Exposé of a bachelor thesis

A Modular Approach to Efficiently Generating Card-Based Cryptographic Protocols

Anne Hoff
Advisor: Michael Kirsten

Oktober 2022

1 Introduction

This thesis aims to contribute to the field of card-based cryptography. This field of research provides simple protocols, where a deck of playing cards can be used to perform complex and secure multiparty calculations (see subsection 3.1). This is done by encoding the input as a sequence of cards and then performing a predefined set of turns and shuffles on the cards until the desired result is reached.
Several different protocols have already been proposed, most famously the so-called 'Five Card Trick' (Boer, 1990), that calculates the logical AND from two input bits using five playing cards.
These protocols and their properties are often found and proven by hand. This might be manageable for small problems, like the logical AND, but becomes increasingly complex with bigger problems.
To automate the creation of protocols, Koch, Schrempp, and Kirsten (2021) have introduced formal verification to the field. They used the technique of software bounded model checking (SBMC) to find new protocols to compute a logical AND.
SBMC iterates through possible turn and shuffle operations until it finds a run which produces the desired output (see subsection 3.2). The challenge of using this technique is that the runtime of the programs grows for more realistic protocols with more input variables and more steps than simple boolean operators. Thus, we need a more refined approach that allows for the use of formal verification for finding protocols for more practical and realistic protocols.
The goal of this thesis attempts to address the problem by creating a modular method of finding card protocols for practical problems. This modular approach would take protocols for easier problems (like the before mentioned

logical AND) as a possible step (apart from turns and shuffles) within a verification program for a more complex protocol. We could, for example, take any functionally complete set of boolean operators and find protocols for them using SBMC. We could then, in programs for protocols for more practical problems, perform one of these boolean operations as a step because we already know how to perform such a protocol. We can then even take these protocols for more practical problems as modules in other problems.
One example of a complicated and realistic problem that this thesis takes a look at is the protocol given by Mizuki, Asiedu, and Sone (2013) (see subsection 3.3).
With our approach, we hope to create more runtime efficient programs that find protocols. It will also become apparent how the found protocols compare regarding the number of cards and steps that are necessary.

2 Research Questions

This thesis thus poses the research question:

Can we create a modular formal verification method for finding protocols for practical card-based cryptographic problems, and is this approach more runtime efficient than the naive approach? How do protocols found by this method compare to those found by the naive approach regarding the number of cards and operations?

More precisely we want to answer the following questions:

- Q1 How do protocols for different functionally complete sets of boolean operators compare regarding the number of cards and operations for their protocols?
- Q2 Does using boolean operators as modules have a higher runtime efficiency than trying to directly find a protocol to calculate a more complex function? Do these protocols found by using boolean operators as modules use more cards and need more steps?
- Q3 Can we find protocols using fewer cards and steps more quickly using the modular verification method instead of finding protocols 'by hand'?
- Q4 Can we use previously found protocols as steps within our bounded model checking program to find protocols for more practical problems?

3 Foundations

3.1 Card-based Cryptography

A deck D is a multiset over a symbol set or deck alphabet. In this thesis, we focus on the two-color deck $\{\clubsuit\heartsuit\}$.

A card can either be face-up (the symbol of the card is showing) or face-down (the symbol of the card is hidden). We also define the special symbol '?', which symbolizes a card that is face-down. Two cards together form an encoding of a bit as follows: ♣♡ encodes a 0 and ♡♣ encodes a 1.
A commitment to x is a pair of two face-down cards that encode the value x.
To compute with cards, a protocol takes a set of commitments as inputs. It then performs several actions on them, for example, shuffling or turning the cards. In the end, an output is produced.
With card-based protocols, we differentiate between three types of security (Koch, Schrempp, and Kirsten, 2021). Probabilistic security, input possibilistic security and output possibilistic security. Probabilistic security means that exact possibilities are given for a sequence, while possibilistic security only states if an input/output is possible. We distinguish input possibilistic security, which states whether the sequence can belong to a specific input, and output possibilistic security, which determines to which output a sequence can belong.
We will also only be considering uniform closed shuffles. Closedness means that the set of possible permutations is invariant under repetition, while uniform possibility expresses that every possible permutation has the same probability when doing a shuffle (Koch, 2019).

3.2 CBMC

We use software bounded model checking, more precisely CBMC, for finding protocols. Given a program and a property, software bounded model checking verifies automatically if the program satisfies the property or not.
This is done symbolically by bounding the length of the runs through the program and translating it into a formula in a decidable logic, for example, a SAT problem. The SAT solver then checks whether a run that doesn't satisfy the property exists. In that case, a counterexample is produced and output. Otherwise, the property is either satisfied or an invalid run is out of bounds.
For finding card-protocols, the input states are all the possible input commitments of the players (for 2 players, there are 4 different input possibilities). The program is then executed as follows: It performs a loop, in which an operation (e.g. a turn or shuffle operation) is executed. The loop ends if the bound (the length of the protocol to be found) is reached, and it is checked if the final state is a valid computation. If it is, the found protocol (i.e. the call trace) is presented to the user.

