Formalization in PVS of Balancing Properties Necessary for Proving Security of the Dolev-Yao Cascade Protocol Model

YURI SANTOS RÊGO[2] and MAURICIO AYALA-RINCÓN[1,2]

Departamentos de [1]Ciência da Computação e [2]Matemática, Universidade de Brasília

In this work, we present an algebraic approach for modeling the two-party cascade protocol of Dolev-Yao and for fully formalizing its security in the specification language of the *Prototype Verification System* PVS. Although cascade protocols could be argued to be a very limited model, it should be stressed here that they are the basis of more sophisticated protocols of great applicability, such as those which allow treatment of multiparty, tuples, nonces, name-stamps, signatures, etc. In the current algebraic approach, steps of the protocol are modeled in a monoid freely generated by the cryptographic operators. Words in this monoid are specified as finite sequences and the whole protocol as a finite sequence of protocol steps, that are functions from pairs of users to sequences of cryptographic operators. In a previous work, assuming that for *balanced protocols* admissible words produced by a potential intruder should be balanced, a formalization of the characterization of security of this kind of protocols was given in PVS. In this work, the previously assumed property is also formalized, obtaining in this way a complete formalization which mathematically guarantees the security of these protocols. Despite such property being relatively easy to specify, obtaining a complete formalization requires a great amount of effort, because several algebraic properties, that are related to the preservation of the balancing property of the admissible language of the intruder, should be formalized in the granularity of the underlying data structure (of finite sequences used in the specification). Among these properties, the most complex are related to the notion of *linkage property*, which allows for a systematic analysis of words of the admissible language of a potential saboteur, showing how he/she is unable to isolate private keys of other users under the assumption of balanced protocols. The difficulties that arose in conducting this formalization are also presented in this work.

1. INTRODUCTION

1.1 Motivation and proposal

The seminal Dolev-Yao (DY) cryptographic protocol model is 30 years old [DY83] and frequently cited because it provides a symbolic model of interest in the analysis

Both authors were supported by the Brazilian Research Council CNPq. This research was funded by the research funding agency of the Brasilia Federal District FAPDF under a PRONEX grant.

of decidability of secrecy under attacks of an active/passive intruder. Although decidability results have become known for more practical and relevant protocols (cf. [TEB05]), a variety of systems used in security verification of cryptographic protocols are known to adopt the Dolev-Yao rules [Cer01, Mao05]. Also, cascade encrypting or multiple encryption is still applied - in more sophisticated ways - in a range of computational systems (e.g. *TrueCrypt*). Therefore, being one of the most basic formal models of multiple encryption and packing a broad definition of an intruder capabilities, the DY two-party cascade protocol model plays a substantial role in analysis and formalization of security of some classes of protocols, as a first step towards a *full* verification of security.

Rules established by these protocols are usually guided by a given algorithm implemented in software or hardware and applied in order to preserve information security in several manners. Although the programming techniques used in this kind of development are of high quality in general, formal mathematical and logical analysis is necessary in several steps of this algorithmic development in order to guarantee that the implemented protocol is in fact secure and efficient. In a broader context, security analysis of cryptographic protocols is a tricky issue: proofs of security are rather difficult to check and there are many cases reported in the literature of protocols and security proofs which were later proved to be wrong [Mea03]. Automated reasoning and formal methods came up to the scene as a possible way to turn the security analysis of protocols more reliable and less error prone. Perhaps the most popular example of this success is the discovery of a possible attack upon the Needham-Schroeder protocol [NS78]. With the help of formal methods, Gavin Lowe discovered a gap in the Needham-Schroeder protocol after seventeen years of its introduction, a period during which the protocol was assumed correct [Low95]. The protocol was then modified and mechanically proved correct [Low96].

In this work, a formal approach to certify security of cryptographic systems is applied in order to prove that the DY two-party cascade protocol model [DY83] is in fact secure, whenever the conditions of security are fulfilled. The technique here adopted was rather straightforward: to specify the model in the language of PVS, and then to formalize the characterization of security. As the Model, as presented in the original paper, carries a canonical relation with algebra, we opted to specify it in an algebraic fashion through the use of the free monoid, that is an algebraic structure with a single associative binary operation and an identity element [Coh89], generated by the (abstract) cryptographic operators. In abstract

algebra, elements in free monoids are commonly viewed as finite sequences over some basic alphabet ([Coh89, Lot02]). By following this approach we were able to use a formalization that is close to the definitions and statements in [DY83]. Despite this being possibly the most straightforward way to fully formalize the theoretical model as originally introduced, several challenges arise. In particular, several low-level auxiliary formalizations related to data structures, such as sequences and sets, were necessary to obtain a complete formalization of security, and although these results are reported as part of this formalization, they can be moved to standard PVS prelude libraries.

The current presentation focuses on the difficulties inherent to the proof of algebraic properties reflected in these data structures, representing conditions under which protocols preserve the necessary properties that guarantee security. Such properties should imply that a potential saboteur is unable to extract private keys of other users using the admissible language, which is built as concatenations of balanced words. The complexity of the formalization of all the basic properties necessary to obtain a *complete* formalization of security, at the level of granularity of the selected data structures, is much higher than the complexity of proofs of the protocols' security properties, from the logical point of view. The choice of PVS was motivated by the fact that this is a theorem prover for higher-order logic, which among other advantages, makes straightforward the specification of properties over protocols, that are sequences of functions, and also the possibility of exploring the power of dependent types for discriminating the intruder from other users, as well as parametrizing words that are constructible in the admissible language of a potential intruder.

In previous work [NdMNAR10], the general lines for the formalization of the characterization of security of the DY model were presented. The necessity part was fully formalized, but several facts regarding the sufficiency part of this characterization were assumed without proof. These facts are related to crucial aspects of invariant properties of balanced words, that are represented as finite sequences of cryptographic operators. In that presentation, the characterization of security of the protocol was formalized; that is, security is guaranteed whenever two conditions hold:

—firstly, an *initial condition* that forces the existence of encryption operators in the first step of the protocol and,

—secondly, a *balancing property* that holds for each step of the protocol.

The latter property essentially forces occurrences of encryption operators for each user, for which at least one decryption operator occurs in any step of the protocol. Security is then a consequence of the fact that, following this balancing discipline in the construction of protocols, the admissible language used by any potential intruder will also be balanced, making impossible the isolation of decryption operators from the whole language allowed to the intruder. Axiomatizing or assuming without proof this fact makes possible the formalization of the sufficiency part of the theorem of security of cascade protocols in the deductive language of the proof assistant PVS ([OS97]), as presented in [NdMNAR10]. However, in order to obtain a *complete* formalization, no unproved assumptions are admissible. In this paper it is shown how a complete formalization was obtained through the proof of an exhaustive series of basic but non trivial lemmas related to the preservation of the balancing property of the admissible language. These lemmas refer to key properties expressed through the notions of user-balanced and linkage properties. The former expresses balancing relative to a specific user and the latter expresses balancing in fragments of words that a potential saboteur can use. In [DY83] the main theorem of security was proved assuming (Lemma 1) that normal balanced words of the admissible language preserve the balancing property. This property was proved through a sequence of lemmas in the appendix of [DY83] (Lemmas 9, 10 and 11), which are related to the most complex part of the DY model, that is the one that expresses the deductive power of a potential saboteur under hypothesis of a balanced protocol. More precisely, such lemmas state that

—concatenation of words satisfying the linkage property preserves this property, and then the admissible language of a potential saboteur preserves this property as well, and,

—normalization preserves the linkage property.

In this way, words built in the admissible language of a potential saboteur according to a balanced protocol will preserve the balancing property, implying that the saboteur will be unable to extract private keys of other users. Formalizing these lemmas is the part of the theory treated in this paper and that completes the formalization started in [NdMNAR10].

1.2 Related work

In addition to [NdMNAR10], other works have applied PVS to check properties of cryptographic protocols. In [DS97, ES00] PVS was used to analyze the security of authentication protocols. In [MR00], it is presented a dedicated strategy in order

to perform proofs of security protocols, which is based on the representation of protocol theories on a state-transition model. The representation used in the last work contrasts with the simple representation of protocols as sequences of functions used in the current algebraic approach that can be considered more adequate for the treatment of the original DY model. In [CMR01] the inductive engine of PVS has been used in order to develop a methodology of proof of inductiveness of secrecy and authenticity properties. That methodology is sound but incomplete failing to prove secrecy of secure protocols; in fact, secrecy is known to be in general a undecidable property (cf. [RT03, AC06]).

