INSTITUT FÜR INFORMATIK

Infinite State AMC-Model Checking for Cryptographic Protocols

Detlef Kähler, Ralf Küsters, Tomasz Truderung

Bericht Nr. 0702
Februar 2007

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Infinite State AMC-Model Checking for Cryptographic Protocols

Detlef Kähler[1], Ralf Küsters[2], and Tomasz Truderung[3]

[1] University of Kiel, Germany
kaehler@ti.informatik.uni-kiel.de
[2] ETH Zurich, Switzerland
ralf.kuesters@inf.ethz.ch
[3] University of Kiel, Germany; Wroclaw University, Poland
tomasz.truderung@ii.uni.wroc.pl

Abstract. Only very little is known about the automatic analysis of cryptographic protocols for game-theoretic security properties. In this paper, we therefore study decidability and complexity of the model checking problem for AMC-formulas over infinite state concurrent game structures induced by cryptographic protocols and the Dolev-Yao intruder. We show that the problem is NEXPTIME-complete when making reasonable assumptions about protocols and for an expressive fragment of AMC, which contains, for example, all properties formulated by Kremer and Raskin in fair ATL for contract-signing and non-repudiation protocols. We also prove that our assumptions on protocols are necessary to obtain decidability.

1 Introduction

The design of cryptographic protocols is highly error-prone as these protocols have to achieve their security goals even in presence of an adversary who controls part of the communication network and in presence of dishonest parties who deviate from the protocol specification. Rigorous analysis of these protocols is therefore indispensable. Several algorithms and tools for the (fully) automatic analysis of cryptographic protocols have been developed and successfully applied (see, e.g., [19, 3]). One of the fundamental results in the area is that the security of protocols can be decided for a bounded number of sessions and w.r.t. the so-called Dolev-Yao intruder, with no restrictions put on the size of messages (see, e.g., [21, 19, 5]). However, these results are restricted to reachability properties, such as secrecy and authentication. They do not apply to cryptographic protocols with more complex, game-theoretic security requirements, such as those for non-repudiation and contract-signing protocols, including, for example, different versions of fairness, timeliness, balance, and abuse-freeness (see, e.g., [17, 16]). For instance, one version of fairness for non-repudiation protocols taken from [17] requires that (dishonest) Bob does not have a strategy (in collaboration with certain communication channels) to reach a state in which he has a proof

of origin but (honest) Alice does not have a strategy (against the other players) to obtain her proof of receipt.

Only recently a first decidability result for a *specific* game-theoretic security property, namely balance, has been obtained [14, 12] (see the related work). The goal of the present work is to study decidability and complexity of cryptographic protocol analysis in a much more general setting in which game-theoretic security properties are expressed in terms of the Alternating-time μ-Calculus (AMC), which strictly contains ATL*, and hence, provided a suitable set of propositional variables, also fair ATL [2].

More precisely, in this paper we formalize the possible executions of protocols along with the Dolev-Yao intruder in terms of a certain class of infinite-state concurrent game structures [2], which we call *security-specific concurrent game structures*. These concurrent game structures have an *infinite* state space since at every execution step the Dolev-Yao intruder can choose messages to be sent to principals among an *infinite* set of possible messages. Similar to [15], we model the realistic situation that (honest and dishonest) principals may take actions at the same time and may receive/write several messages from/to other principals at the same time. Since many cryptographic protocols with game-theoretic security requirements assume resilient channels (also called secure channels here), i.e., channels that, unlike the network, are not under the control of the Dolev-Yao intruder, our model comprises such channels. We distinguish between direct and scheduled secure channels: A direct secure channel is a direct link between principals. Messages sent on scheduled secure channels are first sent to a buffer before being delivered to the intended recipient. The buffer is a player in the security-specific concurrent game structure and may team up with (honest or dishonest) principals or other scheduled secure channels, as can be specified by an AMC-formula. Honest principals are specified by finite edge-labeled trees where an edge is labeled by a rule which describes a possible receive-send action of a principal at the current step. Vertices in these trees may have self-loops to allow a principal to stay in the current state.

