

# ProveIt—A Semi-Automatic Security Analysis Tool

(*Extended Abstract*)

Liina Kamm  
STACC,  
University of Tartu

Sven Laur  
University of Tartu

## I. INTRODUCTION

Many security proofs of cryptographic protocols are conceptually simple, but technically complex [BR04], [Sho04], [Hal05]. As such a semi-automatic prover environment would greatly simplify the design and verification phase for new protocols as well as help with proving existing protocols.

We propose a prover tool—ProveIt—that helps the user make step-by-step proofs by checking whether the specified reduction schema is applicable for the statement in question. The tool is given the formalisation of a security property through a short program also called a game. The user can apply different reduction schemas to the game that is then rewritten according to the applied schema and a new game is obtained. This kind of reduction process assures that the resulting proof is sound.

The tool does not perform automatic proofs but rather assists the user in the proof process. This kind of assistance is necessary to help make the tedious and tricky proof validation tractable. ProveIt tool also makes it easier and more intuitive to carry out cryptographic proofs, making it more accessible to engineers and students.

Our emphasis for this tool is different from existing solutions [Lau03], [BP06]. Many automatic provers convert the initial protocol into a low-level description, such as a process algebraic representation or a control-flow graph, and then perform a series of local substitutions until the final goal is reached. As a result, one loses interpretability and cannot easily guide the prover nor interpret the results in case of security flaws. Our approach is to use high-level protocol descriptions through the entire proof. Secondly, we implement modular proof techniques, which allow to split complex proofs into manageable chunks.

## II. GAME-BASED CRYPTOGRAPHIC PROOFS

We use the protocol rewriting technique [BR04], [Sho04], [Hal05] for proofs. This method is intuitive and assists the prover in finding the distance between the initial and the final cryptographic game.

In this approach, the user inputs a protocol to model a real life attack as a game  $\mathcal{G}$ . The aim is to obtain a simplified game  $\mathcal{G}^*$  where the estimation of the probability of an adversary's success is trivial. This is done by applying certain transformations to the statements of the protocol. Each reduction step changes the game according to a previously specified pattern (reduction schema). Each reduction schema

has a value connected to it. This value shows how much the adversary's advantage changes when the user applies the schema in question. When the game  $\mathcal{G}$  is reduced to game  $\mathcal{G}^*$ , the adversary's advantage can be evaluated by taking the advantage from  $\mathcal{G}^*$  and adding the advantages from each reduction step. This way a cumulative advantage is obtained.

However, when doing these steps by hand, the rewriting technique is often error-prone. Errors can emerge from the misapplication of steps due to oversight or simple typos, as the game usually has to be rewritten several times. In addition, the games can be quite lengthy, so rewriting is rather time consuming and it can be difficult to locate the necessary parts of the game.

## III. APPLICATIONS

The verification of complex protocols is one of the main applications of ProveIt. Consider, for example, multi-party computation (MPC) protocols. When designing new and more efficient protocols for the multi-party computation platform SHAREMIND [BLW08], the designers found it rather tedious and error-prone to prove the soundness of the new protocols by hand. These protocols can be very complex and, therefore, their analysis is complex and tedious. ProveIt can make these proofs tractable. The first milestone of the prover tool is for the designers to be able to verify the soundness of all the protocols used for building the SHAREMIND platform. As new and more efficient protocols are being developed for this platform, ProveIt can save the designers a lot of time, and make the proving process clearer and less tedious.

For the second milestone, the prover tool must be able to make standard security proofs for symmetric key constructions. There are few primitives in symmetric key constructions and the proofs are mostly simple. There are quite a lot of technical details, and the proofs are more diverse than in the case of MPC.

Finally, students studying to prove cryptographic protocols often find it difficult to check whether it is possible to apply a proof step to some statement. They also tend to take longer to find the right steps to apply for achieving their goal. The rewriting itself takes up quite a lot of time and doing it over and over again can make proving feel like a chore instead of the creative task it actually is. This is not a milestone in itself, but rather a by-product of the previous milestones. The tool is mainly aimed at crypto analysts and engineers, but can also be used for educational purposes.

#### IV. FEATURES

The user inputs the protocol in text form using a language that is similar to how the statements are written on paper and in  $\text{\LaTeX}$ . The language is intuitive to those who have written protocols before, and easy to learn to those who have not. The tool takes the protocol as input from the user and parses it into an abstract syntax tree. This will, however, be hidden from the user who will only see the protocol as a group of statements. The user can select a statement or statements and apply a desired reduction schema. The tool will check whether it is possible to apply the chosen schema to the statements. If it is, the transformation will be carried out and a new protocol will be generated with the results acquired during the proof step.