The paper [LHT07] provided a specification and verification in PVS of the intrusion-tolerant protocol enclaves [DCS02]. That work deals with a distributed protocol where the protocol goal has to be fulfilled even when a subset of the players are corrupted by a malicious party and can arbitrarily deviate from the protocol specification (the so-called Byzantine faults). Moreover, [LHT07] is not fully analyzed by using PVS. Its authenticity was treated using the model checker Murphi. Also, [BJ03] reports the use of PVS for formally verifying a system for ordered secure message transmission, but since the PVS specification code was not made available we were unable to reproduce the reported experiments.

Much work on formalization of security of protocols has been done in other proof assistants [ABCL09, BCM11, EKL+11]. This includes, among others, the remarkable inductive approach by L. Paulson in Isabelle [Pau99] and more recently the `Coq` development `CertiCrypt` which includes probability, complexity and game theoretical techniques in order to verify security [BGZB09]. More recently, Benaïssa presented a verification of security of the DY model in `event B` [Ben08]. Also, several automatic protocol verifiers can generate proofs of security of complex protocols that can be checked by the Isabelle/HOL theorem prover [BM09, MCB10]. Outstanding tools that should be mentioned include, among others: Avispa [ABB+05], whose objective is the analysis of security of industrial network protocols; ProVerif [Bla01], which verifies secrecy and authentication of cryptographic protocols in the DY model being able to handle primitives such as shared-key cryptography, hash functions, Diffie-Hellman key agreements, but in contrast to our algebraic approach it is based on a representation of protocols by Horn clauses; Scyther and Scyther-tool [Cre08], whose objective is automated analysis and proofs via Isabelle/HOL regarding various adversary models; Tamarin [SMCB12], which is a successor of the previous tools developed in Haskell, whose specification approach is based on equational and multi-set rewriting systems allowing specification of temporal prop-

erties, for which complex cryptographic properties are checked by rewriting modulo the equational theory; Maude-NPA [EKL+11], that is based on rewriting logic and narrowing in equational theories modulo (C, AC, ACI) and whose analysis relies on performing a so-called variant-based unification technique that through backwards search determines whether or not a final state can be reached; and, EasyCrypt [BGHZB11] which follows a game-based approach for reasoning about computations and properties with adversarial code and is related among others, with tools as CertiCrypt, previously mentioned.

Despite the existence of these works, we believe the current PVS development could be of great interest for the PVS community because it improves the scenario of available public libraries for the analysis of security in this proof assistant and because it chooses an algebraic straightforward specification of the DY framework, that takes advantage of the higher-order capabilities as well as of the flexibility allowed by dependent types of this proof assistant, allowing in this way a formalization approach very close to that of the analytical proofs. As previously mentioned, such framework is still used to model the basis of a class of cryptographic protocols and is known, in some cases, to provide security against all possible adversaries even when we do not consider perfect cryptography.

1.3 Organization

Section 2 presents the algebraic approach used to model the DY model and analytic sketches of the proofs. Section 3 presents details of the formalization of some of the key lemmas involved in the PVS development. Section 4 concludes and presents future work. The whole PVS development is available at http://www.mat.unb.br/∼ayala/publications.html and works in versions of PVS 5.n and 6.0.

2. THE DOLEV-YAO MODEL AND ITS SECURITY CHARACTERIZATION

The model is based on a system of public key cryptography in which each user $u \in U$, where U is a finite set of users, has an encryption operator E_u and a decryption operator D_u. A public secure directory includes all pairs (u, E_u), but for all $u \in U$, only u has knowledge about D_u. Suppose that the malicious users belong also to the set of users U of the system, and that they can obtain information through passive observation or active interaction in the communication network.

Encryption and decryption operators are algebraically inverse elements for each user: for all $u \in U$, $E_u D_u = D_u E_u = \lambda$, where λ denotes the empty word. Users interchange arbitrary information that is codified and decoded through *encryp-*

tion and *decryption* operators. Transmitted messages are arbitrary bit-strings, not drawn from the language of operators, and without any verifiable structure. The purpose of a two-party cascade protocol is transmitting secret messages between two users, which is done by exchanging encrypted messages that are represented as terms in a monoid. Thus, cryptographic operators can be modeled as the monoid freely generated by the encryption and decryption operators modulo the congruence given by $E_u D_u = D_u E_u = \lambda$ for all users $u \in U$ ([Coh89, Lot02]). In this structure only strings built with the generators should be considered.

Given a set of symbols A, we denote by A^* the set of all finite strings over the alphabet of symbols in A. Let $\Sigma = E \cup D = \{E_u \mid u \in U\} \cup \{D_u \mid u \in U\}$, and let Σ^* be the set of all finite strings over the alphabet of symbols in Σ. For all $\gamma \in \Sigma^*$, $|\gamma|$ denotes the length of the string γ, and for each i such that $0 \leq i < |\gamma|$, γ_i denotes the $(i+1)^{th}$ symbol (operator) of the string γ. Also, for every i, j such that $0 \leq i \leq j < |\gamma|$, $\gamma_{i,j}$ denotes the substring of γ from the $(i+1)^{th}$ until the $(j+1)^{th}$ symbol.

An arbitrary cryptographic operator will be denoted as O, and when necessary as O_u, in order to refer to its user $u \in U$. The opposite cryptographic operator of O_u is denoted as O_u^c. Thus, $E_u^c = D_u$ and $D_u^c = E_u$.

As usual when dealing with algebraic structures modulo a congruence, one introduces the concept of normal (or canonical) forms ([Coh89, Lot02]). Whenever $\sigma \in \Sigma^*$ has some substring of the form $O_u O_u^c$, that is, $\sigma = \sigma' O_u O_u^c \sigma''$ for some $u \in U$, $\sigma', \sigma'' \in \Sigma^*$, we say that σ can be reduced into $\sigma'\sigma''$ with respect to u. We can proceed with such reductions, applying the same process now over the word $\sigma'\sigma''$. Reducing σ with respect to a user u until no more reductions are possible provides the normal form of σ *with respect to* u. Denoting by $\overline{\sigma}^u$ the normalization of σ with respect to u and by $\overline{\sigma}$ the normalization of σ with respect to all users, we have that $\overline{\sigma}^u = \overline{\sigma' O_u O_u^c \sigma''}^u = \overline{\sigma'\sigma''}^u$, and analogously $\overline{\sigma} = \overline{\sigma' O_u O_u^c \sigma''} = \overline{\sigma'\sigma''}$.

Thus, normalizing a word means *recursive* elimination of *all* pairs of contiguous cancelable operators, that show up in each step of the process. As an example, let $\sigma = E_a D_c E_b D_b E_c E_a$. Then $\overline{\sigma} = \overline{E_a D_c E_b D_b E_c E_a} = \overline{E_a D_c E_c E_a} = \overline{E_a E_a} = E_a E_a$.

Observe that normalization goes accordingly to the congruence of the monoid: for all $u \in U$, $E_u D_u = D_u E_u = \lambda$. From the algebraic point of view, in this quotient monoid it is only necessary to work with normal or canonical forms. But in the context of the specification, that is the same of cryptography, discrimination between different representations of words or sequences of operators in the same equivalence class is essential.

In [DY83] two-party cascade protocols were defined as sequences of sequences of operators expressing the discipline that any pair of different users should follow to communicate. Each sequence of operators in a protocol is called a step. The user starting the communication sends messages according to the even steps and the other user, the receiver, according to the odd steps. Thus, in even steps only decrypt operators of the user starting the communication are possible and vice versa. Additionally, in [DY83], it is informally assumed that any protocol behaves exactly equal for any pair of users. To formalize the uniform behavior that any pair of users should follow, our choice was to specify protocols as sequences of functions, instead of sequences of sequences. Thus, in our formalization, each step is a function from pairs of users to sequences of operators. This way, restrictions on the use of decrypt operators in each step as well as the uniformity of protocol steps are easily given as functional restrictions.