Based on the security-specific concurrent game structures that we define, game-theoretic security requirements for protocols can conveniently be expressed in terms of AMC-formulas (or alternatively, ATL*-formulas). In order to decide whether a given protocol satisfies a given security property, expressed as AMC-formula, one has to decide the AMC-model checking problem over the security-specific concurrent game structures, where the input to the problem is the protocol (which together with the Dolev-Yao intruder induces the security-specific concurrent game structure) and the AMC-formula.

Our main technical results are as follows: We show that the above model checking problem is undecidable for a class of protocols in which honest principals may be what we call *non-greedy*, i.e., they may ignore received messages even though they conform to the protocol specification. The undecidability result holds for a relatively simple, fixed AMC-formula. Fortunately, in typical protocol specifications, honest principals are greedy, i.e., they do not ignore messages that conform to the protocol specification. Hence, requiring honest principals

to be greedy is reasonable from a practical point of view. We also exhibit another source of undecidability, namely protocols that involve *scheduled* secure channels *from the Dolev-Yao intruder* (i.e., dishonest principals) to honest principals. This undecidability result holds for greedy principals and again a fixed, simple AMC-formula. Since we allow the Dolev-Yao intruder to sent messages over direct secure channels to principals, disallowing scheduled secure channels from the Dolev-Yao intruder to honest principals does not limit the power of the intruder. These undecidability results show that to obtain decidability it is necessary to consider only protocols with greedy principals and without scheduled secure channels from the Dolev-Yao intruder to honest principals. For this class of protocols we indeed obtain decidability, more accurately (co-)NEXPTIME-completeness, of the model checking problem for an expressive fragment of AMC, consisting of what we call $\mathcal{I}$-positive ($\mathcal{I}$-negative) AMC-formulas, where $\mathcal{I}$ is the name of the Dolev-Yao intruder in the concurrent game structure. An AMC-formula φ is $\mathcal{I}$-positive if all subformulas of φ of the form $\langle\langle A\rangle\rangle\bigcirc\psi$ with $\mathcal{I} \in A$ fall under an even number of negations and all subformulas of φ of the form $\langle\langle A\rangle\rangle\bigcirc\psi$ with $\mathcal{I} \notin A$ fall under an odd number of negations; a formula is $\mathcal{I}$-negative if its negation is $\mathcal{I}$-positive. We subsume the set of $\mathcal{I}$-positive and $\mathcal{I}$-negative formulas under the notion $\mathcal{I}$-monotone formulas. The same terminology can be applied to ATL*-formulas. It is easy to see that the property of being $\mathcal{I}$-positive/-negative is invariant under the translation from ATL* to AMC as described in [2]. Kremer and Raskin were the first to express game-theoretic security properties in terms of fair ATL [17, 16]. It turns out that all the properties that they have formulated, including for instance various forms of fairness, timeliness, balance, and abuse-freeness, fall into the $\mathcal{I}$-monotone fragment of ATL*, and hence, the $\mathcal{I}$-monotone fragment of AMC, indicating that the $\mathcal{I}$-monotone fragment suffices for most properties of interest.

The complexity upper bound is proved by a novel combination of techniques from the theory of infinite games, such as parity games and memoryless strategies, and techniques from cryptographic protocol analysis for reachability properties.

Related work. In [24, 17, 16], specific protocols have been analyzed w.r.t. game-theoretic security properties using the finite-state model checkers Murphi and MOCHA, where in [17, 16] several game-theoretic properties have been formulated in fair ATL. The disadvantage of using finite-state model checking is that the Dolev-Yao intruder has to be approximated and actions of dishonest principals have to be anticipated to some extent. The present work shows that fully automated analysis of game-theoretic security requirements is possible also w.r.t. a fine-grained infinite-state model and the standard Dolev-Yao intruder.