3.3 Example of a realistic problem: Voting with a Logarithmic Number of Cards

A vote between two alternatives A_0, A_1 can be realized with $2\lceil logn \rceil + 8$ cards (Mizuki, Asiedu, and Sone, 2013).
For that, each voter P_0 to $P_i (i \in \mathbb{N})$ inputs a commitment of either ♣♡$\widehat{=}$0 if he wants to vote for A_0 or ♡♣$\widehat{=}$1 if he wants to vote for A_1.
Then the sum off all the votes is calculated. This is done by successively adding

the value of the next vote P_j to the current interim result of $P_0 + P_0 + ... + P_{j-1}$ using a half adder.
A half adder adds two binary digits b_1 and b_2 and outputs the sum bit S and the carry C bit. There are multiple ways to implement a half adder with boolean operators. It can, for example, be calculated using an AND and a XOR as $S = b_1 \oplus b_2$ and $C = b_1 \wedge b_2$.
For the voting protocol, alternative A_1 wins if the end result is $> \frac{i}{2}$ and alternative A_0 wins if the end result is $< \frac{i}{2}$. If the end result equals $\frac{i}{2}$ then the vote is tied.

4 Tasks

T1 **Find and compare protocols for different functionally complete sets of boolean operators**

T1.1 **Conduct literature analysis of what bounds and protocols already exist for each of the examined boolean operators**
Details: Collect existing protocols for boolean operators and COPY (duplicates a commitment) from literature.
Main Task
Questions: (Q1)
Artefacts: List of existing protocols for boolean operators (including source, #steps, security, types of used shuffles, ...)

T1.2 **Adapt the CBMC program of Koch, Schrempp, and Kirsten (2021) to other boolean operators (XOR, OR, COPY)**
Details: Koch, Schrempp, and Kirsten (2021) have developed a program for finding protocols for the AND function. This existing program shall be adapted to verify other boolean operators.
Functionally complete sets of operators we will have a look at are: AND, XOR and AND, OR, NOT (AND, NOT and OR, NOT are functionally complete by themselves)
Thus we have to write programs to find protocols for XOR and OR (as NOT is trivial), as well as COPY. The found protocols will have to be brought into readable form and checked for their security.
The list of existing protocols from [T1.1] can be used as a starting point for the number of cards and the amount of steps, when trying to find protocols with the program.
Main Task
Questions: (Q1)
Artefacts: programs for XOR, OR and COPY, protocols for XOR, OR and COPY (including #steps, security, types of used shuffles, ...)

T1.3 **Adapt the CBMC program of Koch, Schrempp, and Kirsten (2021) to even more boolean operators (NAND, NOR)**
Details: Further functionally complete sets of operators we could

have a look at are NAND and NOR
Thus we would have to write programs to find protocols for NAND and NOR.
The found protocols will have to be brought into readable form and checked for their security.
The list of existing protocols from [T1.1] can be used as a starting point for the number of cards and the amount of steps, when trying to find protocols with the program.
Optional Task
Questions: (Q1)
Artefacts: programs for NAND and NOR, protocols for NAND and NOR (including #steps, security, types of used shuffles, ...)

T2 **Compare a program which uses only turns and shuffles to programs using boolean operators as modules for finding protocols (for half adders)**

T2.1 **Adapt the CBMC program of Koch, Schrempp, and Kirsten (2021) to find protocols for half adders**
Details: Koch, Schrempp, and Kirsten (2021) have developed a program for finding protocols for the AND function. This existing program shall be adapted to verify half adders. Thus the result configuration should display the sum and the carry bit.
Main Task
Questions: (Q2)
Artefacts: program and protocol for the half adder

T2.2 **Write a CBMC program that has the boolean operators AND and XOR as modules and finds protocols for half adders**
Details: Now, a program shall be written, that does not just perform turn and shuffle operations, but can also perform predefined boolean operations for which we have already found protocols in [T1]. This means, that the methods for the operations have to be designed and implemented for each boolean operator within the functionally complete set of operators. In this case AND and XOR.
Main Task
Questions: (Q2), (Q4)
Artefacts: program and protocol for the half adder that uses the boolean operations AND and XOR, implementation of the boolean operations AND and XOR as modules

T2.3 **Write a CBMC program that has other boolean operators as modules and finds protocols for half adders**
Details: Similarly to [T2.1], we can write a program that uses the boolean operators of other functionally complete set of operators that were examined in [T1]. Namely AND, OR and NOT from [T1.2] and NAND and NOR from [T1.3].