Definition 1 (Two-party Cascade Protocol)**.** *A two-party cascade protocol α determines how pairs of users in a communication network should communicate and consists of a finite and non empty sequence of steps. Each step is a function α_i from pairs of users to a sequence of operators. Thus,*

$$\alpha = \alpha_{n-1}\alpha_{n-2}\cdots\alpha_2\alpha_1\alpha_0, \text{ where } n \geq 1, \text{ and}$$

$$\alpha_i : U \times U \to \Sigma^*, \text{ for all } 0 \leq i < n.$$

Even steps $\alpha_0, \alpha_2, \ldots$ and odd steps $\alpha_1, \alpha_3, \ldots$ respectively refer to steps of the protocol in which the first user communicates with the second user of the pair and vice versa. Additionally, $\forall i, 0 \leq i < n$, $\forall x, y, u, v \in U$, the following constraints hold:

i) $\alpha_i(x, y) \neq \lambda$ *and is normalized;*

ii) $\alpha_i(x, y) \in \{E_x, D_x, E_y\}^*$ *if i is even, i.e. when x communicates with y;*

iii) $\alpha_i(x, y) \in \{E_y, D_y, E_x\}^*$ *if i is odd, i.e. when y communicates with x;*

iv) $|\alpha_i(x, y)| = |\alpha_i(u, v)|$*;*

v) $\forall\ 0 \leq j < |\alpha_i(x, y)|$:

v.1) $(\alpha_i(x, y))_j = E_x$ *if, and only if* $(\alpha_i(u, v))_j = E_u$*;*

v.2) $(\alpha_i(x, y))_j = E_y$ *if, and only if* $(\alpha_i(u, v))_j = E_v$*;*

v.3) $(\alpha_i(x, y))_j = D_x$ *if, and only if* $(\alpha_i(u, v))_j = D_u$*;*

v.4) $(\alpha_i(x, y))_j = D_y$ *if, and only if* $(\alpha_i(u, v))_j = D_v$*.*

Given a pair of users $x, y \in U$ and a message M, the communication according to a given protocol α is done in the following manner:

x sends a message to y following the first step of the protocol, $\alpha_0(x,y)M$;

y answers to x with $\overline{\alpha_1(x,y)\alpha_0(x,y)}M$;

x answers to y with $\overline{\alpha_2(x,y)\overline{\alpha_1(x,y)\alpha_0(x,y)}}M$,

and so on.

Observe that since normalization is composable, that is for all $\alpha, \beta, \gamma \in \Sigma^*$, $\overline{\alpha\overline{\beta}\gamma} = \overline{\alpha\beta\gamma}$, repeated normalization in the previous step is not necessary. Composability of normalization holds also for normalization with respect to a user and both properties were formalized in PVS and applied at crucial points of our formalization of security of two-party cascade protocols, as will be seen in the Subsection 3.3 (cf. Lemma 2 and Lemma 3 on normalization with separation).

Following the rules of a given protocol α, for any two users $x, y \in U$ in communication, the constraints *ii)* and *iii)* basically restrict the use of the decryption operator D_x and D_y, respectively, to the user sending the message in each step of the protocol (even steps for x and odd for y). Constraints *iv)* and *v)* guarantee that, following the protocol, all pairs of users have to follow exactly the same behavior. Despite unusual, the even-odd fashion of protocol steps point out the (abstract) actions of a possible saboteur, which is our interest in the next section.

2.1 Security Characterization of Cascade Protocols

The following items characterize the admissible language of a possible saboteur in a communication network in which the users follow a well-defined cascade protocol α, that is, a protocol holding properties of Definition 1.

Let x, y and $z \in U$, where z is a possible saboteur. z can force applications of any step α_i of the protocol α, for $i > 0$, twofold: either supplanting x in order to obtain answers from y (odd steps of the protocol) or intercepting an eventual communication started between x and y and supplanting y in order to obtain answers from x (even steps of the protocol). This is described in detail in the two items presented below.

(1) z can obtain $\alpha_i(x,y)$, for all $1 \leq i < |\alpha|$ odd, starting a communication with y supplanting x. In the $(i-1)^{th}$ step of the communication, z sends to y any message M, obtaining as answer $\alpha_i(x,y)M$ (since y has to follow the protocol). This allows z to apply $\alpha_i(x,y)$ to any selected message M, for i odd;

(2) z can obtain $\alpha_i(x,y)$, for all $2 \leq i < |\alpha|$ even, observing passively the network and waiting until x establishes communication with y. Then, in the $(i-1)^{th}$ step of the communication, z intercepts the answer from y to x and replaces it

by sending to x any selected message M. Thus, x answers to z $\alpha_i(x,y)M$. In this way z is able to apply $\alpha_i(x,y)$ to any selected M, for $|\alpha| > i \geq 2$ even.

In addition to the two previous tricks, a potential saboteur z can use the language of words of the monoid generated by all the encryption operators and its own decryption operator.

(3) Since z is a user of the communication network, he can use the language generated by the admissible alphabet $\Sigma_0(z) := E \cup D_z$.

It can be argued that the item two above is not realistic because the saboteur would need to wait for a communication that may never take place. The original Dolev-Yao paper [DY83] discusses an interesting relaxation of the adversary model called the *impatient saboteur*, where a saboteur never waits for a conversation to be initiated. If a cascade protocol is secure, in the sense of the most general adversary considered here, then it is secure against the impatient saboteur.

Definition 2 (Admissible Language)**.** *Given a well-defined cascade protocol α and denoting the set of words, related to the first and second items above, as $\Sigma_1 := \{\alpha_i(x,y) \mid x,y \in U,\ x \neq y,\ 0 < i < |\alpha|\}$, one defines the* **admissible language** *of a possible saboteur z as*

$$\mathcal{AL}(z) := (\Sigma_0(z) \cup \Sigma_1)^*$$

Since a potential saboteur can observe the encoded messages between the other users during all steps of the communication, a protocol will be insecure whenever a saboteur is capable to extract the secret message at some step of the communication. Thus, if a saboteur can build a word using his/her admissible language, that according to the properties of the monoid cancels out the encoding at some step of the communication, he/she will be able to obtain the secret message. This is expressed in the following definition.

Definition 3 (Insecure/Secure Protocol)**.** *Consider a well-defined two-party cascade protocol given as $\alpha = \alpha_{n-1} \cdots \alpha_1 \alpha_0$, $n \geq 1$, and let $x,y,z \in U$ be different users. The protocol is said to be insecure if there exists $\gamma \in \mathcal{AL}(z)$ such that, for some $0 < j < n$*

$$\overline{\gamma(\alpha_{j-1}(x,y) \cdots \alpha_0(x,y))} = \lambda$$

Otherwise the protocol is said to be secure.

2.2 Characterization of the Security of Cascade Protocols

As previously mentioned and as presented in the original [DY83], two properties characterize security of two-party cascade protocols: one being a condition which ensures the presence of encryption in the first step of the protocol, and the other being a balancing property, which in some sense ensures that no decryption operators are "left loose" in any step of the protocol.

Definition 4 (Security Initial Condition - **IC).** *A cascade protocol α satisfies the initial condition (IC) if for all $x, y \in U$, there exists i, $0 \leq i < |\alpha_0(x,y)|$, such that $(\alpha_0(x,y))_i = E_u$, where $u \in \{x,y\}$; i.e., if the initial step of the protocol includes at least an encryption operator.*

Observe that if a protocol does not satisfy the initial condition of security the initial step of a communication is of the form $\alpha_0(x,y)M = D_x^k M$, for some positive integer k. The intruder can then use the word E_x^k in the admissible language to extract the secret message M.

Definition 5 (Balanced Word - **BP).** *A word $\sigma \in \Sigma^*$ has the balancing property (BP) with respect to a user $u \in U$ if, whenever there is some i, $0 \leq i < |\sigma|$, such that $\sigma_i = D_u$, there is j, $0 \leq j < |\sigma|$, such that $\sigma_j = E_u$.*

Definition 6 (Balanced Protocol)**.** *A cascade protocol α is said to be balanced if, for all $x, y \in U$ and for all $|\alpha| > i > 0$, the step $\alpha_i(x,y)$ satisfies BP with respect to x, if i is even, and with respect to y, if i is odd. In other words, for any step of the protocol, if it has a decryption operator, then it has an encryption operator, both for the same user.*

Observe that if the protocol is not balanced an attack can be built as follows. If the i^{th} step of the protocol is not balanced we have two cases to consider.

—Case i is odd. The saboteur z starts a communication with the user x sending any message M^* in the $(i-1)^{th}$ step, for which he will receive as answer $\alpha_i(z,x)M^*$. Since, $\alpha_i(z,x)$ is not balanced, then it includes D_x and $\alpha_i(z,x) \in \{D_x, E_z\}^*$. Thus, $\alpha_i(z,x) = \beta_1 D_x \beta_2$ and $\beta_1^c, \beta_2^c \in \{E_x, D_z\}^*$. Thus the saboteur is able to extract the private key of x from $\beta_1^c \alpha_i(z,x) \beta_2^c$.

—Case $i > 0$ is even. Since the notion of security is conservative, potentially any user x would start a communication with a saboteur z. As in the previous case, z will send to x any message M^* in the $(i-1)^{th}$ step, and then receive as answer $\alpha_i(z,x)M^*$.

The initial condition of security and the condition of balanced protocol will suffice for elaborating the theorem that characterizes security. Before doing that, we refer to a key lemma in this theory.

The following lemma, which is perhaps the most difficult or at least the most elaborated part of the analytic theory of the DY model, deals with the balancing property related to words in the admissible language of an intruder z, i.e. $\mathcal{AL}(z)$. This lemma simplifies the proof of the Theorem 1 of characterization of security, presented at the end of this section, because it encapsulates the subjacent technicalities involved in its formalization.