The tool gives a clear overview of the proof, showing the schemas that have been applied and the changes in the adversary's advantage that have occurred due to these reduction steps. It is always possible to go back to a previous version of the protocol and if the user so desires they can restart the proof from any previous protocol. It is, of course, possible to save an existing proof and continue proving later.

At the moment we have implemented rules for the free step and switching the pseudo-random permutation (PRP) family to the pseudo-random function (PRF) family. The free step is a kind of generic reduction that a user can make, if the schemas they need are not available in the tool yet. This step does not check the statement for any preconditions and it does not add a specific value into the adversary's advantage. This is done by the user instead. This step only helps the user by rewriting the game for them. If this step is used in the proof, one must keep in mind, that the soundness of the proof can no longer be guaranteed by the tool, as it has no way of checking whether the step was sound or not. On the other hand, removing this step would mean that it might not be possible to complete a proof if all the schemas necessary for the proof have not been implemented. The user is advised to use this step only when a more specific schema is not available.

The PRP/PRF switching step applies the reduction schema following from the pseudo-random permutation family definition. The advantage of the adversary is obtained by using the PRP/PRF switching lemma. This step helps the user prove properties of symmetric key constructions.

#### V. ADVANTAGES AND DRAWBACKS

The advantages of the ProveIt tool can be categorised as professional and educational. Firstly, the ability to check whether a reduction schema is applicable for the game can be considered an advantage in both categories. For crypto analysts and engineers, this functionality prevents oversight and helps to ensure that the resulting proof is sound. For students, on the other hand, this functionality is a great help in learning about the different reduction schemas and their applicability in different protocols.

Secondly, the tool helps the user with the most tedious and repetitive part of game-based proving, i.e. game rewriting. This is again relevant in both categories. For the professional,

it helps concentrate on the process of proving the protocol instead of writing the games over and over. For the student, this functionality can save a lot of time, as trying different schemas becomes an easier and less time-consuming task.

As mentioned earlier, the prover tool does not make automatic protocol proofs. On one hand, this has not been the goal of the tool from the beginning, but it is still something to consider. Automatic proving is a computationally complex problem. Even if we implement a method that tries all possible transformations on the protocol, we still need some heuristic algorithm to make these kinds of automatic proofs feasible.

#### VI. RESULTS AND FURTHER WORK

The ProveIt tool is very much a work in progress. We have implemented the protocol and the proof system. From the main part of the tool—reduction schemas—we have so far implemented the free step and switching the pseudo-random permutation family to the pseudo-random function family. As the software is modular, it is not as difficult to add subsequent schemas.

Next, we will add the schema for random masking and go on to test the design and applicability of the tool by proving the protocols given in the articles [BCK96], [BHK<sup>+</sup>99], [BLW08]. This will help us find out bottlenecks and other shortcomings of the tool. As the other target group for ProveIt is people who study how to make cryptographic proofs, we also aim to test the tool in the cryptography course in the University of Tartu. We will give the tool to students and ask them to make proofs with and without the tool to see whether the assumption that the tool is suitable also for educational purposes holds.

#### ACKNOWLEDGMENT

This work has been supported by the European Union Regional Development Fund.

#### REFERENCES

- [BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 1–15, London, UK, 1996. Springer-Verlag.
- [BHK<sup>+</sup>99] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. Umac: Fast and secure message authentication. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 216–233, London, UK, 1999. Springer-Verlag.
- [BLW08] Dan Bogdanov, Sven Laur, and Jan Willemsen. Sharemind: A Framework for Fast Privacy-Preserving Computations. In *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, volume 5283 of *LNCS*, pages 192–206. Springer, 2008.
- [BP06] Bruno Blanchet and David Pointcheval. Automated security proofs with sequences of games. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 537–554. Springer Berlin / Heidelberg, 2006.
- [BR04] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. *Cryptology ePrint Archive*, Report 2004/331, 2004. <http://eprint.iacr.org/>.
- [Hal05] Shai Halevi. A plausible approach to computer-aided cryptographic proofs. *Cryptology ePrint Archive*, Report 2005/181, 2005. <http://eprint.iacr.org/>.

- [Lau03] Peeter Laud. Handling encryption in an analysis for secure information flow. In *Proceedings of the 12th European conference on Programming*, ESOP'03, pages 159–173, Berlin, Heidelberg, 2003. Springer-Verlag.
- [Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. <http://eprint.iacr.org/>.