Combination issues

Stéphanie Delaune and Ivan Gazeau

LSV, CNRS & ENS Cachan, Université Paris Saclay, France

The results presented in this report are based on results that have been published in [3, 5]– works that have been supported by the VIP project.

Abstract. This deliverable concerns the Task 4 of the VIP project: *Modularity issues.* This report aims to sum up the results that have been obtained during the project and that are related to combination issues.

1 Introduction

Security protocols are used in many of our daily-life applications, and our privacy largely depends on their design. Since security protocols are notoriously difficult to design and analyze, formal verification techniques are important. These techniques have become mature and have been used with success to analyse several protocols. For example, a flaw has been discovered in the Single-Sign-On protocol used *e.g.* by Google Apps. It has been shown that a malicious application could very easily get access to any other application (*e.g.* Gmail or Google Calendar) of their users [1]. This flaw has been found when analysing the protocol using formal methods, abstracting messages by a term algebra and using the Avantssar validation platform [2].

However, security protocols used in practice are more and more complex and, often, formal analyzers are not strong enough to handle any kind of attacks and features of protocols. For instance, ProVerif [4] is able to deal with an infinite number of sessions but can only check a strong form of equivalence, namely diff-equivalence. APTE and AKISS are devoted to the problem of checking trace equivalence but they are limited to a finite number of sessions. Moreover, none of these tools (neither ProVerif, nor APTE and AKISS) are able to handle the exclusive or primitive which is used in many protocols.

Since there exist a variety of primitives that a protocol can used, a question that naturally arises is the following one: Can we combine for instance a procedure dedicated to the pure xor theory with another one which is able to handle another set of primtives in order to analyse a protocol that relies on both kinds of operators. This is the kind of combination results we targeted at the beginning of the project. Actually, we did not really solve this problem during the project but we develop two novel approaches that present some modularity features and are therefore suitable to analyse many interesting protocols. In particular, we have developed a method able to deal with the exclusive or operator in combination with an arbitrary theory as soon as it satisfies the generic conditions required by the tool AKISS. Our second work is about designing conditions on protocols which are sufficient to ensure anonymity and unlinkability, and which can then be effectively checked automatically using existing verification tools. Our two conditions correspond to two broad classes of attacks on unlinkability, corresponding to data and control-flow leaks. This second result is not dedicated to a specific class of primitives, and can handle a variety of cryptographic primitives.

2 Integrating XOR in AKISS

The xor primitive is an important primitive as it is used in many security protocols especially RFID protocols. Due to its algebraic properties, this operator is difficult to handle in equivalence checker: as far as we know, when the project started, no tool was able to deal with such an operator.

The AKISS tool is able to check equivalence properties for theories which have *the finite variant property.* This property implies that any term has a normal form which is not the case for the xor due to its associativity and commutativity properties. A solution is to consider equalities modulo associativity and commutativity (AC). Such a solution is correct in theory but is does not work in practice. Indeed, the AKISS procedure which normally terminates on any input, loops in this adapted process even on very simple protocols.

We show in [3] how to integrate the xor operator into AKISS. The first step was to adapt the unification procedure to be able to do unification modulo AC: we obtained a correct tool which does not terminate. The non-termination occurred during what is called the saturation phase: the algorithm saturate a set of knowledge about the process in order to find out all the relevant equalities to test. To obtain termination we put three kinds of restrictions on this saturation procedure and we add a mechanism that removes data that are redundant because of the xor equalities. Finally, we prove that the restrictions and the clean up mechanism preserve the correction of the algorithm.

Prototype. The implementation of the extended version of AKISS has been done in Ocaml under the GNU license as the original version and implied around 2000 new lines of code. This extension preserves the main code structure of the original AKISS. The source code can be found at: http://github.com/akiss/akiss. Even if we did not provide a proof of termination, the tool allows to check equivalence properties on all of our cases studies. The procedure is much longer than for checking equivalence without XOR, which was not surprising as the unification procedure itself requires the use of the Maude tool while the original tool uses a native unification. However, it is likely that additional techniques can be added to improve our benchmarks.

This result has been obtained by David Baelde, Stéphanie Delaune, Ivan Gazeau, and Steve Kremer, and is currently under submission.

3 A method well-suited for a large class of primitives

We identify a large class of 2-party protocols (simple else branches, arbitrary cryptographic primitives) and we devise two conditions that imply unlinkability and anonymity for an unbounded number of sessions. We show how these two conditions can be automatically checked using the ProVerif tool, and we provide tool support for that, namely Ukano. We have analyzed several protocols, among them the Basic Access Control (BAC) protocol as well as the Password Authenticated Connection Establishment (PACE) protocol that are both used in e-passports. We notably establish the first proof of unlinkability for the BAC protocol followed by the Passive Authentication (PA) and Active Authentication (AA) protocols. We also report on an attack that we found on the PACE protocol, and another one that we found on the LAK protocol whereas it is claimed untraceable in [6]. It happens that our conditions are rather tight: we provide an attack every time one of them is not satisfied.