As already mentioned, a first decidability result for a specific security property, namely balance, was proved in [14] (see also [12] for a constraint-based algorithm). The present work considerably generalizes [14, 12] in terms of the security properties that can be checked, as here we consider a comprehensive class of security properties, expressed as $\mathcal{I}$-monotone AMC-formulas. Also, unlike [14, 12], the present work contains undecidability and (tight) complexity-theoretic re-

sults. Finally, following [7], [14, 12] use a model with interleaving semantics, and hence, unlike our real concurrent model, at every time only one principal may be active and a principal can only receive/send one message at a time.

In [9], Corin et al. proposed a procedure for deciding trace-based properties in a variant of LTL with only past temporal operators. While they cannot express game-theoretic security properties, it seems that the (trace-based) properties they have formulated in their logic can also be formulated in the $\mathcal{I}$-monotone fragment of AMC. Complexity-theoretic results are not provided by Corin et al. and they consider a model with interleaving semantics, rather than real concurrency.

Structure of this paper. In the next section, we recall the definition of concurrent game structures and AMC and present parity games for AMC-model checking. The security-specific model that we use, in particular the infinite-state concurrent game structures induced by protocols and the Dolev-Yao intruder are introduced in Section 3. The main results are summarized in Section 4. In Section 5, we illustrate the kind of properties that can be expressed in the fragment of AMC that we consider. The proofs of the main theorems are then presented in Section 6 to 8. We conclude in Section 9.

2 AMC and Parity Games

Following [1, 2], in this section we recall the definition of concurrent game structures and AMC. We also introduce parity games for AMC-model checking.

2.1 Concurrent Game Structures

Our definition of a concurrent game structure differs from the one in [2] in two aspects: First, the structures that we consider may have an *infinite* state space and in one state players may have an *infinite* number of possible moves. Second, while in [2] a move of a player is identified with a natural number, in our setting it is more convenient to allow arbitrary values; in the context of cryptographic protocol moves will be vertices of trees and terms.

We define concurrent game structures as follows. A *concurrent game structure (CGS)* is a tuple $S = \langle \Sigma, Q, \mathbb{P}, \pi, \Delta, \delta \rangle$ where

- Σ is a non-empty, finite set of *players*,
- Q is a (possibly infinite) set of *states*,
- $\mathbb{P}$ is a finite set of *propositional variables/propositions*,
- $\pi : Q \to 2^{\mathbb{P}}$ is a *labeling function* (which assigns every state to the set of propositions true in this state),
- Δ is a function which for each state $q \in Q$ and each player $a \in \Sigma$ returns a (possibly infinite) set $\Delta(q, a)$ of *moves* available at state q to player a.

  For $A \subseteq \Sigma$ and $q \in Q$, an (A, q)-*move* is a function c which maps every $a \in A$ to a move $c(a) \in \Delta(q, a)$. Given $A \subseteq \Sigma$ and a state q, we write $\Delta^A(q)$

for the set of (A, q)-moves. An (A, q)-move is called a *partial move* if $A \neq \Sigma$, and a *total move* if $A = \Sigma$.

- δ is a *transition function* which, for each state q and each total move $c \in \Delta^{\Sigma}(q)$, returns a state $\delta(q, c) \in Q$ (the state obtained when in state q all players simultaneously perform their moves according to c).

A *computation* of S is an infinite sequence $\lambda = q_0, q_1, \ldots$ of states such that for each $i \geq 0$, the state q_{i+1} is a *successor* of q_i, i.e., $q_{i+1} = \delta(q_i, c)$ for some total move $c \in \Delta^{\Sigma}(q_i)$. We call λ a q-computation if $q_0 = q$. We refer to the ith state q_i in λ by $\lambda[i]$, to the sequence $q_i, q_{i+1}, \ldots, q_j$ by $\lambda[i, j]$, and to the sequence $q_i, q_{i+1}, \ldots$ by $\lambda[i, \infty]$.

