Merging the Cryptographic Security Analysis and the Algebraic-Logic Security Proof of PACE

Lassaad Cheikhrouhou[1], Werner Stephan[1],
Özgür Dagdelen[2], Marc Fischlin[2], Markus Ullmann[3]

[1]German Research Center for Artificial Intelligence (DFKI GmbH)
{lassaad,stephan}@dfki.de
[2]Technische Universität Darmstadt
marc.fischlin@gmail.com oezguer.dagdelen@cased.de
[3]Federal Office for Information Security (BSI)
markus.ullmann@bsi.bund.de

Abstract: In this paper we report on recent results about the merge of the cryptographic security proof for the Password Authenticated Connection Establishment (PACE), used within the German identity cards, with the algebraic-logic symbolic proof for the same protocol. Both proofs have initially been carried out individually, but have now been combined to get "the best of both worlds": an automated, error-resistant analysis with strong cryptographic security guarantees.

1 Introduction

The cryptographic protocol PACE (Password Authenticated Connection Establishment) is designed by the BSI for the establishment of authenticated radio frequency connections between contactless cards and readers [UKN+]. PACE is deployed in the German (electronic) identity card and should be used instead of the BAC (Basic Access Control) protocol in the inspection procedure of machine readable travel documents (ePassports) [EAC]. The protocol realizes a password-authenticated Diffie-Hellman (DH) key agreement between an RF-chip and a terminal. A successful run yields a fresh session key that is used by the reader to establish a secure connection to the RF-chip.

Two Views on the Security: The security of PACE has been analyzed by following two state-of-the-art approaches. A (symbolic) algebraic-logic security proof of PACE [CS10], in the Dolev-Yao (DY) model has been carried out in the Verification Support Environment (VSE) tool, yielding a machine generated proof of all its security properties. The DY model is based on a message algebra given by cryptographic operations and equations. The operations define the syntax of protocol messages while the equations in an abstract way formalize the computations that are necessary for the honest participants to execute the protocol. A subset of these operations is available for the adversary to analyze observed messages and to synthesize new ones. This attacker model is very powerful with respect to an unrestricted access (of the adversary) to the communication lines. On the other hand,

it limits the abilities of the adversary in attacking cryptography by assuming that exactly the algebraic (or symbolic) computations can be carried out by the adversary.

Concurrently, a cryptographic security analysis of PACE [BFK09] has been carried out based on the Bellare Rogaway (BR) security model, yielding a pencil-and-paper proof of the confidentiality and authenticity of the session key. The cryptographic proof follows the common complexity-theoretic approach to show that any security breach of the key exchange protocol within the security model can be used to refute any of the underlying assumptions and primitives. Vice versa, if the underlying primitives of the protocol are secure, then the proof shows – in terms of the complexity of the algorithms and concrete probabilities – how secure the PACE protocol is. Here, the adversary can be arbitrary (Turing) machines whose abilities to communicate with the system are determined by the model. Security proofs in the BR model consider strong adversaries who control the network, i.e., eavesdrop, modify and inject messages, and may learn leaked session keys or long-term keys of users.

Merging the Views: In this paper we describe an approach to merge the cryptographic security analysis in [BFK09] and the algebraic-logic security proof in [CS10], aiming at a reliability improvement in the security of PACE. Our merging approach is based on an *adequate* formalization of the BR model used within PACE that allows us to attribute public bit strings (values) to symbolic expressions for corresponding applications of the cryptographic functions. Sequences of alternating intruder queries and protocol oracle responses (BR traces) are attributed this way a *symbolic structure* that allows us to define DY computations in the BR model.

The main result is the theorem that symbolic traces of DY computations in the BR model are *equivalent* to symbolic traces in the algebraic-logic security proof. Equivalence is defined wrt. the capabilities and the knowledge of the adversary. This means that the algebraic-logic security proof of PACE in VSE provides us with a machine generated proof for the confidentiality of the session key in the BR-model, though for Turing machines restricted to DY computations.