We now give an intuitive overview of these two conditions. In order to do this, assume that we want to design a mutual authentication protocol between a tag T and a reader R based on symmetric encryption, and we want this protocol to be unlinkable. We note $\{m\}_k$ the symmetric encryption of a message m with a key k and we assume that k is a symmetric key shared between T and R.

A first attempt to design such a protocol is presented using Alice & Bob notation as follows (n_R is a fresh nonce):

$$\begin{array}{l} 1.\ R \to T : n_R \\ 2.\ T \to R : \{n_R\}_k \end{array}$$

This first attempt based on a challenge-response scheme is actually linkable. Indeed, an active attacker who systematically intercepts the nonce n_R and replaces it by a constant will be able to infer whether the same tag has been used in different sessions or not by comparing the answers he receives. Here, the tag is linkable because, for a certain behavior (possibly malicious) of the attacker, some relations between messages leak information about the agents that are involved in the execution. Our first condition, namely *frame opacity*, actually checks that all outputted messages have only trivial relations that can therefore not be exploited by the attacker.

Our second attempt takes the previous attack into account and randomizes the tag's response and should achieve mutual authentication by requiring that the reader must answer to the challenge n_T. This protocol can be as follows:

$$\begin{array}{l} 1.\ R \to T : n_R \\ 2.\ T \to R : \{n_R, n_T\}_k \\ 3.\ R \to T : \{n_T\}_k \end{array}$$

Here, Alice & Bob notation shows its limit. It does not specify how the reader and the tag are supposed to check that the messages they received are of the expected form, and how they should react when the messages are not well formed.

This has to be precisely defined, since unlinkability depends on it. For instance, assume the tag does not check that the message he receives at step 3 contains n_T, and aborts the session if the received message in not encrypted with its own k. In such an implementation, an active attacker can eavesdrop a message $\{n_T\}_k$ sent by R to a tag T, and try to inject this message at the third step of another session played by T'. The tag T' will react by either aborting or by continuing the execution of this protocol. Depending on the reaction of the tag, the attacker will be able to infer if T and T' are the same tag or not.

In this example, the attacker adopts a malicious behavior that is not detected immediately by the tag who keeps executing the protocol. The fact that the tag passes successfully a conditional reveals crucial information about the agents that are involved in the execution. Our second condition, namely *well-authentication*, basically requires that when an execution deviates from the honest one, the agents that are involved cannot successfully pass a conditional.

Our main theorem states that these two conditions, frame opacity and well-authentication, are actually sufficient to ensure both unlinkability and anonymity. This theorem is of interest as our two conditions are fundamentally simpler than the targeted properties: frame opacity can be expressed using diff-equivalence and well-authentication is a trace property. In fact, they are both in the scope of existing automatic verification tools like ProVerif.

This result has been obtained by Lucca Hirschi, David Baelde and Stéphanie Delaune and has been published in the proceedings of S&P'16.

4 Conclusion

We have proposed two generic results allowing one to deal with a variety of primitives. Regarding the first result, directions for future work include the improvement of the implementation, e.g. avoid calling Maude unnecessarily. Another direction is to consider other AC operators such as Diffie-Hellman exponentiation and bilinear pairings. Regarding our second result, we plan to develop a mature implementation of our tool in order to make it widely accessible for the design and study of privacy-preserving 2-party protocols. We could also try to translate our conditions into more comprehensive guidelines helping the design of new privacy-enhancing protocols.

References

1. A. Armando, R. Carbone, L. Compagna, J. Cuéllar, and M. L. Tobarra. Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps. In *Proc. 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*, pages 1–10. ACM Press, 2008.
2. A. Armando et al. The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures. In *Proc. 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*, volume 7214, pages 267–282. Springer, 2012.

3. D. Baelde, S. Delaune, I. Gazeau, and S. Kremer. Verification of privacy-type properties for security protocols with XOR. Technical report, 2016.
4. B. Blanchet and A. Podelski. Verification of cryptographic protocols: Tagging enforces termination. In *Foundations of Software Science and Computation Structures (FoSSaCS'03)*.
5. L. Hirschi, D. Baelde, and S. Delaune. A method for verifying privacy-type properties: the unbounded case. In M. Locasto, V. Shmatikov, and Ú. Erlingsson, editors, *Proceedings of the 37th IEEE Symposium on Security and Privacy (S&P'16)*, San Jose, California, USA, May 2016. IEEECSP. To appear.
6. T. van Deursen and S. Radomirovic. Attacks on rfid protocols. *IACR Cryptology ePrint Archive*, 2008:310, 2008.