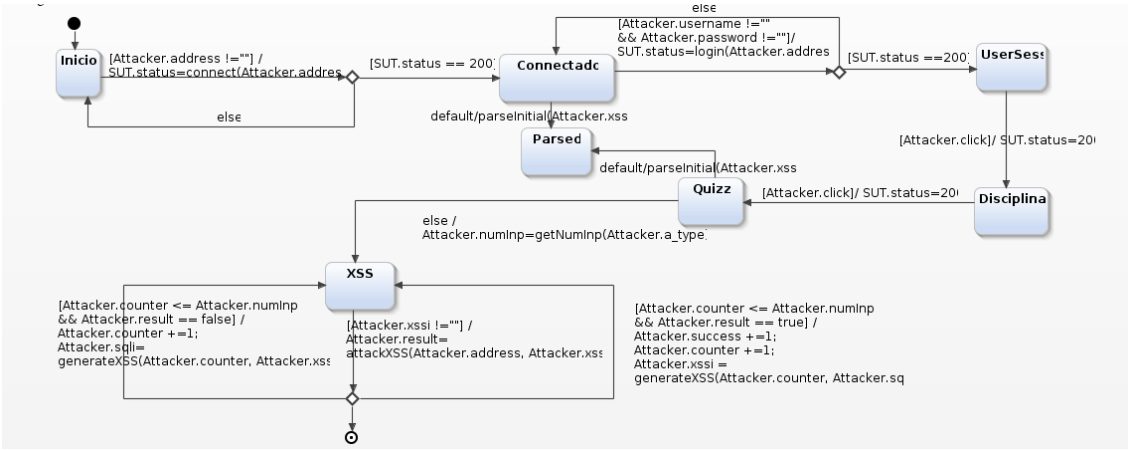


Figura 1: Padrão de ataque para SQLI

4

4.3. Escolha dos critérios de seleção dos testes

Como geralmente há um grande número de casos de teste que podem ser obtidos a partir de um modelo de estados, são usados diversos critérios de seleção para determinar um conjunto reduzido de casos de teste. Um exemplo de critério é o de cobertura de transições, i.e., o conjunto de testes deve conter casos de teste que visitem todas as transições do modelo de estados. Porém, de acordo com Schieferdecker et al. (2012), estes critérios não são suficientes, pois são baseados no fluxo de controle do sistema, e os ataques visam mais os dados. Assim, para este caso, seguiremos o critério para seleção dos dados, aplicando Strings de ataque XSS.

4.4. Geração e execução dos casos de teste

Há dois modos de realizar a execução de testes: online e off-line. Na execução online o modelo é executado em paralelo com a aplicação e off-line quando se geram testes para depois executar separadamente do modelo. Neste trabalho a geração será online.

4.5. Análise dos resultados

No XSS, o sucesso é detectado analisando a resposta na busca de elementos HTML inesperados, que normalmente não devem ocorrer. Se o atacante enviar uma carga (payload) útil dentro de um script, o mesmo script é enviado de volta para uma vítima.

4.6 Avaliação do método proposto

Segundo Felderer (2016), para mostrar evidências de que o método proposto é útil, deve se considerar a maturidade do sistema em que o método é aplicado. Um outro aspeto é relativo à obtenção de medidas quantitativas de efetividade e da eficiência. Para medir a efetividade, uma proposta consiste em avaliar o número de vulnerabilidades detectadas, pelo total de casos de teste realizados. Para medir a eficiência, precisamos comparar o custo da detecção de falhas de vulnerabilidade com o de outras propostas.

5. Estado atual e trabalhos futuros

Até o momento já foi implementado o modelo do atacante. Estamos a trabalhar na determinação de critério de seleção de testes dos dados de entrada, pois por ora temos poucos dados de entrada para simular os ataques. Poderemos incluir outros critérios de testes além dos referidos na seção 4.3.

5

6. References

Avizienis, Algirdas, et al. "Basic concepts and taxonomy of dependable and secure computing." *IEEE transactions on dependable and secure computing* 1.1 (2004): 11-33.

Bozic, Josip, and Franz Wotawa. "Security testing based on attack patterns." *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on*. IEEE, 2014.