Apart from that, our formalization of the BR model for PACE provides us with the formal means to define computational problems arbitrary adversary machines are confronted with (relative to the protocol model). The ultimate goal is an axiom-based formal proof of PACE's security in the BR model relative to an exhaustive listing of formally defined computational problems. In the paper we describe only the formalization of PACE's BR model (Sec. 4) and the equivalence of DY computations both in the BR and the DY model (Sec. 5). Prior to that we introduce PACE (Sec. 2) and we review its cryptographic security analysis and its algebraic-logic security proof (Sec. 3).

Related Work about Proof Integrations: Abadi and Rogaway [AR02, AR07] were the first to aim at bridging the gap between the two views on cryptography, the formal or symbolic view, and the complexity-based or computational view, through linking the two worlds explicitly. Their soundness theorem for encryption schemes is accomplished by mapping symbolic expressions to (probabilistic) cryptographic ensembles. Since then further works aimed at closing the gap between the two worlds in the same spirit (e.g., [BPW03, IK03, MW04a, MW04b, CLC08]).

A supportive approach towards linking the cryptographic and the symbolic world is given by the reactive simulatability (RSIM) framework of Pfitzmann and Waidner [PW00, PW01] and Canetti's concurrently proposed and similar-in-spirit universal composition (UC) framework [Can01]. The idea of the frameworks is to analyze cryptographic protocols in presence of ideal functionalities, similar to idealized primitives in the DY setting. So-called composition theorems then allow to conclude security of the combined protocol when the ideal functionalities are replaced by secure sub protocols. Hence, this allows to decompose the analysis of complex, potentially multi-session protocols into analysis of more handy single-session sub protocols. A series of soundness results [BPW04, CH06, Pat05, MH07, BU08] for these frameworks shows that any symbolically secure protocol (in a symbolic version of the RSIM/UC framework) is also secure in the computational framework; often such formal analysis have also been carried out explicitly. However, in order to be secure in the RSIM/UC frameworks, protocols need to satisfy very strong security guarantees in the first place. This rules out numerous protocols, especially quite a number of practical protocols. This means that the method is not applicable to a large class of protocols, and it is, in particular, unknown whether it applies to PACE.[1]

While computer-aided verification of *cryptographic protocols* is a well-established discipline, computer-aided verification of *cryptographic proofs* (in the computational model) is a quite new approach to combine the benefits of automated, error-free verification and the flexibility of cryptographic proofs to consider arbitrary adversaries strategies and to reason about the security of a protocol in terms of reduction to primitives (instead of idealizing the primitives). There are several approaches to build systems along this line, among which the recently proposed framework EasyCrypt [BGHB11] stands out. Our approach follows the same idea of verifying game-based proofs for the PACE protocol with the help of automated tools. Here, we used the available cryptographic proof for PACE [BFK09] and the previous efforts for the formal analysis for PACE in VSE [CS10] to merge the cryptographic reasoning with the symbolic analysis.

2 The PACE Protocol

PACE is a password-based key agreement protocol with mutual entity authentication. It takes place between the RF-chip (A) of a contactless smart card and an (inspection) terminal (B). After a successful run of PACE, an RF-chip and a terminal share a fresh session key, and the terminal can establish a secure connection to the RF-chip of the smart card using the established session key.

We have to point out that a successful PACE protocol run between an RF-chip A and a terminal B is only possible if the terminal has learned the appropriate password $pwd(A)$ of the RF-chip A at the outset, e.g., if the user typed it in at the terminal, or if it is read off the machine-readable zone in passports. This password $pwd(A)$ is stored on the RF-chip in secure memory and the way it is utilized in PACE guarantees that the chip originates

[1] Note that the cryptographic analysis of PACE has indeed been carried out in the game-based BPR security model, not in the UC framework.

from the smart card at hand.

2.1 Cryptographic Functions

PACE *messages* are computed out of passwords, random values (nonces), and basic generators for the underlying elliptic curve group, using the following (abstract) functions:

- $enc(\cdot,\cdot)$, $dec(\cdot,\cdot)$ for (symmetric) encryption and decryption, respectively,
- $dh(\cdot,\cdot)$ for the computation of Diffie-Hellman values, and
- $mac(\cdot,\cdot)$, $gen(\cdot,\cdot)$ for the computation of mac values and (fresh) generators, respectively.