Lemma 1 (BP for Normalized Words of the Admissible Language of Balanced Protocols)**.** *Given a balanced cascade protocol α and $z \in U$ then, for all $\eta \in \mathcal{AL}(z)$, $\overline{\eta}$ satisfies BP with respect to* ***all*** *users $a \in U,\ a \neq z$.*

Lemma 1, was only axiomatized (i.e., assumed without proof) in [NdMNAR10] in order to present a formal proof of the Theorem 1 of characterization of security, and its current formalization depends on Lemmas 9, 10 and 11 (according to the original numeration in [DY83]), that will be presented below. Lemma 9 is necessary in order to prove Lemma 10 and both the latter and Lemma 11 are necessary in order to conclude the proof of Lemma 1. In the formalization of these lemmas, two additional definitions related to the balancing property relative to a specific user are necessary.

Definition 7 (User-Balanced Word)**.** *Let $a \in U$ and $\pi \in \Sigma^*$. One says that π is a-balanced if the following implication is true: ($\pi = D_x \delta D_y$, for some $x, y \in U, x \neq a \neq y$ and $\overline{\delta}^a \cap D \subseteq \{D_a\}$) implies that $\overline{\delta}^a$ is balanced with respect to a.*

Definition 8 (Linkage Property - **LP**)**.** *Let $z \in U$ and $\eta \in \Sigma^*$. One says that η satisfies the linkage property with respect to z if, for any π subword of the word $D_z \eta D_z$, π is a-balanced, for all $a \in U, a \neq z$.*

The analysis of whether a word is balanced then reduces to the verification of the balancing property for all its subwords for which only decryption operators (for a unique user) occur. This is done by checking whether the linkage property holds.

Lemma 9 (Linkage Property Composition)**.** *Let $\eta, \mu \in \Sigma^*$ be words that satisfy LP w.r.t. z. Then $\eta\mu$ satisfies LP w.r.t. z.*

Lemma 10 (Linkage Property for the Admissible Language)**.** *Consider a balanced cascade protocol α, $z \in U$ and let $\eta \in \mathcal{AL}(z)$. Then η satisfies LP w.r.t. z.*

Lemma 11 (Normal Forms Preserve Linkage Property)**.** *Let $\eta \in \Sigma^*$ such that η satisfies LP. Then $\overline{\eta}$ satisfies LP too.*

Sketches of analytical proofs of the previous three lemmas are available in [DY83], but their complete formalizations require a large series of proofs of mundane properties of the data structures being used in order to represent protocols. Although these proofs are reported as part of this full formalization, it should be stressed here that several of them could be moved directly to other PVS libraries about properties of the involved data structures.

The proof of Lemma 10 is by induction on the inductive construction of the admissible language $((\Sigma_0(z) \cup \Sigma_1)^*)$ and depends on proving that words in $\Sigma_0(z) \cup \Sigma_1$ satisfy LP (basis of the induction) and application of Lemma 9 in the inductive step.

The proof of Lemma 11 is also done by induction. In this case, on the number of recursive steps applied in the normalization of the word η. In the induction basis, it is proved that after eliminating from η the first, from left to right, occurrence of contiguous opposite operators, either $D_u E_u$ or $E_u D_u$, for some $u \in U$, the resulting word satisfies LP. In the inductive step this argument is applied once again.

In PVS, the formalization of Lemma 11 depends on the specification of the notion of normalization that is given basically through two specified functions presented below. The function `first_cancelable` takes as argument a reducible sequence and detects the first contiguous occurrence of opposite operators. The second function, `normalizeseq`, uses the first function in order to detect recursively the first occurrence of contiguous opposite operators and eliminates them from the sequence.

```
first_cancelable(seq : reducibleseq) : RECURSIVE nat =
 IF   areopcomplements?(seq(0),seq(1)) THEN 0
 ELSE 1 + first_cancelable(^(seq,(1,seq`length-1)))
 ENDIF
 MEASURE seq`length-1

normalizeseq(seq : seqOps) : RECURSIVE seqOps =
 IF normalseq?(seq) THEN seq
 ELSE LET (firstCancPos : nat) = first_cancelable(seq) IN
  IF firstCancPos=0 THEN normalizeseq(seq^(2,seq`length-1))
  ELSE normalizeseq(seq^(0,firstCancPos-1) o
            seq^(firstCancPos+2,seq`length-1))
  ENDIF
 ENDIF
 MEASURE seq`length
```

Several decisions taken during the specification are relevant in order to obtain a full formalization of the main lemmas. Observing the function `first_cancelable`,

one notices that the input sequence `seq` is indexed from 0 to its length minus one (`seq`length - 1`). The function `areopcomplements?` checks whether two operators either are opposite or not. The type of the parameter of the function `first_cancelable` is the type of reducible sequences that is a subtype of the type of sequences of operators (`SeqOps`), as used as the parameter of the function `normalizeseq`. The operators "^" and "o" denote, respectively, subsequences and concatenation (or append) of sequences. Thus, `(seq^(0,firstCancPos-1) o seq^(firstCancPos+2,seq`length-1)` denotes the sequence obtained by eliminating the operators at positions `firstCancPos` and `firstCancPos + 1` of the sequence `seq`, that, in other words, is the sequence obtained by eliminating the first contiguous occurrence of opposite operators in `seq`, since `firstCancPos` was set as the position of the first cancelable contiguous occurrence.

The next theorem, whose proof depends on Lemma 1, characterizes security of two-party cascade protocols.

Theorem 1 (Characterization of Security of Cascade Protocols ([DY83]))**.** *A two-party cascade protocol is secure if and only if*

—*it satisfies the initial condition and*

—*is balanced.*

The proof of Theorem 1 is divided in the proof of necessity and the proof of sufficiency:

—The former, that is to prove that a secure protocol satisfies the initial condition and should be balanced, is obtained by contrapositive argumentation: if a protocol does not satisfy the initial condition or is not balanced it is proved to be insecure.

—The latter is proved by contradiction. Let $x, y \in U$, and suppose that x starts communication with y. Suppose, by reduction to the absurd, that the protocol satisfies IC and is balanced, but is insecure; thus, there exists $\gamma \in \mathcal{AL}(z)$, such that $\overline{\gamma\alpha_0(x,y)} = \lambda$. A separation in two cases is then possible: in the first case, E_y appears in $\alpha_0(x,y)$, and then it is possible to show that $\overline{\gamma}$ is not balanced, because it should contain an operator D_y (since D_y does not occur in $\alpha_0(x,y)$), which contradicts Lemma 1; in the second case, E_y does not occur in $\alpha_0(x,y)$, which implies that D_x also does not occur in the first step of the protocol and consequently Lemma 1 is contradicted again.

Here, it is relevant to stress that the formalization of this theorem was firstly given in [NdMNAR10] and was exactly based on the previous logical reasoning for

analyzing necessity and sufficiency. But only in the current work it is presented a full formalization in which it is included the complete treatment of Lemma 1, which implied an effort much greater than the necessary for this logical analysis.

3. FORMALIZATION IN PVS

In this section, several deductive techniques applied in the formalization of the balancing properties of the DY model are presented. The formalization is available in the PVS files. Here only a brief description is possible focusing on the most relevant aspects.

The *Prototype Verification System* PVS is a theorem prover for higher order logic with an elaborated type system in which subtyping and dependent types are allowed. In a higher order logic language, as the one of the specification language of PVS, one can quantify over relational and functional variables. This makes straightforward the specification of properties of two-party cascade protocols that are sequences of relational (functional) objects according to Definition 1. For instance, sufficiency lemma related to the proof of Theorem 1 is specified as the lemma below, in which a well-defined protocol `prot` is universally quantified. Also, dependent types are used in order to quantify over triplets of users `x, y` and `z` such that are mutually different.

```
alpha0_and_bal_secure : LEMMA
FORALL (prot : welldefined_protocol,
        x : U, y : U | x /= y,
        z : U | z /= x AND z /= y) :
alpha0ContainsE?(prot, x, y) AND
balanced_cascade_protocol?(prot) =>
           secure_protocol?(prot, x, y, z)
```

The basic data structures used in the formalization are `finite_sequences` and `sets` that are available in the *prelude* theory of PVS[OS97]. The whole hierarchy of the formalization is presented in Fig. 1. The main theory, named `CascadeProtocolsSecurity`, contains the specification of Theorem 1 and imports theories for the formalization of sufficiency and necessity, respectively, `SecurityNecessity` and `SecuritySufficiency`. The theory `Examples` contains simple examples of application of the theory to prove security of specific protocols. The focus in this paper is on the formalization of lemmas related to balancing properties (Lemmas 9, 10 and 11), which are formalized inside the theories `UserBalancingProperty` and `UserMonoidCryptOps`.

