Compiler Design in Cryptography

LITERATURE REVIEW
REPORT
SUBMITTED TO

M S RAMAIAH INSTITUTE OF TECHNOLOGY
(Autonomous Institute, Affiliated to VTU)

SUBMITTED BY

Name	USN
Avnish Pathak	**1MS20CS147**
Jay Jariwala	**1MS20CS053**
Harsh Dutta Tewari	**1MS20CS050**
Prateek Singh	**1MS20CS084**

As part of the Course **Compiler Design-CS61**

SUPERVISED BY
Faculty
Dr. Parkavi.A

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

M S RAMAIAH INSTITUTE OF TECHNOLOGY

Apr,2023

Department of Computer Science and Engineering

M S Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)

Bangalore – 54

CERTIFICATE

This is to certify that Avnish Pathak(1MS20CS147), Jay Jariwala(1MS20CS053), Harsh Dutta Tewari(1MS20CS050), Prateek Singh(1MS20CS084) have completed the **"Compiler Design in Cryptography"** as part of Literature review. I declare that the entire content embodied in this B.E CSE, 6th semester report contents are not copied.

Submitted by

Name	USN
Avnish Pathak	1MS20CS147
Jay Jariwala	1MS20CS053
Harsh Tewari	1MS20CS050
Prateek Singh	1MS20CS084

Guided by

Dr. Parkavi.A
(Dept of CSE, RIT)

(Assistant Professor, Dept. of CSE, RIT)

Department of Computer Science and Engineering

M S Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)

Bangalore – 54

Evaluation Sheet

USN	Name	Literature survey and Explanation Skills (5)	Documentation & Plagiarism checkup (5)	Total Marks (10)
1MS20CS147	Avnish Pathak			
1MS20CS053	Jay Jariwala			
1MS20CS050	Harsh Tewari			
1MS20CS084	Prateek Singh			

Evaluated By

Name: Dr. Parkavi.A
Designation: Associate Professor
Department: Computer Science & Engineering, RIT
Signature:

Table of Contents

Abstract

Although cryptography and compiler design appear to be different subjects, they really have a lot in common. While compiler design focuses on converting source code into machine code, cryptography is concerned with protecting the privacy of data. Compilers, however, can be extremely important in maintaining the safety of cryptographic systems. The uses of compiler design in cryptography will be discussed in this paper. This paper gives an overview of how compiler design is used in cryptography.

Cryptographic systems may be made more secure by using compilers. The code may be subjected to security evaluations by compilers to find any possible flaws, including buffer overflows or integer overflows. In order to thwart assaults, they can also integrate security measures like address space layout randomization (ASLR) and stack canaries.

Consequently, compiler design is crucial to cryptography. It may be used to create cryptographic protocols, enhance the security of cryptographic systems, and optimize cryptographic algorithms. The function of compilers in assuring cryptography's security will become ever more crucial as the discipline continues to develop.

Introduction

A crucial component of contemporary encryption is compiler design. The science of cryptography, which is used to provide secure communications and transactions, significantly depends on the adoption of reliable cryptographic algorithms and protocols. By allowing these algorithms and protocols' effective implementation on diverse computer platforms, compilers play a crucial role in maintaining the security of these systems.

Cryptography is impacted by compiler design because optimized cryptographic algorithms use less resources and are more effective. Because to this improvement, cryptographic systems may work more quickly and securely, which reduces their susceptibility to assaults. The implementation of cryptographic protocols, which have a range of various components, is another significant use of compiler design in cryptography. A large portion of the implementation procedure may be automated by compilers, lowering the risk of human mistake and guaranteeing that the protocol is implemented properly.

By doing security evaluations and adding security features like stack canaries and ASLR, compilers can be utilized to increase the security of cryptographic systems. They are also necessary for developing secure software systems, including digital signatures, electronic banking, and secure communications. In order to create these applications and guarantee their efficiency and security, compilers are required.

Finally, compiler design is crucial in the area of cryptography. It is essential to the enhancement of the security of cryptographic systems, the implementation of cryptographic protocols, the optimization of cryptographic algorithms, and the development of safe software systems. The importance of compilers in assuring the security of these systems will increase as the area of cryptography develops.