Let $c \in \Delta^{A}(q)$ and $c' \in \Delta^{A'}(q)$ for $A, A' \subseteq \Sigma$ and $q \in Q$ with $A \subseteq A'$. We write $c \sqsubseteq c'$ if $c(a) = c'(a)$ for every $a \in A$. For a state q, a set of players $A \subseteq \Sigma$, and an (A, q)-move $c \in \Delta^{A}(q)$, we say that a state $q' \in Q$ is a *c–successor of q* if there is a total move $c' \in \Delta^{\Sigma}(q)$ with $c \sqsubseteq c'$ and $q' = \delta(q, c')$.

2.2 AMC

Following [1, 2], we now recall the definition of the alternating μ-calculus (AMC).

Syntax of AMC-Formulas. An AMC-formula over the set $\mathbb{P}$ of propositions, the set $\mathcal{V}$ of variables, and the set Σ of players is one of the following:

- $p \in \mathbb{P}$,
- $X \in \mathcal{V}$,
- $\neg\varphi$ if φ is an AMC-formula,
- $\varphi_1 \vee \varphi_2$ if φ_1 and φ_2 are AMC-formulas,
- $\langle\langle A \rangle\rangle \bigcirc \varphi$ if $A \subseteq \Sigma$ and φ is an AMC-formula,
- $\mu X.\varphi$ if φ is an AMC-formula and all free occurrences of X (i.e., those that do not occur in a subformula of φ starting with μX) fall under an even number of negations.

We use the following common abbreviations: $[\![A]\!] \bigcirc \varphi = \neg\langle\langle A \rangle\rangle \bigcirc \neg\varphi$, $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$, and $\nu X.\varphi = \neg\mu X.\neg\varphi[X/\neg X]$ where $\varphi[X/\neg X]$ is obtained from φ by replacing every free occurrence of X in φ by $\neg X$ and vice versa. Using these abbreviations we can write every AMC-formula in negation normal form (also called positive normal form). An AMC-formula is in *negation normal form* if every negation symbol only occurs immediately in front of a proposition or a variable.

An AMC-formula is a *sentence* if it does not contain free variables, i.e., all variables are bounded by a fixed-point operator.

The *size* of an AMC-formula φ, denoted $|\varphi|$, is defined inductively in the obvious way.

Semantics of AMC-Formulas. To define the semantics of AMC-formulas, we first need some definitions and notations. Given a game structure $S =$

$\langle \Sigma, Q, \mathbb{P}, \pi, \Delta, \delta \rangle$, a valuation F is a function from the set of variables $\mathcal{V}$ to 2^Q, i.e., subsets of Q. For F, a variable X, and a set $M \subseteq Q$, we denote by $F[X := M]$ the valuation that maps X to M and agrees with F on all other variables.

An AMC-formula φ is interpreted as a mapping φ^S from valuations to state sets. Intuitively, $\varphi^S(F)$ denotes the set of states in which φ is satisfied under the valuation F in the structure S. The mapping φ^S is defined inductively as follows:

- $p^S(F) = \{q \in Q \mid p \in \pi(q)\}$ for $p \in \mathbb{P}$.
- $X^S(F) = F(X)$ for $X \in \mathcal{V}$.
- $(\neg\varphi)^S(F) = Q \setminus \varphi^S(F)$.
- $(\varphi_1 \vee \varphi_2)^S(F) = \varphi_1^S(F) \cup \varphi_2^S(F)$.
- $(\langle\!\langle A \rangle\!\rangle \bigcirc \varphi)^S(F) = \{q \in Q \mid$ there exists $c \in \Delta^A(q)$ such that for every $c' \in \Delta^\Sigma(q)$ with $c \sqsubseteq c'$ we have $\delta(q, c') \in \varphi^S(F)\}$.
- $(\mu X.\varphi)^S(F) = \bigcap\{M \subseteq Q | \varphi^S(F[X := M]) \subseteq M\}$, i.e., $(\mu X.\varphi)^S(F)$ is the least fixed-point of the function that maps $M \subseteq Q$ to $\varphi^S(F[X := M])$. (Note that this function is monotonic.)