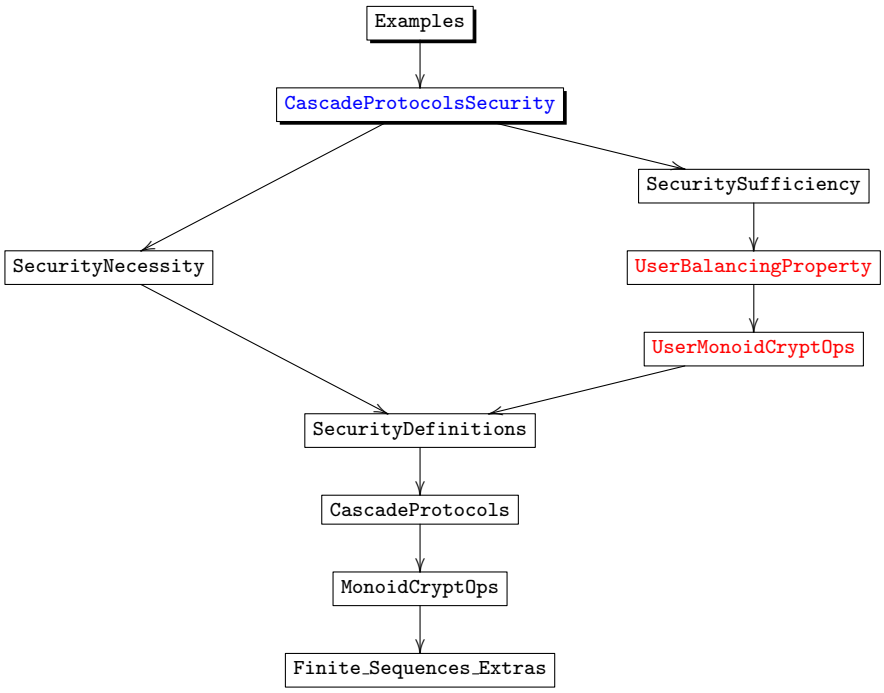

Fig. 1. Hierarchy of theories and subtheories — formalization of security of cascade protocols

The theory `finite_sequences_extras` imports the prelude theory for finite sequences and includes additional necessary lemmas and properties about this data structure, as well as about finite sets, that are not available in the PVS *prelude* library.

The theory `MonoidCryptOps` includes general specifications about the cryptographic operators and the theory of monoids freely generated by the language of cryptographic operators modulo the congruence given by elimination of contiguous opposite operators. In this theory, the notion of normal form is given.

The theory `CascadeProtocols` formalizes the basic notions about two-party cascade protocols. The theory `SecurityDefinitions` formalizes the notions of security of cascade protocols.

The theory `UserMonoidCryptOps` includes specifications and formalizations about properties of sequences of cryptographic operators relative to specific users. In this theory, notions similar to the ones given in `MonoidCryptOps` are specified; for instance, the notion of normal form relative to a specific user is given. These relativizations are necessary in order to deal with notions such as user balanced and

linkage property (eg, Defs. 7 and 8), among others, that are essential to formalize the central balancing property (Lemma 1) necessary in the proof of sufficiency of the main security characterization theorem.

The theory `UserBalancingProperty` includes the formalizations of the crucial Lemmas (1, 9, 10 and 11) as well as specifications of the notion of user-balanced and the linkage property.

As previously mentioned, the crucial part of the Theory is included in the theories `SecurityNecessity` and `SecuritySufficiency` formalizing necessity and sufficiency of Theorem 1. Here, the focus is on the balancing properties necessary for sufficiency that are formalized in the theories `UserBalancingProperty` and `UserMonoidCryptOps`.

3.1 Verification of Balancing Lemma 1

Using Lemmas 9, 10 and 11, Lemma 1 was formalized following essentially the analytic proof in [DY83]. However, it was detected the need of an additional technical property in order to obtain a full formalization: for any word of the form $D_z \eta D_z$, where $\eta \in \Sigma^*$, if for some $a \in U$ the operator D_a occurs in η, then there exists a subsequence of $D_z \eta D_z$ of the form $D_x \delta D_y$ containing the occurrence of D_a and such that $x \neq a \neq y$. Analytically, this property is very simple, but technically its formalization is non trivial. This kind of mundane properties are recurrent in the formalization and represent a great deal of the whole formalization effort.

Lemma 1 was formalized inside the theory `UserBalancingProperty`, and its specification is given as below.

```
userBalancing : LEMMA
  FORALL (prot : welldefined_protocol,
              z : U,
          gamma : gammaT | gamma_welldef?(prot,gamma, z),
              w : U | w /= z) :
  balanced_cascade_protocol?(prot) =>
    balancedseq_wrt?(normalizeseq(extract_gamma(gamma)), w)
```

This PVS lemma specifies the following: let `prot` be a well-defined protocol, $z, w \in U$ be different users and $\gamma \in \mathcal{AL}(z)$ a word in the admissible language of the protocol `prot` for the intruder z. Thus, if `prot` is a balanced protocol, then the normalization of γ, that is $\overline{\gamma}$, is balanced with respect to the user w. In other words, for all users different from a possible saboteur z, the normalization of any admissible word is balanced with respect to the other users.

In the specification above, `gammaT` represents the type of finite sequences of allowed strings (finite sequences of operators) for the model of protocols under consideration; `gama_welldef?` is a tertiary relation that expresses the fact that the finite sequence of words `gamma` is a sequence of words either in $\Sigma_0(z)$ or Σ_1 (according to the protocol `prot`), that is, the concatenation of words in `gamma` belongs to the admissible language of the intruder z. `balanced_cascade_protocol?` and `balancedseq_wrt?` are *boolean* unary and binary relations, respectively, for balanced protocols and words balanced with respect to a user. The function `extract_gamma` builds the word of concatenation of words in the finite sequence `gamma`, that is a word in $\mathcal{AL}(z)$. `normalizeseq`, as previously mentioned, recursively builds the normalization of the input word according to the congruence of the monoid, eliminating all contiguous opposite operators.

The proof of Lemma 1 is done applying Lemmas 10 and 11 as follows:

> Let $\eta \in \mathcal{AL}(z)$ be a word in the admissible language. Suppose, by contradiction, that for some $a \in U$, $\overline{\eta}$ does not satisfy the balancing property with respect to a. Thus, D_a occurs in $\overline{\eta}$, but E_a does not, which implies that $\overline{\eta}$ does not satisfy the linkage property. This contradicts Lemmas 10 and 11, since in first place, η satisfies the linkage property because it is an admissible word built from a balanced protocol and, in second place, normalizations of words that satisfy the linkage property preserve this property.

The language of proof of PVS follows the Gentzen sequent style. PVS uses an interactive proof language in which inference rules of the sequent calculus are applied by means of proof commands. The proof is started by a sequent containing as antecedent the premises of the conjecture or objective to be proved and as succedent its conclusion. Both the succedent and antecedent of a sequent are a sequence of formulas. The former is interpreted as the conjunction and the latter as the disjunction of the corresponding sequence of formulas. Proof commands should be applied until the proof is concluded or until one detects errors in the conjecture. Proofs are stored in a file of proof commands.

The command `prove` starts the proof of some selected objective. This is illustrated below for the Lemma 1. Items above the symbol `|-------` represent the antecedent or premises of the sequent, and items below this symbol, the succedent or conclusions.

```
|-------
[1] FORALL (prot : welldefined_protocol, z : U,
           gamma : gammaT | gamma_welldef?(prot,gamma, z),
               w : U | w /= z) :
```

```
balanced_cascade_protocol?(prot) =>
   balancedseq_wrt?(normalizeseq(extract_gamma(gamma)), w)
```

The consequent starts exactly as Lemma 1. By an application of the proof command of Skolemization and propositional simplification one obtains the following sequent.

```
{-1} balanced_cascade_protocol?(prot)
|-------
{1}balancedseq_wrt?(normalizeseq(extract_gamma(gamma)),w)
```

Observe that formulas in the antecedent and in the succedent are indexed respectively with negative and positive integers. Active formulas, which are those involved in the last proof command, are indexed inside curly brackets, and the other ones in square brackets.

As antecedent or unique premise of the last sequent one has that `prot` is a (well-defined) balanced protocol and one should prove that the normalization of `extract_gamma(gamma)`, that is `normalizeseq(extract_gamma(gamma))`, named `reducedGamma` below, is balanced with respect to `w`. Lemmas 10 and 11 can be invoked applying the PVS proof command `lemma`. The application of this command includes as new premises in the antecedent of the sequent the selected lemmas that are adequately instantiated according to the objective being proved, obtaining in this way the sequent below.

```
{-1} linkage_property?(extract_gamma(gamma), z) =>
                              linkage_property?(reducedGamma, z)
[-2] balanced_cascade_protocol?(prot) =>
                       linkage_property?(extract_gamma(gamma), z)
[-3] balanced_cascade_protocol?(prot)
|-------
{1} balancedseq_wrt?(reducedGamma, w)
```

At this point, according to the analytic proof previously explained, concluding the formula in the succedent seems trivial, but it involves technicalities as those previously mentioned. In fact, the development of the proof is not short, and several properties related to finite sequences and the theory of monoids are necessary. More than ten additional technical lemmas were necessary in order to conclude the proof of this lemma, which uses more than 160 proof commands.