Chandramouli, Ramaswamy, and Mark Blackburn. "Automated testing of security functions using a combined model and interface-driven approach." *System Sciences, 2004.*

Felderer, Michael, et al. "Model-based security testing: a taxonomy and systematic classification." *Software Testing, Verification and Reliability* 26.2 (2016): 119-148.

Felderer, Michael. "Current State and Challenges for Model-Based Security Testing". SECTEST 2015. Graz, Austria. Access on March 2017: https://www.slideshare.net/m_felderer/current-state-and-challenges-for-modelbased-security-testing .

Goodenough, John B., and Susan L. Gerhart. "Toward a theory of test data selection." *IEEE Transactions on software Engineering* 2 (1975): 156-173.

Miller, Charlie, and Zachary NJ Peterson. "Analysis of mutation and generation-based fuzzing." *Independent Security Evaluators, Tech. Rep* (2007). Access on Feb/2017: ftp://fre-mirror.picosecond.org/defcon/defcon15-cd/Speakers/Miller/Whitepaper/dc-15-miller-WP.pdf

Morais, Anderson, Ana Cavalli, and Eliane Martins. "A model-based attack injection approach for security validation." *Proceedings of the 4th International Conference on Security of Information and Networks*. ACM, 2011.

Büchler, Matthias, Johan Oudinet, and Alexander Pretschner. "Semi-automatic security testing of web applications from a secure model." *Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference on*. IEEE, 2012.

[Schneider et al. 2012] Schneider, Martin, et al. "Online model-based behavioral fuzzing." *Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on*. IEEE, 2013

Schieferdecker, Ina, Juergen Grossmann, and Martin Schneider. "Model-based security testing." *arXiv preprint arXiv:1202.6118* (2012).

Utting M, Pretschner A, Legeard B. A taxonomy of model-based testing approaches. Software Testing Verification and Reliability 2012; 22(2):297–312

Using ontologies and machine-learning techniques for malware identification in Android environment

Luiz C. Navarro[1], **Ricardo Dahab**[1]

[1]Institute of Computing - University of Campinas (Unicamp), Campinas, SP, Brazil

luiz.navarro@students.ic.unicamp.br, rdahab@ic.unicamp.br

Abstract. *The objective of this article is to expose results obtained with the use of ontologies and machine learning in the creation and analysis of a model of the ecosystem of Android smartphone applications. More than results on the identification of malware using only high-level information about resources, interfaces and permissions declared on the application manifests, the methodology employed created a framework to analyze systems or phenomena described by complex graphs, identifying most discriminative structures to reconstruct a model simpler and easier to analyze. An ontology for modeling the Android permissions ecosystem was defined and populated with information from manifests of 2,790 apps (181 system, 118 partners, 860 benign downloaded and 1631 malware) resulting in a semantic graph of 40,000 nodes and 76,000 edges. Applying the Bag of Graph technique and the decision forest machine learning algorithm we could isolate 51 most discriminative features responsible for the best classifier performance on malware identification. It allowed the reconstruction of a much simpler graph, which is possible to plot and visualize.*

Resumo. *O objetivo deste artigo é expor os resultados obtidos com o uso de ontologias e aprendizado de máquina na criação e análise de um modelo do ecossistema de aplicativos de smartphones Android. Mais do que os resultados na identificação de malware usando apenas informações de alto nível sobre recursos, interfaces e permissões declaradas nos manifestos de aplicação, a metodologia utilizada criou uma estrutura para analisar sistemas ou fenômenos descritos por gráficos complexos, identificando as estruturas mais discriminativas que permitem reconstrução de um modelo simplificado e mais fácil de analisar. Uma ontologia para modelagem do ecossistema de permissões do Android foi definida e preenchida com informações dos manifestos de 2.790 aplicativos (181 do sistema, 118 de aplicações de parceiros, 860 downloads benignos e 1631 malware) resultando em um gráfico semântico de 40.000 vértices e 76.000 arestas. Aplicando a técnica do Bag of Graph e o algoritmo de aprendizado da máquina de florestas de decisão, isolamos 51 caracteristicas mais discriminativas responsáveis pelo melhor desempenho do classificador na identificação dos malware, o que permitiu a reconstrução de um grafo mais simples e possível de visualizar.*

1. Introduction

This work started as part of a broader research project of creating models which allow us to analyze systems from a structural point of view, identifying modules, properties and relationships which are more vulnerable or more critical under a security or reliability point

of view. To achieve the target the problem was divided and reduced to smaller experiments to check techniques, and one of them produces the results exposed here which shows a promising way for modeling and analyze problems involving structural relationships in a complex ecosystem.

