

# SpringerBriefs in Computer Science

## Series editors

Stan Zdonik, Brown University, Providence, USA  
Shashi Shekhar, University of Minnesota, Minneapolis, USA  
Xindong Wu, University of Vermont, Burlington, USA  
Lakhmi C. Jain, University of South Australia, Adelaide, Australia  
David Padua, University of Illinois Urbana-Champaign, Urbana, USA  
Xuemin Sherman Shen, University of Waterloo, Waterloo, Canada  
Borko Furht, Florida Atlantic University, Boca Raton, USA  
V.S. Subrahmanian, University of Maryland, College Park, USA  
Martial Hebert, Carnegie Mellon University, Pittsburgh, USA  
Katsushi Ikeuchi, University of Tokyo, Tokyo, Japan  
Bruno Siciliano, Università di Napoli Federico II, Napoli, Italy  
Sushil Jajodia, George Mason University, Fairfax, USA  
Newton Lee, Newton Lee Laboratories, LLC, Tujunga, USA

More information about this series at <http://www.springer.com/series/10028>

Denise Demirel • Lucas Schabhüser  
Johannes Buchmann

# Privately and Publicly Verifiable Computing Techniques

A Survey

 Springer

Denise Demirel  
Theoretische Informatik  
Technische Universität Darmstadt  
Darmstadt, Hessen, Germany

Lucas Schabhüser  
Theoretische Informatik  
Technische Universität Darmstadt  
Darmstadt, Hessen, Germany

Johannes Buchmann  
Theoretische Informatik  
Technische Universität Darmstadt  
Darmstadt, Hessen, Germany

ISSN 2191-5768 ISSN 2191-5776 (electronic)  
SpringerBriefs in Computer Science  
ISBN 978-3-319-53797-9 ISBN 978-3-319-53798-6 (eBook)  
DOI 10.1007/978-3-319-53798-6

Library of Congress Control Number: 2017931973

© The Author(s) 2017

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature  
The registered company is Springer International Publishing AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

Verifiable computing refers to methods that allow delegating the computation of a function on outsourced data to a server, such that the data owner and/or third parties can verify that the result has been computed correctly. Those approaches are even more useful when they provide a verification process that is more efficient than performing the computation locally. To address this challenge, many techniques have been proposed. In this work, the first comprehensive survey of all existing constructions is provided. In doing so, we are concerned with a setting where three parties are involved: A *client* who provides some input data, a *server* who evaluates a function on the input data, and a *verifier* who verifies the correctness of the result. Schemes dealing with a more complicated setting of multiple clients, verifiers, or servers are beyond the scope of this work. Furthermore, we do not consider approaches that rely on replication, trusted hardware, remote attestation, or spot checking.

For all approaches that match our setting and allow for a sufficiently efficient verification process, we provide a brief description of the approach and highlight the properties the solution achieve. More precisely, we analyze which level of security it provides, how efficient the verification process is, whether anyone or only the client can check the correctness of the result, which function class the verifiable computing scheme supports, and whether privacy with respect to the input and/or output data is given. Based on this analysis we compare the different approaches and outline possible directions for future work.

Darmstadt, Germany  
June 2016

Denise Demirel  
Lucas Schabhüser  
Johannes Buchmann

# Acknowledgments

This work has been co-funded by the European Union’s Horizon 2020 research and innovation program under Grant Agreement No 644962. In addition, it has received funding from the DFG as part of project “Long-Term Secure Archiving” within the CRC 1119 CROSSING. The authors wish to thank Daniel Slamanig and David Derler for their valuable input and helpful discussion during the writing of this work.

# Contents

<b>1 Introduction</b> .....	1
1.1 Motivation .....	1
1.2 Roadmap.....	2
1.3 Organisation .....	3
References .....	3
<b>2 Preliminaries</b> .....	5
2.1 Verifiable Computation .....	5
2.2 Properties of Verifiable Computing Schemes.....	6
2.2.1 Security.....	7
2.2.2 Privacy.....	8
2.2.3 Efficiency.....	10
References .....	10
<b>3 Proof and Argument Based Verifiable Computing</b> .....	13
3.1 Introduction to Proof and Argument Based Approaches .....	13
3.2 Interactive Proof Based Approaches .....	14
3.2.1 Verifiable Computation with Massively Parallel Interactive Proofs .....	15
3.2.2 Allspice: A Hybrid Architecture for Interactive Verifiable Computation .....	15
3.3 Interactive Argument Based Approaches .....	16
3.3.1 Pepper: Making Argument Systems for Outsourced Computation Practical (Sometimes).....	17
3.3.2 Ginger: Taking Proof-Based Verified Computation a Few Steps Closer to Practicality .....	17
3.3.3 Zaatar: Resolving the Conflict Between Generality and Plausibility in Verified Computation.....	17
3.3.4 Pantry: Verifying Computations with State.....	17
3.3.5 River: Verifiable Computation with Reduced Informational Costs and Computational Costs .....	18