3.2 Verification of the Linkage Property for the Admissible Language (Lemma 10)

While illustrating the use of PVS in the formalization, we present the outlines of the proof of Lemma 10.

Lemma 10 about linkage property for the admissible language is specified as

```
balanced_prot_imp_linkage_in_sigmas : LEMMA
  FORALL(prot:  welldefined_protocol, z : U,
         eta : gammaT | gamma_welldef?(prot,eta, z)) :
    balanced_cascade_protocol?(prot) =>
                linkage_property?(extract_gamma(eta), z)
```

This states that, given a balanced cascade protocol `prot` and a user $z \in U$, all words of the admissible language $\mathcal{AL}(z)$, built in the specification as `extract_gamma(eta)`, satisfy the linkage property.

The proof is started by applying the command `prove` obtaining the initial sequent below.

```
  |-------
{1} FORALL (prot: welldefined_protocol, z: U,
            eta: gammaT | gamma_welldef?(prot, eta, z)):
     balanced_cascade_protocol?(prot) =>
        linkage_property?(extract_gamma(eta), z)
```

The proof is by induction in the length of the sequence `eta`. This method is selected by applying the proof command `(measure-induct+ "eta`length" ("eta"))` to which PVS returns the following sequent having as first premise the inductive hypothesis, that is for any sequence with length less than the length of the initial sequence, called now `x!1`, it satisfies the linkage property.

```
{-1} FORALL (y: {eta: gammaT | gamma_welldef?(prot, eta, z)}):
       y`length < x!1`length =>
         balanced_cascade_protocol?(prot) =>
          linkage_property?(extract_gamma(y), z)
{-2} balanced_cascade_protocol?(prot)
  |-------
{1} linkage_property?(extract_gamma(x!1), z)
```

Analytically, it is enough to apply induction and Lemma 9, but some specificities of the data structure should be considered. The case in which the length of `x!1` is zero, is proved easily. Now, if `x!1` has length equal to one, some considerations are necessary. Supposing that `x!1`length = 1`, one applies an auxiliary lemma called `admissible_language_sat_link_property`, that states that all words of the language $\Sigma_0(z)$ or $\Sigma_1$ satisfy the linkage property. Invoking this auxiliary lemma gives the sequent below, where the formula `[1] x!1`length = 0` in the succedent excludes the case in which the length of `x!1` is zero.

```
{-1}  FORALL (prot: welldefined_protocol, z: U, delta: seqOps):
        (balanced_cascade_protocol?(prot) AND
          ( member(delta, sigma2_3(prot)) OR
    wellDefInSigma1?(delta, z)) )
         => linkage_property?(delta, z)
```

```
[-2]  x!1‘length = 1
[-3]  FORALL (y: {eta: gammaT | gamma_welldef?(prot, eta, z)}):
        y‘length < x!1‘length =>
         balanced_cascade_protocol?(prot) =>
          linkage_property?(extract_gamma(y), z)
[-4]  balanced_cascade_protocol?(prot)
  |-------
[1]   x!1‘length = 0
[2]   linkage_property?(extract_gamma(x!1), z)
```

Lemma `admissible_language_sat_link_property` appearing as premise {-1} guarantees in this case that if `x!1` is a unique word, that is a sequence of operators, it satisfies the linkage property.

Supposing now that the length of `x!1` is greater than one, one has the sequent below. The formulas `[1] x!1`length = 0` and `[2] x!1`length = 1` in the succedent exclude the cases in which the length of `x!1` is either one or zero and are excluded in the sequel.

```
[-1]  FORALL (y: {eta: gammaT | gamma_welldef?(prot, eta, z)}):
        y‘length < x!1‘length =>
         balanced_cascade_protocol?(prot) =>
          linkage_property?(extract_gamma(y), z)
[-2]  balanced_cascade_protocol?(prot)
  |-------
[3]   linkage_property?(extract_gamma(x!1), z)
```

In order to apply the induction hypothesis, one instantiates it, that is the antecedent `[-1]`, with the finite sequence `x!1` without its first word, that is the sequence `x!1^(1, x!1`length - 1)`, by applying the command `(inst -1 ‘‘x!1^(1, x!1`length - 1)’’)`. Since the sequence `x!1^(1, x!1`length - 1)` has length less than the length of `x!1`, this sequence can be used in the induction hypothesis. This together with the fact that `prot` is a balanced protocol gives rise to the simplification of the induction hypothesis (after this instantiation) as the premise {-1} in the sequent below, that states that `extract_gamma(x!1^(1, x!1`length - 1))` satisfies the linkage property.

```
{-1}  linkage_property?(extract_gamma(x!1^(1, x!1‘length - 1)), z)
[-2]  balanced_cascade_protocol?(prot)
  |-------
[3]   linkage_property?(extract_gamma(x!1), z)
```

Selecting the previous sequence for the instantiation of the induction hypothesis is adequate for the application of Lemma 9, because this lemma guarantees that the

concatenation of words that satisfy the linkage property also satisfy this property. Lemma 9 is specified in PVS with the name `linkage_property_composition`, and its invocation at this point of the proof gives the sequent below, in which the first premise corresponds to this lemma.

```
{-1}  FORALL (mu, eta: seqOps, z: U):
        linkage_property?(mu, z) AND linkage_property?(eta, z) =>
         linkage_property?(mu o eta, z)
[-2]  linkage_property?( extract_gamma(x!1^(1, x!1`length - 1)), z )
[-3]  balanced_cascade_protocol?(prot)
  |-------
[3]   linkage_property?(extract_gamma(x!1), z)
```

The first word of the finite sequence `x!1`, that is `x!1`seq(0)`, belongs either to $\Sigma_0(z)$ or $\Sigma_1$ and consequently, it satisfies the linkage property, as previously mentioned. By induction hypothesis, the rest of the sequence, that is `x!1^(1, x!1`length - 1)`, satisfies this property as well. Lemma 9 is instantiated with `mu` as `x!1`seq(0)`, the first element of `x!1`, and `eta` as the rest of the sequence. Since `x!1`seq(0) o extract_gamma(x!1 ^ (1, x!1`length - 1)) = extract_gamma(x!1)`, one concludes that the linkage property also holds for `x!1`. Proving this equality also requires several additional technicalities (almost ninety PVS proof commands are applied) not presented here, but available in the PVS formalization. One obtains as last sequent the one presented below.

```
{-1}  linkage_property?(x!1`seq(0), z) =>
       linkage_property?(extract_gamma(x!1), z)
[-2]  linkage_property?( extract_gamma(x!1^(1, x!1`length - 1)), z )
[-3]  balanced_cascade_protocol?(prot)
  |-------
[3]   linkage_property?(extract_gamma(x!1), z)
```

At this point, it is enough to guarantee that `x!1`seq(0)` satisfies the linkage property. This is done in the same way as in the case where `x!1`length = 1` by application of the auxiliary lemma `admissible_language_sat_link_property`.