The algebraic properties that are necessary to run the protocol are expressed by three equations:

- For encryption and decryption we have

  $$dec(m_0, enc(m_0, m_1)) = m_1 \text{ and } enc(m_0, dec(m_0, m_1)) = m_1.$$

  The second equation guarantees the absolute indistinguishability of failed and successful decrypting attempts. This property is necessary to obtain the resistance against offline password testing (see section 3.2).

- For the computation of a *common* DH key we have

  $$dh(dh(m_0, m_1), m_2) = dh(dh(m_0, m_2), m_1).$$

2.2 PACE Steps

To run the protocol with an RF-chip A, the terminal B does not only have to learn the password $pwd(A)$. It also has to access (in a protocol pre-phase) the domain parameters that include a *static* generator g for DH exchange in steps 2+3.

In the description of the protocol steps below we additionally make use of the meta-operator $\%$ to separate the sender's view (the left-hand side of $\%$) from the receiver's view (the right-hand side of $\%$). Unstructured messages in steps 1-5 below means that the receiver accepts any message without practically any check. Compare this with steps 6 and 7 below. Here, the respective receiver sees a message that can be compared with an expression determined from the own knowledge (the right-hand side of $\%$).

1. $A \longrightarrow B \ : \ enc(pwd(A), s_A) \,\%\, z$
2. $B \longrightarrow A \ : \ dh(g, x_1) \,\%\, X1$
3. $A \longrightarrow B \ : \ dh(g, y_1) \,\%\, Y1$

4. $B \longrightarrow A \ : \ dh(gen(dh(g, dec(pwd(A), z)), dh(Y1, x_1)), x_2) \ \% \ X2$

5. $A \longrightarrow B \ : \ dh(gen(dh(g, s_A), dh(X1, y_1)), y_2) \ \% \ Y2$

6. $B \longrightarrow A \ : \ mac(dh(Y2, x_2), Y2)$
$\% \ mac(dh(X2, y_2), dh(gen(dh(g, s_A), dh(X1, y_1)), y_2))$

7. $A \longrightarrow B \ : \ mac(dh(X2, y_2), X2)$
$\% \ mac(dh(Y2, x_2), dh(gen(dh(g, dec(pwd(A), z)), dh(Y1, x_1)), x_2))$

A starts the protocol by sending a nonce s_A encrypted with the own password $pwd(A)$ to B. The decryption of this message z by B with the password that B can determine while B is connected with A results in s_A, provided this password equals $pwd(A)$. The first DH exchange in steps 2+3 establishes a first DH value that is used with s_A and the static generator g in the computation of a *fresh* generator for the subsequent DH exchange in steps 4+5. The composition of these parameters by gen guarantees that the resulting generator is cryptographically as strong as g and binds this generator with the intermediate of s_A to the password $pwd(A)$. Thus, the DH value established in steps 4+5 can be determined only by participants that know the password. Its use in steps 6+7 to compute the mac authenticates the sender for the receiver. Each mac can be created only by a communication partner who has participated in the DH exchange of steps 4+5 after using the password.

3 The Cryptographic and Symbolic Security Analyses

3.1 The Cryptographic Proof

The PACE protocol is accompanied by a cryptographic security proof [BFK09]. The cryptographic proof follows the classical paper-and-pencil approach of defining a security model, describing the adversary's capabilities and goals, specifying a set of underlying cryptographic assumptions, and a mathematical proof that the adversary cannot succeed within the model under the assumptions.

As for the model, the authors in [BFK09] used the widely-deployed BR model [BR94] for authenticated key exchange (or, to be precise, the variations of Bellare-Pointcheval-Rogaway [BPR00] and of Abdalla et al. [AFP06] for the password-based case).The BR model defines a game between the adversary and a challenger oracle in which the powerful adversary can: (a) observe interactions between honest participants (i.e., RF-chips and terminals), (b) inject or modify transmissions in such communications, or even take over the side of one of the parties, (c) corrupt players, and (d) learn the derived DH key in executions. We note that the model considers multi-session executions, which may run concurrently. The adversary's goal is now to distinguish derived fresh DH keys from independent random strings in so-called test sessions, the idea being that good keys must still look random to the adversary. The model now demands that the adversary cannot distinguish the two cases, defined within the usual cryptographic notion of having negligible advantage over distinguishing the two cases by pure guessing with probability $\frac{1}{2}$.