Note that if φ is a sentence, then the interpretation of φ in the structure S is uniquely determined independently of a valuation function F. In fact, $\varphi^S(F) = \varphi^S(F')$ for all valuation functions F and F', i.e., φ^S is a constant mapping. We therefore simply write φ^S instead of $\varphi^S(F)$ for some F.

Given a state q of a CGS S and a sentence φ, we write

$$(S, q) \models \varphi$$

if $q \in \varphi^S$.

Deciding $(S, q) \models \varphi$ for a given finitely represented CGS S, a state q in S, and a sentence φ is an AMC-model checking problem. The main purpose of this paper is to study this problem for a class of CGSs induced by cryptographic protocols.

We note that AMC is more expressive than ATL* (and hence, ATL).

Theorem 1. *[2] AMC is more expressive than ATL*, and hence, provided a suitable set of propositional variables, also more expressive than fair ATL. The alternation-free fragment of AMC is more expressive than ATL.*

2.3 Parity Games and AMC-Model Checking

In this section, we first recall the definition of parity games and then, similar to the case of modal μ-calculus, associate with every CGS S, state q, and AMC-sentence φ a parity game in which player 0 has a winning strategy iff $(S, q) \models \varphi$.

Parity Games. Following [11], we now recall the definition of parity games.

A *parity game* G is a tuple (V, V_0, V_1, E, v_I, l) where V is a (possibly infinite) set of vertices partitioned into sets V_0 and V_1 (i.e., $V_0 \cup V_1 = V$ and $V_0 \cap V_1 = \emptyset$),

$v_I \in V$ (the initial vertex), $E \subseteq V \times V$ is a set of edges, and l is a coloring function from V into the set of colors $\{0, \ldots, m\}$, for some natural number m, such that (V, E) is a directed, leafless graph.

The parity game G is played by two players, player 0 and 1. A play of G starts by putting a token on vertex v_I. Now, if in a play a token is put on a vertex v (initially $v = v_I$) with $v \in V_i$ for $i \in \{0, 1\}$, then player i chooses a successor v' of v, i.e., $(v, v') \in E$, and moves the token to v'. Then, if $v' \in V_j$, for $j \in \{0, 1\}$, it is player j's turn to move the token to a successor of v', and so on. This continues forever. (Note that by definition of parity games, every vertex has a successor.) Formally, a *play* p is an infinite sequence $v_0, v_1, v_2, \ldots$, of vertices such that $v_0 = v_I$ and $(v_i, v_{i+1}) \in E$ for every $i \geq 0$. The play p is *winning for player 0* if the maximum color occurring infinitely often in p, i.e., the color $\max\{k \mid k$ occurs infinitely often in the sequence $l(v_0), l(v_1), \ldots\}$, is even. Otherwise, the play is *winning for player 1*.

A *strategy* f of player i is a function that for every finite prefix of a play, ending in a vertex $v \in V_i$, selects a successor v' of v, i.e., $(v, v') \in E$. A play $v_0, v_1, \ldots$ is *consistent with* f, if for each n such that $v_n \in V_i$, we have $v_{n+1} = f(v_0, v_1, \ldots, v_n)$. A strategy of player i is *winning*, if each play consistent with this strategy is winning for player i.