3.3 Formalization of technical properties

A great amount of the effort done in this formalization is related to the construction of proofs of specific properties of sequences representing the quotient monoid generated by the cryptographic operators. To illustrate this, we present the formalization of a specific lemma which guarantees that the word user-balancing property of Definition 7 holds for subsequences of the concatenation of sequences satisfying the linkage property of Definition 8, and which is applied in the proof of Lemma 9. To do so, it is necessary to characterize normalization relative to a user z of

the concatenation of sequences δ and σ, that is $\overline{\delta\sigma}^z$. Two main cases are to be considered: either the last operator of δ and the first of σ are operators for the user z, or not. In the first case, let $\delta = \delta' O_z^k$ and $\sigma = (O_z^c)^j \sigma'$ be normal words with respect to z, where $k, j \geq 0$ represent the number of adjacent complementary operators O_z and O_z^c occurring in a row in the suffix of δ and in the prefix of σ, respectively. In the second case, $\delta = \delta' O_u$ or $\sigma = O_u \sigma'$ for $u \neq z$. Two specific auxiliary lemmas arise:

Lemma 2 (Relative normalization with separation)**.** *Let $\delta \in \Sigma^*$, $z, a \in U$, such that $z \neq a$. Then $\delta = \delta' O_a \delta''$, for some $\delta', \delta'' \in \Sigma^*$, implies*

$$\overline{\delta}^z = \overline{\delta'}^z O_a \overline{\delta''}^z$$

Lemma 3 (Relative normalization without separation)**.** *Let $\delta, \sigma \in \Sigma^*$ be normal sequences with respect to $z \in U$, that is $\overline{\sigma}^z = \sigma$ and $\overline{\delta}^z = \delta$, and let $j, k \geq 0$ be such that j and k are maximal with $\delta = \delta' O_z^j$ and $\sigma = (O_z^c)^k \sigma'$. Then, $j \geq k$ implies*

$$\overline{\delta\sigma}^z = \delta' O_z^{j-k} \sigma'$$

Otherwise,

$$\overline{\delta\sigma}^z = \delta' (O_z^c)^{k-j} \sigma'$$

Here, we explain the formalization of the former lemma, that has been specified in PVS as the lemma `user_normalize_break` in the theory `UserMonoidCryptOps` included below.

```
user_normalize_break : LEMMA FORALL (seq : seqOps, z, a : U, i : nat) :
  (a /= z & i < seq`length - 1 & user(seq(i)) = a) =>
     normalizeseqZ(seq, z) =
      IF i = 0 THEN
        seq^(0,0) o normalizeseqZ(seq^(1, seq`length - 1), z)
      ELSE
        normalizeseqZ(seq^(0,i-1), z) o seq^(i,i) o
                      normalizeseqZ(seq^(i+1,seq`length - 1), z)
      ENDIF
```

The formalization of this lemma is based on the application of a number of auxiliary lemmas proved by induction, from which two key lemmas discriminate the case in which the first part of the sequence is normal with respect to z and the opposite case. The latter case, is specified as the lemma below.

```
  user_normalize_separation2 : LEMMA FORALL (seq : seqOps,
                    a: U, z:U | a /= z,  i : below[seq`length]) :
```

```
(reducibleseqZ?(seq,z) AND user(seq(i)) = a ) =>
  LET k = first_cancelableZ(seq,z) IN
   k < i =>
    normalizeseqZ(seq,z) =
     normalizeseqZ(seq^(0, i-1),z) o seq^(i,i) o
                  normalizeseqZ(seq^(i + 1 , length(seq) - 1), z)
```

The proof of this lemma consists of more than two hundred proof steps and is done basically by application of two additional auxiliary technical lemmas: the first one states that when δ is normal with respect to z, $\overline{\delta O_a \sigma}^z = \delta O_a \overline{\sigma}^z$, and the second one that, in general $\overline{\overline{\alpha}^z \beta}^z = \overline{\alpha\beta}^z$. From these lemmas, one obtains $\overline{\delta O_a \sigma}^z = \overline{\overline{\delta}^z O_a \sigma}^z = \overline{\delta}^z O_a \overline{\sigma}^z$. The former lemma is specified as `user_normalize_separation1`, presented below.

```
user_normalize_separation1 : LEMMA FORALL (seq : seqOps,
                 a: U, z:U | a /= z, i : nat | i < seq`length) :
(reducibleseqZ?(seq,z) AND user(seq(i)) = a ) =>
 LET k = first_cancelableZ(seq,z) IN
 k > i =>
  normalizeseqZ(seq,z) =
     seq^(0, i) o normalizeseqZ(seq^(i + 1 , seq`length - 1), z)
```

The formalization of this lemma is by induction on the length of the sequence `seq` and consists of more than five hundred lines of proof commands in which thirty one invocations to other auxiliary technical lemmas are necessary.

This explanation can continue in this way, enumerating a long series of necessary auxiliary lemmas, which are related to the algebraic properties of the monoid freely generated by the cryptographic operators, the quotient monoid and the quotient monoid relative to a specific user, as well as to its representation as the data structure of sequences of operators. Analogously, the formalization of Lemma 3, specified in the theory `UserMonoidCryptOps` as Lemma `user_normalize_composition`, consists of more than 750 proof commands in which more than ninety invocations to other auxiliary lemmas are done. Summarizing, what is relevant to clarify at this point is that most of the necessary formalization work is related to the mechanical proofs of these auxiliary technical lemmas. All these proofs can be executed through the specification code that is available.

3.4 Verification of security of specific protocols

To illustrate how security of specific protocols can be checked a subtheory `Examples` (see Fig. 1) was added. Basically, a protocol step should be defined as a function

from pairs of users into sequences of cryptographic operators. For instance, $\alpha_0 : U \times U \to \Sigma^*$ such that $(u, v) \mapsto E_u E_v D_u$ is specified as

```
alpha_0 : alphabeta  = LAMBDA(x,y:U) : E_x o E_y o D_x
```

A protocol `prot0` is specified as sequences of functions of this form. For instance, `prot0 = alpha_0 o alpha_1 o alpha_2 o alpha_3`, where

```
alpha_0 : alphabeta  = LAMBDA(x,y:U) : E_x o E_y o D_x
alpha_1 : alphabeta  = LAMBDA(x,y:U) : E_y o E_x o D_y
alpha_2 : alphabeta  = LAMBDA(x,y:U) : D_x o E_y o E_x o E_y o D_x
alpha_3 : alphabeta  = LAMBDA(x,y:U) : E_y o E_x o E_x o D_y
```

Then, security of `prot0` is proved as a corollary of the main theorem (`theorem1` in the theory `CascadeProtocolsSecurity`) after proving that `prot0` is balanced and satisfies the initial condition.

Observe that whenever it is not possible to prove the initial condition (Def. 4), or if the balancing property (Def. 6) fails for a step of a protocol, counterexamples can be built according to the observations after the corresponding definitions.

It is important to stress here that the original Dolev-Yao result is all about proving that security of cascade protocols is decidable, yet the formalization does not provide a decision procedure for each protocol. To prove the security of a protocol, one has to actually construct a proof in PVS that it is balanced and satisfies the initial condition.

Quantitative Data.

The whole PVS development, consists of the ten theories depicted in Fig. 1 in which the specification part consists of approximately 1.600 lines of code (or almost 80 KB), and the proof part consists of approximately 54.000 lines of proof commands (or 3.7 MB), all that including comments.

Auxiliary lemmas related to the data structure of sequences, the monoid and the quotient monoid relative to a specific user, were specified respectively in the PVS theories `finite_sequences_extras`, `MonoidCryptOps` and `UserMonoidCryptOps` (see Fig. 1). These three theories alone consist of approximately 890 lines of specification code (or 43 KB) and almost 28.000 lines of proof commands (or 1.7 MB). The whole development consists of 384 proofs, from which 209 are TCCs (type correctness conditions generated, but not necessarily automatically proved, by the prover) and the other 175 are lemmas fully formalized.

Regarding the formalization of necessity and partially formalization of sufficiency in [NdMNAR10], although several proofs in other theories were adjusted to obtain the current development, what is essentially new is inside the theories `UserMonoidCryptOps` and `UserBalancingProperty`. These two theories consist of 717 lines of specification (almost 37 KB) and almost 39.000 lines of proof commands (2.84 MB). Both these theories include 209 theorems from which 123 are TCCs.

4. CONCLUSION AND FUTURE WORK

The formalization of the theorem of characterization of security of the DY model of two-party cascade protocols was concluded, based on a full formalization about the preservation of balancing properties of normalization of words of the admissible language of potential malicious users. A great variety of properties about the monoid freely generated by the language of encryption and decryption operators was necessary, as well as properties about the data structure of finite sequences. The latter was used twofold: firstly, to represent words of the monoid, that are finite sequences of cryptographic operators and secondly, to represent protocols, that are finite sequences of protocol steps, that are basically functions of pairs of users into sequences of operators in this monoid.

Several algebraic properties related to normalization of words in the quotient monoid according to the congruence given by the elimination of opposite cryptographic operators were necessary. And particularly, these properties and other additional specific ones were adapted for the case of the quotient monoid relative to the congruence restricted to a specific user. The latter was necessary in order to establish some crucial properties, such as the linkage property and the property of being user-balanced, that are essential in order to prove preservation of this balance in the whole admissible language of any potential saboteur.