The cryptographic proof now defines a set of assumptions, such as the hardness of the so-called PACE-DH number-theoretic problem (which is related to the DH problem), the security of the underlying message authentication code, and idealizing the deployed cipher and the hash function as a random oracle. Under these assumptions, it is shown (mathematically) that the PACE protocol is secure. The caveat here is that the proof takes into account the fact that the protocol is password-based, i.e., relies on low-entropy secrets, which can, in principle, be guessed by the adversary. The security claim shows that the – trivial and inevitable – on-line guessing of passwords, where the adversary tries to predict the password and then tests its guess in an execution with an honest party, is (essentially) the best adversarial strategy. In other words, PACE achieves optimal security.

The proof itself is carried out with the common technique of game hopping. That is, one starts with the actual attack and gradually eliminates success strategies of the adversary in each game hop, e.g., the ability to forge MACs. In the final game, the adversary provably cannot distinguish the DH keys from random keys anymore. One then accounts for the loss in the adversary's success probabilities and sums up all the losses for the hops. This sum gives an upper bound on the adversary's advantage.

The next theorem states the cryptographic security of the PACE protocol (for more general cases where the first Diffie-Hellman key exchange is substituted by a canonical protocol Map2Point). It obviously relates the adversary's characteristics like running time and success probability to the ones for the underlying primitives and assumptions. The theorem roughly shows that the best strategy for the adversary (unless it can break one of the primitives) is to guess the password (with probability $1/N$ among all passwords) and to mount a test run for this guess. Since there can be in total q_e executions the overall success probability is roughly q_e/N.

Theorem 3.1 ([BFK09]) *Let* Map2Point *be canonical and assume that the password is chosen from a dictionary of size N. In the random oracle model and the ideal cipher model we have*

$$\begin{aligned} \boldsymbol{Adv}^{ake}_{PACE}(t,Q) \quad \leq \quad & \frac{q_e}{N} + q_e \cdot \boldsymbol{Adv}^{gPACE-DH}_{\mathsf{Map2Point}}(t^*, N, q_h) \\ & + q_e \cdot \boldsymbol{Adv}^{forge}_{mac}(t^*, 2q_e) + \frac{2q_eN^2 + 8q_e^2N + q_cq_e}{\min\{q, |Range(\mathcal{H})\}} \end{aligned}$$

The resources of an adversary are captured by the time $t^* = t + \mathcal{O}(Q^2)$ and number of oracle queries $Q = (q_e, q_c, q_h)$ where q_e denotes the number of key exchanges, q_c the queries to the cipher oracle and q_h the queries to the random oracle. The theorem says that the advantage $\mathbf{Adv}^{ake}_{PACE}(t, Q)$ of any adversary having resources $(t*, Q)$ breaking the security of PACE is bounded by the pure guessing strategy q_e/N, the advantage of forging a MAC $\mathbf{Adv}^{forge}_{mac}(t^*, 2q_e)$, and the DH related hard problem gPACE-DH, denoted by $\mathbf{Adv}^{gPACE-DH}_{\mathsf{Map2Point}}(t^*, N, q_h)$ plus some negligible term. In short, an adversary is merely as successful as blind guessing whenever a secure MAC scheme is used in PACE.

The security reduction of PACE does not consider explicitly the hardness of the (ideal) block cipher. Instead, the security of the underlying block cipher is implicitly captured in the hardness of gPACE-DH problem and modeled as an ideal cipher.

3.2 The Algebraic-Logic Proof

The symbolic analysis of PACE has been carried out in the VSE tool, [CS10]. The resulting algebraic-logic proof handles explicitly the standard security properties (mutual authentication and confidentiality of session keys) and the following three properties, which are typical for password protocols:

- forward secrecy of session keys, i.e. that a successfully guessed password does not help to reveal the session keys that had been generated in previous runs,
- resistance against offline password testing, i.e. that the obtained protocol messages by an active intruder cannot be exploited offline to determine a *correct* password from a given list of candidate passwords, and
- forward secrecy of passwords, i.e. a *stronger* form of the resistance property where the intruder may use disclosed session keys in addition to protocol messages.