In this paper, we applied the method to a complex ecosystem consisting of a commercial Android smartphone loaded with the operating system and applications provided in the factory software image, a bunch of benign applications downloaded from the Android store, and applications stored in our malware repository. The target was established to analyze the permissions [Enck et al. 2009, Felt et al. 2011, Felt et al. 2012] and resources which are related to malware applications, and we used as a single source the manifests contained inside the application package (APK files) which are publicly available and does not require any technique to reverse engineer, code execution monitoring or treat software obfuscation at the code level [Enck et al. 2010, Chanajitt et al. 2016, Das et al. 2016].

It is important to realize at this point that the target here is not to find resources and relationship direct involved in the attack itself, but understand what structures and relationships are important in malware activity. Just as an example, a malware that steals contacts information needs the network access to send this information out of the smartphone environment, then although the network resources are not directly involved in the attack they are an important part of the malware structure and relationships.

First of all, to capture the information available on the application manifests we need to choose how to build a model which can capture the entities and their relationships and provide the analysis's foundation. A database appeared to be a good choice to store all the data captured and provided some tools to query and analyze the environment, but ontologies [Gruber 2009, Guarino et al. 2009] on this case shows more advantages for building models, as by definition they are focused on semantics, standardization of concepts, open and easy to evolve as new concepts, structures and relationships needed to be represented.

As an ontology model was built and populated with data captured from the application manifests this simple model became so complex for trivial analysis using statistical and query tools to identify the important relationships which are related to the malware behavior. Then the Bag of Graphs [Silva et al. 2014] technique was used to convert the graph into feature vectors, and a Random Forest [Breiman 2001, Criminisi et al. 2012, Murphy 2012] classifier was trained, and the features importance recursively analyzed producing the ranking of most discriminant features. Looking at the performance of ranked features that produced the best results then the most discriminant was selected to reconstruct a simpler graph which could be visualized.

The little history above explains and summarizes how the work presented here evolved. The results shown in this paper demonstrate that the techniques applied in the proposed method have a significant potential to constitute a framework to address problems with the same context: ontology model, complex environment, class related structures and relationships to be identified.

The same conceptual method was successfully applied to the forensics problem of source printer attribution, where the objective is to identify which laser printer printed

a questioned document, exposing the regions on the document which contains important textures which are discriminative for a target printer.

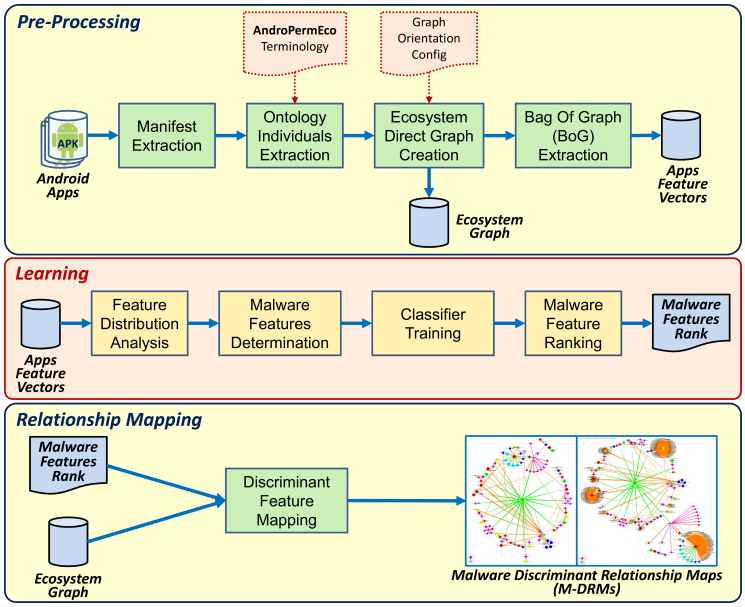


Figure 1. Proposed Method process.