Part of the effort invested in the formalization of the characterization of security of the DY model was concentrated on the proof of basic technical properties over the structure of monoids and its representation as sequences. The formalization of these auxiliary lemmas is worth because it was fundamental in order to conclude the full formalization of security of the DY model, but also because these properties could be incorporated to basic PVS libraries about monoids and sequences. This formalization represents an important kernel that can be applied in order to obtain further formalizations of logical properties of other variants of the DY model such as protocols for impatient saboteurs. For impatient saboteurs, the definition of admissible language uses only the language of words of the monoid generated by all

the encryption operators and its own decryption operator, and also words obtained starting communication with other users, perhaps supplanting some other user; but words obtained by intercepting messages between other users are avoided (cf. items (1), (2) and (3) before Definition 2). The characterization of security of cascade protocols for the impatient saboteur (Theorem 6 in [DY83]) relaxes the imposition of the balancing property only for odd steps of the protocol, while the initial condition extends to normalization of all messages sent during the communication between users $(\alpha_0(x,y), \overline{\alpha_1(x,y)\alpha_0(x,y)}, \ldots)$. To formalize this new characterization of security following the analytical proof in [DY83], several results from the current PVS theory could be reused: monoid and relative-to-a-user monoid properties, normalization, etc, but specific properties for new definitions are necessary (e.g. user substrings, strongly user-balanced, etc.). More elaborated variants of the DY model such as name-stamp protocols (as presented in [DY83]) will require too much additional effort since the language of cryptographic operators and their basic laws changes, which will imply a different specification of the basic concepts and, therefore, a completely new formalization. The same applies for other models such as multiparty models, models with authentication mechanisms, models with blind signatures, etc.

Another important point to stress here, is that the choice of a specific data structure to represent protocols (in our case sequences of functions) determines all proofs in the theory. But it is not necessary to change all proofs when other data structures are chosen. This is illustrated in this paper, when the logical sketch of the proofs of the Theorem 1 of characterization of security and of the Lemma 1 of balancing property of normalizations of words of the admissible language were explained. The current formalization works for the specific data structure of finite sequences that was selected to represent the quotient monoids, but other data structures such as lists and strings could be selected reusing proofs through formalizations of isomorphical properties between finite sequences and the other chosen data structure.

References

[ABB+05] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA tool for the automated validation of internet security protocols and

applications. In *Proceedings of the 17th international conference on Computer Aided Verification*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer Verlag, 2005. 1.2

[ABCL09] M. Abadi, B. Blanchet, and H. Comon-Lundh. Models and proofs of protocol security: A progress report. In *21st International Conference on Computer Aided Verification (CAV'09)*, volume 5643 of *Lecture Notes in Computer Science*, pages 35–49. Springer Verlag, 2009. 1.2

[AC06] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theor. Comput. Sci.*, 367(1-2):2–32, 2006. 1.2

[BCM11] D. Basin, C. Cremers, and C. Meadows. *Model Checking Security Protocols*, chapter 24. Springer Verlag, 2011. To appear. 1.2

[Ben08] N. Benaïssa. Modelling Attacker's Knowledge for Cascade Cryptographic Protocols. In *ABZ '08: Proc. of the 1^{st} Int. Conf. on Abstract State Machines, B and Z*, volume 5238 of *Lecture Notes in Computer Science*, pages 251–264. Springer Verlag, 2008. 1.2

[BGHZB11] G. Barthe, B. Grégoire, S. Heraud, and S. Zanella-Béguelin. Computer-Aided Security Proofs for the Working Cryptographer. In *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 71–90. Springer Verlag, 2011. 1.2

[BGZB09] G. Barthe, B. Grégoire, and S. Zanella Béguelin. Formal certification of code-based cryptographic proofs. In *36^{th} ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages POPL*, pages 90–101, 2009. 1.2

[BJ03] M. Backes and C. Jacobi. Cryptographically Sound and Machine-Assisted Verification of Security Protocols. In *20th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2607 of *Lecture Notes in Computer Science*, pages 675–686. Springer Verlag, 2003. 1.2

[Bla01] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 82–96. IEEE Computer Society, 2001. 1.2

[BM09] A. D. Brucker and S. Mödersheim. Integrating Automated and Interactive Protocol Verification. In *Formal Aspects in Security and Trust, 6th International Workshop, FAST 2009*, volume 5983 of *Lecture Notes in Computer Science*, pages 248–262. Springer Verlag, 2009. 1.2

[Cer01] I. Cervesato. The Dolev-Yao Intruder is the Most Powerful Attacker. In *Proceedings of the Sixteenth Annual Symposium on Logic in Computer Science - LICS'01*, pages 16–19. IEEE Computer Society Press, 2001. 1.1

[CMR01] V. Cortier, J. Millen, and Harald Ruess. Proving secrecy is easy enough. In *14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 97–110. IEEE Comp. Soc. Press, 2001. 1.2

[Coh89] D. E. Cohen. *Combinatorial Group Theory: a topological approach.* Cambridge UP, 1989. 1.1, 2

[Cre08] C. J. F. Cremers. The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In *Computer Aided Verification, 20th International Conference, CAV*, volume 5123 of *Lecture Notes in Computer Science*, pages 414–418, 2008. 1.2

[DCS02] B. Dutertre, V. Crettaz, and V. Stavridou. Intrusion-tolerant enclaves. In *Proc. of the IEEE International Symposium on Security and Privacy*, pages 216–224, 2002. 1.2

[DS97] B. Dutertre and S. Schneider. Using a PVS Embedding of CSP to Verify Authentication Protocols. In *Theorem Proving in Higher Order Logics, TPHOL's 97*, volume 1275 of *Lecture Notes in Computer Science*, pages 121–136. Springer Verlag, 1997. 1.2

[DY83] D. Dolev and A. C. Yao. On the Security of Public Key Protocols. *IEEE. T. on Information Theory*, 29(2):198–208, 1983. 1.1, 2, 2.1, 2.2, 2.2, 2.2, 1, 3.1, 4

[EKL+11] S. Escobar, D. Kapur, C. Lynch, C. Meadows, J. Meseguer, P. Narendran, and R. Sasse. Protocol analysis in Maude-NPA using unification modulo homomorphic encryption. In *Proceedings of the 13th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP*, pages 65–76. ACM, 2011. 1.2

[ES00] N. Evans and S. Schneider. Analysing Time Dependent Security Properties in CSP Using PVS. In 6^{th} *European Symposium on Research in Computer Security ESORICS*, volume 1895 of *Lecture Notes in Computer Science*, pages 222–237. Springer Verlag, 2000. 1.2

[LHT07] M. Layouni, J. Hoofman, and S. Tahar. Formal Specification and Verification of the Intrusion-Tolerant Enclaves Protocol. *International Journal of Network Security*, 5(3):288–298, 2007. 1.2

[Lot02] M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge UP, 2002. 1.1, 2

[Low95] G. Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Information Processing Letters*, 56(3):131–133, 1995. 1.1

[Low96] G. Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. *Software - Concepts and Tools*, 17(3):93–102, 1996. 1.1

[Mao05] W. Mao. A structured operational semantic modelling of the Dolev-Yao threat environment and its composition with cryptographic protocols. *Computer Standards & Interfaces*, 27(5):479–488, 2005. 1.1

[MCB10] S. Meier, C. J. F. Cremers, and D. A. Basin. Strong Invariants for the Efficient Construction of Machine-Checked Protocol Security Proofs. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010*, pages 231–245. IEEE Computer Society, 2010. 1.2

[Mea03] C. Meadows. Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends. *IEEE J. on Selected Areas in Communications*, 21(1):44–54, 2003. 1.1

[MR00] J. K. Millen and H. Rueß. Protocol-independent secrecy. In *IEEE Symposium on Security and Privacy*, pages 110–209, 2000. 1.2

[NdMNAR10] R.B. Nogueira, F.L.C. de Moura, A. Nascimento, and M. Ayala-Rincón. Formalization Of Security Proofs Using PVS in the Dolev-Yao Model. In *Computability in Europe CiE 2010 (Booklet)*, 2010. 1.1, 1.2, 2.2, 2.2, 3.4

[NS78] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Comm. of the ACM*, 21:993–999, 1978. 1.1

[OS97] Sam Owre and Natarajan Shankar. The formal semantics of PVS. Technical report, SRI-CSL-97-2, Computer Science Laboratory, SRI International, Menlo Park, CA, August 1997. Available at http://pvs.csl.sri.com/. 1.1, 3

[Pau99] L. C. Paulson. Proving Security Protocols Correct. In *14^{th} Annual IEEE Symposium on Logic in Computer Science LICS*, pages 370–383, 1999. 1.2

[RT03] M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions, composed keys is np-complete. *Theor. Comput. Sci.*, 299(1-3):451–475, 2003. 1.2

[SMCB12] B. Schmidt, S. Meier, C. J. F. Cremers, and D. A. Basin. Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In *25th IEEE Computer Security Foundations Symposium, CSF*, pages 78–94. IEEE, 2012. 1.2

[TEB05] F.L. Tiplea, C. Enea, and C. V. Birjoveanu. Decidability and complexity results for security protocols. In Edmund M. Clarke, Marius Minea, and Ferucio Laurentiu Tiplea, editors, *VISSAS*, volume 1 of *NATO Security through Science Series D: Information and Communication Security*, pages 185–211. IOS Press, 2005. 1.1