3.4	Non-interactive Argument Based Approaches.....	18
3.4.1	Pinocchio: Nearly Practical Verifiable Computation.....	19
3.4.2	Geppetto: Versatile Verifiable Computation .....	19
3.4.3	SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge .....	20
3.4.4	Succinct Non-interactive Zero Knowledge for a von Neumann Architecture .....	20
3.4.5	Buffet: Efficient RAM and Control Flow in Verifiable Outsourced Computation .....	20
3.4.6	ADSNARK: Nearly Practical and Privacy-Preserving Proofs on Authenticated Data .....	20
3.4.7	Block Programs: Improving Efficiency of Verifiable Computation for Circuits with Repeated Substructures.....	21
	References .....	21
<b>4</b>	<b>Verifiable Computing from Fully Homomorphic Encryption.....</b>	<b>23</b>
4.1	Definitions for Fully Homomorphic Encryption .....	23
4.2	Verifiable Computing Schemes Based on FHE.....	24
4.2.1	Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers.....	24
4.2.2	Improved Delegation of Computation Using Fully Homomorphic Encryption.....	25
	References .....	25
<b>5</b>	<b>Homomorphic Authenticators .....</b>	<b>27</b>
5.1	Definitions for Homomorphic Authenticators .....	27
5.2	Verifiable Computing Schemes Based on MACs .....	30
5.2.1	Verifiable Delegation of Computation on Outsourced Data .....	30
5.2.2	Generalized Homomorphic MACs with Efficient Verification .....	30
5.2.3	Efficiently Verifiable Computation on Encrypted Data .....	31
5.3	Signature Based Verifiable Computing on Linear Functions .....	31
5.3.1	Programmable Hash Functions Go Private: Constructions and Applications to (Homomorphic) Signatures with Shorter Public Keys .....	31
5.4	Signature Based Verifiable Computing for Polynomial Functions ....	32
5.4.1	Homomorphic Signatures with Efficient Verification for Polynomial Functions .....	32
5.4.2	Algebraic (Trapdoor) One-Way Functions and Their Applications .....	32
5.5	Signature Based Verifiable Computing Using Homomorphic Encryption .....	33
	References .....	34

- 6 Verifiable Computing Frameworks from Functional Encryption and Functional Signatures** ..... 37
  - 6.1 Verifiable Computation from Functional Encryption ..... 37
    - 6.1.1 Verifiable Computation from Attribute Based Encryption ..... 38
    - 6.1.2 Delegatable Homomorphic Encryption with Applications to Secure Outsourcing of Computation ..... 38
  - 6.2 Verifiable Computation from Functional Signatures ..... 39
    - 6.2.1 Functional Signatures and Pseudorandom Functions ..... 39
  - References ..... 41
- 7 Verifiable Computing for Specific Applications** ..... 43
  - 7.1 From Secrecy to Soundness: Efficient Verification via Secure Computation ..... 43
  - 7.2 Signatures of Correct Computation ..... 44
  - 7.3 Efficient Techniques for Publicly Verifiable Delegation of Computation ..... 44
  - 7.4 Efficient Computation Outsourcing for Inverting a Class of Homomorphic Functions ..... 45
  - 7.5 Secure Delegation of Elliptic-Curve Pairing ..... 45
  - 7.6 Efficiently Verifiable Computation on Encrypted Data ..... 46
  - 7.7 Verifiable Delegation of Computation over Large Datasets ..... 46
  - 7.8 Batch Verifiable Computation with Public Verifiability for Outsourcing Polynomials and Matrix Computations ..... 46
  - 7.9 TrueSet: Nearly Practical Verifiable Set Computations ..... 46
  - References ..... 47
- 8 Analysis of the State of the Art** ..... 49
  - 8.1 Security, Privacy, and Efficiency ..... 49
  - 8.2 Long-Term Privacy ..... 53
  - 8.3 Implementations ..... 54
  - References ..... 54
- 9 Conclusion** ..... 57
  - References ..... 58
- A Assumptions** ..... 59
  - References ..... 62