We note that these properties are implied by the BR model. They have been verified in the VSE tool where the protocol model consists of a set $\mathcal{T}_{PACE}$ of (finite) sequences (traces) $tr = \langle e_0, \ldots, e_{n-1} \rangle$ of (protocol-) *events*. The main events are of the form $says(a, b, m)$ where a protocol participant a sends a (symbolic) message m to some other participant b.

The set $\mathcal{T}_{PACE}$ is defined *inductively* by using predicates (rules) $R(tr, e)$ that describe the conditions for a given trace tr to be extended by a new event e. There is a predicate R_i for each line i of the protocol and an additional predicate R_f for the *fake case*. We have $R_f(tr, says(spy, b, m))$ for an arbitrary participant b iff the intruder (named spy) is able to *derive* m out of the observable messages in tr. We denote the set of (immediately) *observable messages* by $spies(tr)$ and the set of derivable messages (from $spies(tr)$) by $DY(spies(tr))$. The latter is an extension of $spies(tr)$ given by the reasoning capabilities of a DY intruder, i.e. by the application of the functions and the equations in Sec. 2.1.

In this algebraic-logic model, PACE properties are formalized on arbitrary traces from $\mathcal{T}_{PACE}$ and the corresponding machine-proofs are by induction on these event traces.

4 A Symbolic Formalization of the BR-Model

The global structure of the BR model, as depicted in Fig. 1, consists of two components: a PPT *adversary* machine and the *oracle* $\mathcal{O}$ that reacts on *queries* from the adversary by simulating steps of a given protocol. The most important *query* is of the form $send(adr, m)$ where m is a protocol message that has to be answered by a certain participant in a certain session, both given by adr, according to the protocol rules and the execution state which is part of the overall oracle state. The response value may depend on the application of cryptographic primitives where idealized functions are realized by *probabilistic* choices of the oracle. The adversary uses these responses to generate subsequent queries

by possibly *guessing* (sub-) messages and/or performing computations that are not necessarily included in the message algebra used in the DY model (see Sec. 2.1). To break the protocol, after termination of a particular session the adversary has to distinguish the session key stored in the oracle state from a random value with a considerably high probability. The test item is presented by the oracle as a reaction to a special query.

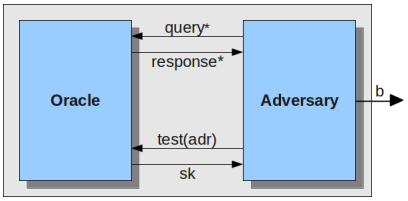

Figure 1: The BR Model

In [CSDF10] we formalize the BR model by specifying $\mathcal{O}$ as a state transition system and by defining *computation trees* $Comp(\mathcal{O} \parallel \mathcal{A})$, see Fig. 2, generated by the joint execution of the oracle and an arbitrary but fixed adversary $\mathcal{A}$.

Let $S_{\mathcal{O}}$ be the set of oracle states, $T_{\mathcal{A}}$ be the set of states of the adversary, $Q_{\mathcal{A}}$ the set of queries that can be generated by the adversary, and $R_{\mathcal{O}}$ the set of possible responses of the oracle. As mentioned above queries may contain protocol messages to be processed by the oracle.

Nodes of computation trees are quadruples (s, t, r, p) and (s, t, q, p), where $s \in S_{\mathcal{O}}$, $t \in T_{\mathcal{A}}$, $r \in R_{\mathcal{O}}$, and $q \in Q_{\mathcal{A}}$. The probability $p \in [0, 1]$ captures probabilistic choices of $\mathcal{O}$ and $\mathcal{A}$, respectively.