Literature Study

Sl No.	Title	Author/Year	Summary	Result/Remark
1.	Compiler-Based Techniques to Secure Cryptographic Embedded Software Against Side-Channel Attacks	Giovanni Agosta, Alessandro Barenghi, Gerardo Pelosi (2016)	The authors suggested a compiler-assisted masking method that transforms the original code to get rid of secret-dependent control and data flows, producing constant-time code. The method was tested against a number of cryptographic algorithms, and it was discovered to offer a high level of protection against side-channel attacks with no performance effect.	The proposed technique can effectively secure cryptographic embedded software against side-channel attacks with minimal performance overhead.
2.	Alchemy: A Language and Compiler for Homomorphic Encryption Made Easy	Eric Crockett, Chris Peikert, Chad Sharp (2018)	The paper introduces a novel programming language, Alchemy, that aims to simplify the development of homomorphic encryption applications. The authors emphasize that Alchemy provides a user-friendly programming interface, enabling developers to create such applications without needing detailed knowledge of the underlying encryption scheme. They also present a compiler that translates Alchemy programs into native	The paper highlights the potential of Alchemy to advance the field of homomorphic encryption and provides valuable insights into the development of homomorphic encryption applications. Currently, there are the current limitations of homomorphic encryption techniques, such as computational inefficiency, and to

			code to facilitate efficient execution. The paper provides practical examples of the Alchemy language, including encrypted searching, secure computation of statistical functions, and secure data analysis. The authors also discuss the current limitations of homomorphic encryption techniques and propose future research directions to overcome these limitations.	overcome these future research directions are suggested by authors to overcome these limitations.
3.	CVS:A compiler for the analysis of Cryptographic Protocols	Antonio Durante,Riccardo Focard,Roberto Gorrieri	The research paper describes a compiler called CVS that is capable of analyzing cryptographic protocols. It can automatically detect vulnerabilities and verify the security properties of protocols. The compiler is designed to be efficient and scalable, and it uses a combination of static and dynamic analysis techniques. The paper demonstrates the effectiveness of the CVS compiler by analyzing several popular cryptographic protocols.	CVS is a compiler that can identify different kinds of attacks by analyzing cryptographic protocols. Yet, the accuracy of the protocol model and the excellence of the verification circumstances determine how effective it is. Although this is the case, it is still a strong tool that has been used to examine numerous protocols.

4	cPLC – A Cryptographic Programming Language and Compiler	Endre Bangerter, Stephan Krenn Martial Seifriz,Ulric UltesNitshe(2018)	The cPLC language is designed to support a wide range of cryptographic operations, including encryption, decryption, hashing, digital signatures, and key management. It includes features such as constant-time execution, automatic memory management, and built-in protection against side-channel attacks. The cPLC compiler is designed to generate efficient and secure code for various hardware platforms, including microcontrollers, embedded systems, and desktop computers. It supports a range of optimization techniques, such as loop unrolling, constant folding, and instruction scheduling, to improve code performance. One of the key features of cPLC is its support for constant-time execution, which is critical for cryptographic operations. Constant-time execution ensures that the execution time of an algorithm is independent of the input data.	The paper concludes that cPLC is a promising approach to developing secure and efficient cryptographic programs. It can provide significant improvements in performance and security compared to traditional programming languages and compilers. However, the paper also acknowledges that further research is needed to evaluate the scalability and applicability of cPLC to real-world applications.

Conclusion

Since compilers translate human-readable programming languages into machine-readable code that computers can execute, they are a crucial component of encryption. Programming languages like C, C++, and Java are used to construct cryptographic protocols and algorithms, including symmetric and asymmetric encryption, digital signatures, and secure hash functions. It is crucial to make sure that the compiler produces safe code that can withstand attacks from hackers and other bad actors.

To avoid unauthorized access to the software, compiler designers must integrate security protections including data encryption, code obfuscation, and other strategies. Additionally, they must make sure that the created code is resistant to well-known attacks and take into account the security implications of the programming language used to develop the software.

Performance is a crucial component of compiler design in cryptography. Because cryptographic methods can be computationally demanding, ineffective code execution can have a major negative influence on the software's performance. To guarantee that the programmer operates effectively while ensuring security, compiler designers must optimize the code that the compiler generates. This involves lowering memory use, minimizing the amount of CPU instructions that are performed, and using other speed optimization approaches.

In conclusion, for safe cryptographic software to function, compiler design is crucial since it guarantees that the resulting code is both efficient and secure. Compiler designers may produce software that is resistant to assaults and offers top performance by integrating security features and optimizing code execution.