2. Proposed Method

Given the problem of understanding the most important structures associated to malware on Android resources and permissions, we applied the conceptual framework using as the original model an ontology created from the data capture from application's manifests of Android smartphones ecosystem, and a Bag of Graph transformation to map ontology graph into feature vector for the Android's applications. The back modeling creates ontology graphs from the most discriminant features reducing the very large original ontology graph to a graph which is possible to plot and visually analyze.

The steps of the method are depicted on the Figure 1.

3. Experiments and Results

An experimental ecosystem has been built using a smartphone Samsung Galaxy S4 factory installation containing 181 APKs from Android v. 4.4 system and 118 partner's application. A set of 1,631 APKs of malware available in our lab repository and 860 APKs of benign applications were downloaded from Google Play Store (only APKs with more than 500,000 downloads were selected).

Over 2,790 APK files generated a list of 180,192 lines in N-Triple file format [Beckett(W3C) 2014] being 102 terminology lines, and 180,090 lines of axioms, which were then transformed into a directed graph with 39,928 nodes and 76,067 edges. For consistency and correctness when validating the list with ontology, the N-Triple lines were imported in Protege software [Stanford-BMIR 2017] with no errors.

A Bag of Graph with 41,255 words of graph was extracted from the ontology graph, creating a feature matrix of 2,587 rows (apps) by 41,255 columns (words of graph).

3.1. Datasets on Experiments

The complete dataset of 2,790 apps (vectors) was divided randomly selecting rows to compose partitions according to Table 1. Partition 0 was separated for final tests, and the other 5 partitions were combined in 5 permutations (experiments) of 4 partitions of training and 1 partition for validation.

Partition Number	Malware Apps	Benign Apps	Total Records
0	184	94	278
1	290	177	467
2	289	177	466
3	289	177	466
4	290	177	467
5	289	176	465

Table 1. Partitions of feature vectors for the experiments.

Tables 2 and 3 exhibit validation and final test results on each of 5 experiments for the 51 most discriminant malware features. The low standard deviation on all metrics with the five combinations of partitions, using different datasets on validation and the same separated dataset for the final test, indicates robustness and generalization (reduced overfitting) of classifiers.

Exp. Num.	AUC %	Accuracy %	Recall %	Precision %	F1Score %
1	79.40	73.98	82.41	79.67	81.02
2	80.32	71.29	76.47	78.65	77.54
3	80.76	74.05	75.78	81.72	78.64
4	79.83	72.47	78.28	79.37	78.82
5	79.06	72.76	81.31	78.86	80.07
Mean	79.87	72.91	78.85	79.65	79.22
σ	0.61	1.03	2.62	1.09	1.20

Table 2. ROC Curve AUC (Area Under Curve) of the trained classifier and Validation test results for 51 most discriminant features.

Traversing the entire graph of the ecosystem selecting only nodes and properties that fits on the constructions of the words of graph associated with the 51 most discriminant features we found a small portion of the entire ecosystem graph, and looking from the initial nodes of malware and stopping when a node from the most discriminant features of the ranked 51 is achieve resulted on the Figure 2.

It is interesting to see the permissions where a high percentage of malwares are connected (just the most ranked one). The most used is android_pemission_SEND_SMS, and a impressive number is connected to android_pemission_VIBRATE.

Exp. Num.	Accuracy %	Recall %	Precision %	F1Score %
1	70.24	77.72	80.34	79.01
2	70.76	76.63	81.03	78.77
3	72.63	77.17	82.56	79.78
4	69.73	79.89	79.46	79.67
5	73.18	79.35	82.49	80.89
Mean	71.31	78.15	81.18	79.62
σ	1.36	1.26	1.21	0.74

Table 3. Final test results for 51 most discriminant features using the trained classifier on each experiment.