Optional Task
Questions: (Q2), (Q4)
Artefacts: program and protocol for the half adder that uses other boolean operations, implementation of the other boolean operations as modules

T3 **Apply the method of using modules for finding protocols to the the Two-Candidate Voting Problem, that uses a logarithmic number of cards**

T3.1 **Description of all the steps in the Two-Candidate Voting Protocol by Mizuki, Asiedu, and Sone (2013)**
Details: Mizuki, Asiedu, and Sone (2013) proposed a two-candidate voting protocol that uses a logarithmic number of cards. This protocol is however not sufficiently explained in their paper to reproduce it step-by-step. Therefore an in depth explanation of each performed action will be written, so potential voters could follow it like a manual.
Main Task
Questions: (Q3)
Artefacts: step-by-step explanation of how the protocol works (e.g. with a KWH tree by Koch, Walzer, and Härtel (2015))

T3.2 **Write a CBMC program that uses modules and finds protocols for the Two-Candidate Voting Problem**
Details: We will design a program, that uses different modules to find protocols for the two-candidate voting problem. Unlike the other programs, the input state will be a commitment of one voters chosen alternative and a set of commitments of the current sum of all previous votes. This will give us a protocol for adding one commitment on to a current sum of all the previous inputs. The size of set of possible input states will therefore only be $2n$ (instead of 2^n if we would look at all possible commitments for all voters). We will use previously found protocols from [T1] and [T2] as well as previously implemented modules for the boolean operators from [T2] in this program, to find a protocol for this specific problem.
Main Task
Questions: (Q3), (Q4)
Artefacts: program and protocol for the two-candidate voting problem

5 Time Schedule

Preparation	Weeks 1-2 (KW 40-41)	
Retrace work and code by Koch, Schrempp, and Kirsten (2021)	Week 1-2 (KW 40-41)	October 2022
Milestone: [T1] Finished		
[T1] Protocols for Boolean Operators	Weeks 3-5 (KW 42-44)	
[T1.1] Literature Analysis	Week 3 (KW 42)	
[T1.2] CBMC programs for XOR, OR and COPY	Weeks 3-5 (KW 42-44)	November 2022
[T1.3] CBMC programs for NAND and NOR	optional	
Milestone: [T1] Finished		
[T2] Compare Turns and Shuffles to Modules (Half Adder)	Weeks 6-8 (KW 45-47)	
[T2.1] (naive) CBMC program for the Half Adder	Weeks 6 (KW 45)	
[T2.2] CBMC program for the Half Adder using the boolean operators AND and XOR	Week 7-8 (KW 46-47)	
[T2.3] CBMC program for the Half Adder using other boolean operators	optional	
Milestone: [T2] Finished		
[T3] Apply Method to Two-Candidate Voting Problem	Weeks 9-11 (KW (48-50)	
[T3.1] Description of the Two-Candidate Voting Protocol	Week 9 (KW 48)	December 2022
[T3.2] CBMC program for Two-Candidate Voting Problem	Weeks 10-11 (KW 49-50)	
Milestone: [T3] Finished		
Writing the Thesis	Weeks 12-13 (KW 51-52)	
Write missing sections	Week 12 (KW 50)	
Proof reading and cleanup	Week 13 (KW 51)	
Milestone: Thesis Submission		
Buffer	Weeks 14-15 (KW 1-2)	January 2023

References

Boer, Bert den (1990). "More Efficient Match-Making and Satisfiability The Five Card Trick". en. In: *Advances in Cryptology — EUROCRYPT '89*. Ed. by Jean-Jacques Quisquater and Joos Vandewalle. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 208–217. ISBN: 978-3-540-46885-1. DOI: 10.1007/3-540-46885-4_23.

Koch, Alexander (2019). *Cryptographic Protocols from Physical Assumptions*. de. DOI: 10.5445/IR/1000097756. URL: https://publikationen.bibliothek.kit.edu/1000097756 (visited on 09/09/2022).

Koch, Alexander, Michael Schrempp, and Michael Kirsten (Apr. 2021). "Card-Based Cryptography Meets Formal Verification". en. In: *New Generation Computing* 39.1, pp. 115–158. ISSN: 1882-7055. DOI: 10.1007/s00354-020-00120-0. URL: https://doi.org/10.1007/s00354-020-00120-0 (visited on 09/08/2022).

Koch, Alexander, Stefan Walzer, and Kevin Härtel (2015). "Card-Based Cryptographic Protocols Using a Minimal Number of Cards". en. In: *Advances in Cryptology – ASIACRYPT 2015*. Ed. by Tetsu Iwata and Jung Hee Cheon. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 783–807. ISBN: 978-3-662-48797-6. DOI: 10.1007/978-3-662-48797-6_32.

Mizuki, Takaaki, Isaac Kobina Asiedu, and Hideaki Sone (2013). "Voting with a Logarithmic Number of Cards". en. In: *Unconventional Computation and Natural Computation*. Ed. by David Hutchison et al. Vol. 7956. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 162–173. ISBN: 978-3-642-39074-6. DOI: 10.1007/978-3-642-39074-6_16. URL: http://link.springer.com/10.1007/978-3-642-39074-6_16 (visited on 09/02/2022).