References

1. **https://www.geeksforgeeks.org/compiler-design/**
2. **https://symmetriccybersecurity.com/cryptography-basics/**
3. **https://shantanusharma.net/2019/05/28/compiler-design-for-security/**
4. **https://people.eecs.berkeley.edu/~daw/papers/secure_compiler_design.pdf**
5. G. Agosta, A. Barenghi and G. Pelosi, "Compiler-Based Techniques to Secure Cryptographic Embedded Software Against Side-Channel Attacks," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*
6. Eric Crockett, Chris Peikert and Chad Sharp, "Alchemy: A Language and Compiler for Homomorphic Encryption Made Easy" in *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS).*
7. M. Burrows, M. Abadi, and R. Needham. "*A Logic of Authentication". Proceedings of the Royal Society of London, 426:233–271, 1989*.
8. M. Seifriz, "CPLc: A Generic Compiler for Fast Prototyping Crypto-graphic Protocols," Master's thesis, University of Fribourg, 2010.

Proof of Literature Survey

Paper-1

G. Agosta, A. Barenghi and G. Pelosi, "Compiler-Based Techniques to Secure Cryptographic Embedded Software Against Side-Channel Attacks," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 8, pp. 1550-1554, Aug. 2020, doi: 10.1109/TCAD.2019.2912924.

Abstract: Side-channel assaults provide a real and present danger to the safety of computer systems, from small devices to high performance platforms. The methods used to analyze a side-channel vulnerability in an implementation or automatically construct countermeasures, which rely on techniques common to compiler systems, are briefly systematized in this work. The use of dynamic compilation techniques to stop a side-channel attacker from creating a model of the attacked programmer is a significant advancement in countermeasures approaches that will receive special attention. We highlight possible future research directions in this area as we draw to a close.

URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8695829&isnumber=9142465

Paper-2

Abstract:

Eric Crockett, Chris Peikert and Chad Sharp, "Alchemy: A Language and Compiler for Homomorphic Encryption Made Easy," in *CCS '18: 2018 ACM SIGSAC Conference on Computer & Communications Security Oct. 15–19, 2018, Toronto, ON, Canada. ACM, New York, NY, USA,* doi:10.1145/3243734.3243828.

Abstract: The paper presents Alchemy, a programming language designed to simplify the development of homomorphic encryption applications. The language provides a user-friendly interface for developers, eliminating the need for detailed knowledge of the encryption scheme. The authors also introduce a compiler that translates Alchemy programs into native code for efficient execution. The paper includes practical examples of Alchemy's use, such as encrypted searching and secure computation of statistical functions. While discussing the current limitations of homomorphic encryption techniques, the authors suggest future research directions to overcome these limitations. The paper highlights the potential of Alchemy to advance the field and provides valuable insights into the development of homomorphic encryption applications..

URL: https://dl.acm.org/doi/10.1145/3243734.3243787

Paper-3

Antonio Durante1, Riccardo Focardi, Roberto Gorrieri, “CVS: A Compiler for the Analysis of Cryptographic Protocols,” in *Proceedings of the 12th IEEE Computer Security Foundations Workshop,*doi: 10.1109/CSFW.1999.779774

Abstract: A compiler called CVS is made specifically for analyzing cryptographic protocols. It uses a protocol description as input to create a finite state machine model, which is then used to automatically check the protocol's security features. The verification procedure entails creating a list of verification requirements that the protocol must meet in order to be secure. The compiler will produce a counterexample demonstrating how the protocol can be attacked if any of these conditions are not met. TLS, SSH, and Kerberos are just a few of the protocols that have been analyzed using CVS. It has been demonstrated to be effective in identifying a number of attacks and vulnerabilities, including key compromise attacks, replay attacks, and man-in-the-middle attacks. But, like with any instrument, the efficiency

Paper-4

M. Seifriz, “CPLc: A Generic Compiler for Fast Prototyping Crypto-graphic Protocols,” Master’s thesis, University of Fribourg, 2018

Abstract—Cryptographic two-party protocols are used ubiqui-tously in everyday life. While some of these protocols are easy to understand and implement (e.g., key exchange or transmission of encrypted data), many of them are much more complex (e.g., e-banking and e-voting applications, or anonymous authentication and credential systems). For a software engineer without appropriate cryptographic skills the implementation of such protocols is often difficult,time consuming and error-prone. For this reason, a number of compilers supporting programmers have been published in recent years. However, they are either designed for very specific cryptographic primitives (e.g., zero-knowledge proofs of knowl-edge), or they only offer a very low level of abstraction and thus again demand substantial mathematical and cryptographic skills from the programmer.

URL:https://www.researchgate.net/publication/224259131_cPLC_-_A_cryptographic_programming_language_and_compiler