Starting from the initial states $s_0 \in S_{\mathcal{O}}$ and $t_0 \in T_{\mathcal{A}}$ the computation tree grows by alternating transition steps of the adversary $\mathcal{A}$ (●→●) and the oracle $\mathcal{O}$ (●→●). Steps of the adversary depend on the current state $t \in T_{\mathcal{A}}$ and the current response $r \in R_{\mathcal{O}}$. The state of the adversary which is not explicitly given can be thought of (among others) as representing past responses of the oracle. Outcomes of steps of the adversary consist of a new state $t' \in T_{\mathcal{A}}$ and a query $q \in Q_{\mathcal{A}}$. Since the adversary is given by a probabilistic machine, there is a finite set (or list) of outcomes, each equipped with a probability. Similarly, steps of the oracle depend on its current internal state $s \in S_{\mathcal{O}}$ and the current query $q \in Q_{\mathcal{A}}$. Outcomes of computations of the oracle consist of a new state $s' \in S_{\mathcal{O}}$ and a response $r \in R_{\mathcal{O}}$. Again, idealized (cryptographic) functions modeled by random choices lead to several possible outcomes.

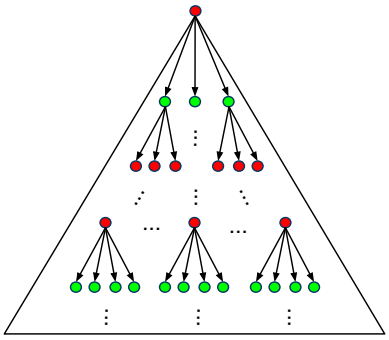
Figure 2: A computation tree $Comp(\mathcal{O} \parallel \mathcal{A})$

The behavior of the combined system is given by two functions

$$step_{\mathcal{O}} : S_{\mathcal{O}} \times Q_{\mathcal{A}} \to (S_{\mathcal{O}} \times R_{\mathcal{O}} \times [0, 1])^+ \quad \text{and} \quad step_{\mathcal{A}} : T_{\mathcal{A}} \times R_{\mathcal{O}} \to (T_{\mathcal{A}} \times Q_{\mathcal{A}} \times [0, 1])^*.$$

We specify the function $step_{\mathcal{O}}$ for PACE by transition rules that are *similar* to the rules $R(tr, e)$ in the inductive definition of the set $\mathcal{T}_{PACE}$ of (DY) traces (see Sec. 3.2). In particular, we use *symbolic expressions* to represent the control part of the session states as part of $s \in S_{\mathcal{O}}$. The symbolic expressions are interpreted by an operator $[\![.]\!]$ that evaluates

symbols. For instance, $[\![pwd(i)]\!]$ is evaluated to the password (value) $\pi \in PWD$ of the i-th participant acting in role A (i.e. $A[i]$) and $[\![nonce(l)]\!]$ to the l-th nonce (value) randomly chosen from a set RND.

Probabilistic values are generated using the oracle by certain additional state components. For example, the first PACE message is generated by a participant ($A[i]$) with password $\pi = [\![pwd(i)]\!]$ as follows. We consider all choices for a nonce $s_A = [\![nonce(l)]\!]$ from the set RND. For each s_A we check whether the state component $or_C(s) \subseteq PWD \times RND \times RND$ already contains a triple (π, s_A, z) in which case, we generate the response value as $[\![enc]\!](\pi, s_A) = z$. Otherwise, we consider all choices of z from a set of n possible values (i.e. from a subset of RND given by $or_C(s)$) to obtain $z = [\![enc]\!](\pi, s_A)$ after (π, s_A, z) was inserted into $or_C(s')$. The probability of the alternative transition steps generated this way is determined by the the cardinality $|RND|$ and n. Obviously, the same visible response z might lead to different successor states hidden in the oracle. In the control component of the oracle, we keep $enc(pwd(i), nonce(l))$ as the last successful step of this session. In this way, the control states and the responses of the oracle exhibit the same structure as traces in the symbolic (DY) model.

Computation paths end by situations where $\mathcal{A}$ halts, given by an *empty* image of $step_{\mathcal{A}}$.

Without further analyzing computation paths in the tree the only knowledge about the probabilities p in the outcomes of $step_{\mathcal{A}}$ is that their sum is one.