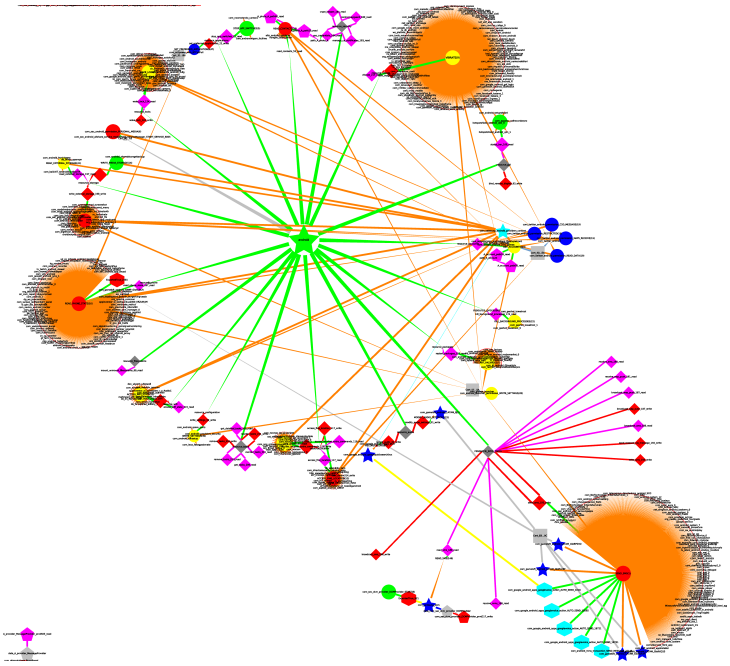


Figure 2. Android most important permissions ontology exploiting surface with malwares attached. Picture shows only the most important attachment for each malware. Zoom in for details on PDF.

4. Conclusion and Future Work

In this paper, we have introduced two new methods to address the problem of mapping the relationships and characteristics of malicious software in smartphones. We provided an extensible framework for mapping the analyzed elements in the Android system using ontologies, as well as a random forest-based method for automatically extracting meaningful information from the ontological map obtained from the new mapping algorithm. The experimental results in the Android ecosystem considered showed that the methodology proposed in this article was able to detect 51 important elements relative to malware activity, a remarkable result for security applications.

References

Beckett(W3C), D. (2014). Rdf 1.1 n-triples. Technical report, World Wide Web Consortium (W3C).

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Chanajitt, R., Viriyasitavat, W., and Choo, K.-K. R. (2016). Forensic analysis and security assessment of android m-banking apps. *Australian Journal of Forensic Sciences*, 0(0):1–17.

Criminisi, A., Shotton, J., and Konukoglu, E. (2012). Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends® in Computer Graphics and Vision*, 7(2-3):81–227.

Das, S., Liu, Y., Zhang, W., and Chandramohan, M. (2016). Semantics-based online malware detection: Towards efficient real-time protection against malware. *IEEE Transactions on Information Forensics and Security*, 11(2):289–302.

Enck, W., Gilbert, P., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P., and Sheth, A. N. (2010). Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 393–407, Berkeley, CA, USA. USENIX Association.

Enck, W., Ongtang, M., and McDaniel, P. (2009). Understanding android security. *IEEE Security Privacy*, 7(1):50–57.

Felt, A. P., Chin, E., Hanna, S., Song, D., and Wagner, D. (2011). Android permissions demystified. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 627–638, New York, NY, USA. ACM.

Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. (2012). Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 3:1–3:14, New York, NY, USA. ACM.

Gruber, T. (2009). Ontology. In *Encyclopedia of Database Systems*, page 0. Springer-Verlag, US.

Guarino, N., Oberle, D., and Staab, S. (2009). What is an ontology? In *Handbook on Ontologies*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg.

Murphy, K. P. (2012). *Adaptive basis function models*, chapter 16, pages 543–587. Adaptive computation and machine learning. The MIT Press.

Silva, F. B., Tabbone, S., and d. S. Torres, R. (2014). Bog: A new approach for graph matching. In *2014 22nd International Conference on Pattern Recognition*, pages 82–87, Stockholm, Sweden. ICPR-2014.

Stanford-BMIR (2017). Protege software. Accessed: 2017-06-19.