5 Analyzing (BR) Computation Trees

In this section, we describe the analysis of attacks based on computation trees. Attacks are given by paths where *the final problem* is solved, i.e. the adversary distinguishes a session key (by an honest participant) from a random value. Every success computation path contains a terminated session by an honest participant where the responses are computed as reaction on $send$ queries generated by the adversary. The messages in these queries have to be accepted by the honest participant as simulated by the oracle according to the protocol rules. In general, the adversary has to solve the problem of generating a correct protocol message by using knowledge obtained from all previously observed responses. The main question is whether there can be an adversary $\mathcal{A}$ that is able to solve all these problems including the final problem with a considerably high probability.

First we consider the adversary machines that are restricted to DY reasoning. We define this kind of machines relative to computation trees: (a) In a *DY-computation tree* $Comp(\mathcal{O} \parallel \mathcal{A})$, the steps of $\mathcal{A}$ are *deterministic*, i.e. every red node may not have more than *a* child node. This must not be confused with the fact that there are different strategies for adversaries. (b) Each value in a query must be associated a symbolic expression $\epsilon \in DY(\overline{\epsilon})$, where the list $\overline{\epsilon}$ contains the symbolic expressions associated to the corresponding directly observable (message) values.

The following theorem allows us to exploit the algebraic-logic proof as a VSE proof for the security of PACE (in the BR model) against DY restricted adversary machines. Note that DY adversaries cannot be removed by complexity considerations, since DY computations

are feasible.

Theorem 5.1 ([CSDF10]) *Let $Comp(\mathcal{O} \parallel \mathcal{A})$ be a (PACE) DY-computation tree. For all list $\bar{\epsilon}$ of the symbolic expressions associated to the public values on a computation path in $Comp(\mathcal{O} \parallel \mathcal{A})$ there exists a protocol trace tr in the DY model of PACE such that*

$$\forall \epsilon : \epsilon \in DY(\bar{\epsilon}) \Leftrightarrow \epsilon \in DY(spies(tr)).$$

This theorem is proved by induction on computation paths, [Neu11].

Regarding arbitrary adversary machines, the success to break the protocol shall be traced back to a *complete* list of local computation problems that are induced, as explained above, by protocol rules and that fit to the (partly idealized) assumptions on the cryptographic operations. This is work in progress. Up to now, we identified systematically a list of local (sub-) problems that we formally defined owing to notions given by computation trees, [CSDF10, Neu11]. Each local problem is defined by an input-output relation which is specified by input computation paths (in $Comp(\mathcal{O} \parallel \mathcal{A})$) and by *correct* outputs defined relative to associated states $s \in S_{\mathcal{O}}$. Further work will be an axiomatic system that can be used to prove the completeness of the identified problem list.

6 Conclusion

In spite of well-developed frameworks which guarantee soundness of algebraic-logic security proofs with respect to cryptographic security, their applicability to practical protocols is quite limited (see Sec. 1). For this reason, two independent security analysis for PACE were performed, a cryptographic security analysis and an algebraic-logic security proof, to explore PACE by applying state-of-the-art techniques. As the mathematical foundations are quite different, i.e., complexity theory versus mathematical logic, both analyses for PACE existed more or less concurrently at the outset of our merging attempt. Now, the described formalization of the BR-model provides us with a uniform formal framework for algebraic-logic and (computational-)cryptographic reasoning. This enables us to merge the cryptographic security analysis and the algebraic-logic security proof of PACE as described in this paper. We obtained a closed formal security analysis for PACE. The consequence is that the proven properties of the PACE protocol in both approaches can be linked. This enhances the reliability of the cryptographic (pencil-and-paper) proof as it can be supported by an accurate, formal algebraic-logic verification.

Here, we have only described the merging of a cryptographic security analysis and the algebraic-logic security proof of the password-based security protocol PACE. However, our approach, formalizing cryptographic analysis methodologies, seems to be much more general and applicable to a broad class of security protocols. This estimation has stimulated the project ProtoTo, funded by the German Federal Ministry of Education and Research (BMBF), about merges in related cases.

References

[AFP06] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-Based Authenticated Key Exchange in the Three-Party Setting. *IEE Proceedings — Information Security*, 153(1):27–39, March 2006.

[AR02] Martín Abadi and Phillip Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[AR07] Martín Abadi and Phillip Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). *Journal of Cryptology*, 20(3):395, July 2007.

[BFK09] Jens Bender, Marc Fischlin, and Dennis Kügler. Security Analysis of the PACE Key-Agreement Protocol. In *Proceedings of the Information Security Conference (ISC) 2009*, volume 5735 of *Lecture Notes in Computer Science*, pages 33–48. Springer-Verlag, 2009.

[BGHB11] Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. Computer-Aided Security Proofs for the Working Cryptographer. In *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 71–90. Springer, 2011.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In Bart Preneel, editor, *Advances in Cryptology — Eurocrypt '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 139+, 2000.

[BPW03] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A Composable Cryptographic Library with Nested Operations. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 03: 10th Conference on Computer and Communications Security*, pages 220–230. ACM Press, October 2003.

[BPW04] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A General Composition Theorem for Secure Reactive Systems. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 336–354. Springer, February 2004.

[BR94] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, August 1994.

[BU08] Michael Backes and Dominique Unruh. Limits of Constructive Security Proofs. In Josef Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 290–307. Springer, December 2008.

[Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001.

[CH06] Ran Canetti and Jonathan Herzog. Universally Composable Symbolic Analysis of Mutual Authentication and Key-Exchange Protocols. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403. Springer, March 2006.

[CLC08] Hubert Comon-Lundh and Véronique Cortier. Computational soundness of observational equivalence. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 08: 15th Conference on Computer and Communications Security*, pages 109–118. ACM Press, October 2008.

[CS10] Lassaad Cheikhrouhou and Werner Stephan. Meilensteinreport: Inductive Verification of PACE. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, 2010.

[CSDF10] Lassaad Cheikhrouhou, Werner Stephan, Özgür Dagdelen, and Marc Fischlin. Meilensteinreport: Integrating the Cryptographic Security Analysis and the Algebraic-Logic Security Proof of PACE. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, 2010.

[EAC] Technical Guideline: Advanced Security Mechanisms for Machine Readable Travel Documents – Extended Access Control (EAC), Password Authenticated Connection Establishment (PACE), and Restricted Identification (RI). Technical Report TR-03110, Version 2.05, Federal Office for Information Security (BSI).

[IK03] Russell Impagliazzo and Bruce M. Kapron. Logics for Reasoning about Cryptographic Constructions. In *44th Annual Symposium on Foundations of Computer Science*, pages 372–383. IEEE Computer Society Press, October 2003.

[MH07] Hirofumi Muratani and Yoshikazu Hanatani. Computationally Sound Symbolic Criteria for UC-secure Multi-Party Mutual Authentication and Key Exchange Protocols. In *IEICE Tech. Rep.*, volume 106 of *ISEC2006-150*, pages 59–64, Gunma, March 2007. Thu, Mar 15, 2007 - Fri, Mar 16 : Gunma Univ. (Kiryu Campus) (IT, ISEC, WBS).

[MW04a] Daniele Micciancio and Bogdan Warinschi. Completeness Theorems for the Abadi-Rogaway Language of Encrypted Expressions. *Journal of Computer Security*, 12(1):99–130, 2004.

[MW04b] Daniele Micciancio and Bogdan Warinschi. Soundness of Formal Encryption in the Presence of Active Adversaries. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer, February 2004.

[Neu11] Stephan Rouven Neumann. Integration der kryptographischen Sicherheitsanalyse und des algebraisch-logischen Sicherheitsbeweises von PACE. Master's thesis, Saarland University, Germany, 2011.

[Pat05] A.R. Patil. *On symbolic analysis of cryptographic protocols*. Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2005.

[PW00] Birgit Pfitzmann and Michael Waidner. Composition and Integrity Preservation of Secure Reactive Systems. In S. Jajodia and P. Samarati, editors, *ACM CCS 00: 7th Conference on Computer and Communications Security*, pages 245–254. ACM Press, November 2000.

[PW01] Birgit Pfitzmann and Michael Waidner. A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In *IEEE Symposium on Security and Privacy*, pages 184–, 2001.

[UKN$^+$] Markus Ullmann, Dennis Kügler, Heike Neumann, Sebastian Stappert, and Vögeler Matthias. Password Authenticated Key Agreement for Contactless Smart Cards. In *Proceedings of the 4-th Workshop on RFID Security, Budapest 2008*, pages 140–161.