A strategy is *memoryless* (or *positional*), if it depends only on the last vertex, i.e., if $v_0, v_1, \ldots, v_n$ and $v'_0, v'_1, \ldots, v'_{n'}$ are prefixes of plays with $v_n = v'_{n'}$, then $f(v_0, v_1, \ldots, v_n) = f(v'_0, v'_1, \ldots, v'_{n'})$. We therefore often represent a memoryless strategy of player i by a function from V_i to V such that, for each $v \in V_i$, if $f(v) = v'$, then $(v, v') \in E$. A memoryless strategy f of player i in a parity game G induces a subgraph of (V, E) where all outgoing edges of vertices $v \in V_i$ are deleted except for the edge to $f(v)$. We call this graph a *strategy graph of player i* or the *strategy graph of player i induced by f*. Obviously, f is a winning strategy for player i iff all infinite paths in the induced strategy graph starting from the initial vertex v_I are winning for player i. We will sometimes assume that strategy graphs contain only vertices reachable from the initial vertex. Obviously, if such a graph is winning for player i, then each vertex v in this graph is winning for player i in the sense that each infinite path in this graph starting in v is winning for player i.

We summarize well-known and fundamental facts about parity games.

Fact 1 *[18, 20, 10] (see also [26]) Parity games are determined, i.e., either player* 0 *or player* 1 *has a winning strategy. The player who has a winning strategy has a memoryless one.*

Parity Games for AMC-Model Checking. In this section, we associate with every CGS S, state q_0 of S, and AMC-sentence φ in negation normal form a parity game $G^{\varphi}_{(S,q_0)}$ such that player 0 wins $G^{\varphi}_{(S,q_0)}$ iff $(S, q_0) \models \varphi$. Our construction follows that of modal μ-calculus (see, e.g., [25, 11]) and is similar to the one in [23]. However, instead of first turning φ into an equivalent alternating parity tree automaton and then using this tree automaton to obtain the parity game, we construct the parity game directly from φ and S.

Throughout the rest of this section, let $S = \langle \Sigma, Q, \mathbb{P}, \pi, \Delta, \delta \rangle$ be a concurrent game structure, $q_0 \in Q$ be a state in S, and φ be an AMC-sentence in negation normal form. We assume, w.l.o.g., that for each variable X in φ there is exactly one subformula of the form $\mu X.\psi$ or $\nu X.\psi$ in φ. (This can obviously be guaranteed by renaming variables.) We refer to this subformula by φ^X.

In what follows, we refer to subformulas of φ by *standard subformulas*. Now, given S, $q \in Q$, and φ, we define the set $\mathrm{Sub}^q_S(\varphi)$ to consist of the following (standard and non-standard) subformulas of φ:

(a) ψ for every standard subformula ψ of φ,

(b) $\Box_c\psi$ for every standard subformula $\langle\langle A\rangle\rangle\bigcirc\psi$ of φ and $c \in \Delta^A(q)$,

(c) $\Diamond_c\psi$ for every standard subformula $[\![A]\!]\bigcirc\psi$ of φ and $c \in \Delta^A(q)$.

We will call elements of $\mathrm{Sub}^q_S(\varphi)$ *subformulas of* φ where, as mentioned, the formulas in (a) are called *standard subformulas of* φ and those in (b) and (c) are called *nonstandard subformulas of* φ. Note that $\Diamond_c$ and $\Box_c$ occur only as top symbols of subformulas; they are not nested.

Now, the parity game $G^{\varphi}_{(S,q_0)} = (V, V_0, V_1, E, v_I, l)$ for S, q_0, and φ is defined as follows (see below for a brief discussion of the differences to the construction for the model μ-calculus): The set V of vertices consists of all tuples of the form (q, ψ) where $q \in Q$ and $\psi \in \mathrm{Sub}^q_S(\varphi)$. The initial vertex v_I is (q_0, φ). The set V_0 consists of vertices of the form (q, ψ) where $q \in Q$ and ψ is of one of the following forms:

$$\psi' \vee \psi'', \qquad \langle\langle A\rangle\rangle\bigcirc\psi', \qquad \Diamond_c\psi'.$$

All remaining vertices belong to V_1. The set E of edges is the smallest set satisfying the following conditions:

- $(\ (q,p)\ ,\ (q,p)\ ) \in E$ for every $(q,p) \in V$.
- $(\ (q,\neg p)\ ,\ (q,\neg p)\ ) \in E$ for every $(q,\neg p) \in V$.
- $(\ (q,X)\ ,\ (q,\varphi^X)\ ) \in E$ for every $(q,X) \in V$.
- $(\ (q,\mu X.\psi)\ ,\ (q,\psi)\ ) \in E$ for every $(q,\mu X.\psi) \in V$.
- $(\ (q,\nu X.\psi)\ ,\ (q,\psi)\ ) \in E$ for every $(q,\nu X.\psi) \in V$.
- $(\ (q,(\psi \vee \psi'))\ ,\ (q,\psi)\ ) \in E$ and $(\ (q,(\psi \vee \psi'))\ ,\ (q,\psi')\ ) \in E$ for every $(q,(\psi \vee \psi')) \in V$.
- $(\ (q,(\psi \wedge \psi'))\ ,\ (q,\psi)\ ) \in E$ and $(\ (q,(\psi \wedge \psi'))\ ,\ (q,\psi')\ ) \in E$ for every $(q,(\psi \wedge \psi')) \in V$.
- $(\ (q,\langle\langle A\rangle\rangle\bigcirc\psi)\ ,\ (q,\Box_c\psi)\ ) \in E$ for every $(q,\langle\langle A\rangle\rangle\bigcirc\psi) \in V$ and $c \in \Delta^A(q)$.
- $(\ (q,[\![A]\!]\bigcirc\psi)\ ,\ (q,\Diamond_c\psi)\ ) \in E$ for every $(q,[\![A]\!]\bigcirc\psi) \in V$ and $c \in \Delta^A(q)$.
- $(\ (q,\Box_c\psi)\ ,\ (q',\psi)\ ) \in E$ for every $(q,\Box_c\psi) \in V$ and c-successor q' of q.
- $(\ (q,\Diamond_c\psi)\ ,\ (q',\psi)\ ) \in E$ for every $(q,\Diamond_c\psi) \in V$ and c-successor q' of q.

Let $\|\varphi\|$ be the depth of φ when φ is viewed as a syntax tree. The coloring function l is defined as follows:[4] The color of a state $s = (q, \psi)$ is defined as follows:

[4] Using the alternation depth of formulas, one can obtain a coloring function that assigns smaller colors. This is useful to achieve more efficient algorithms. However,

– $l(s) = 1$, if $\psi = p$ and $p \notin \pi(q)$ or $\psi = \neg p$ and $p \in \pi(q)$,
– $l(s) = 2\|\psi\|$, if $\psi = \nu X.\psi'$ for some ψ',
– $l(s) = 2\|\psi\| + 1$, if $\psi = \mu X.\psi'$ for some ψ', and
– $l(s) = 0$ otherwise.

The above definition of $G^{\varphi}_{(S,q_0)}$ is similar to the case of modal μ-calculus with the following difference: In case of the modal μ-calculus, when a play reaches a position (q, ψ) with ψ of the form $\Box\psi'$ or $\Diamond\psi'$, then one of the players chooses a successor q' of q and the play continues with (q', ψ). In case of AMC, we have a family of modal operators $\langle\langle A\rangle\rangle\circ$ and $[\![A]\!]\circ$ and when a play reaches a position $(q, \langle\langle A\rangle\rangle\circ\psi')$ or $(q, [\![A]\!]\circ\psi')$ then we use an intermediate state before a successor of q is chosen: first one of the players moves to a position of the form $(q, \Box_c\psi')$ or $(q, \Diamond_c\psi')$, and then the opponent chooses a successor q' of q and the play continues with (q', ψ').

Similar to the case of the modal μ-calculus (see, e.g., [25, 11]), one shows the following proposition:

Proposition 1. *For S, q_0, and φ as above we have that $(S, q_0) \models \varphi$ iff player 0 has a (memoryless) winning strategy in the parity game $G^{\varphi}_{(S,q_0)}$.*

3 Our Protocol and Intruder Model

We now introduce our protocol and intruder model. Similar to [15], we consider a real concurrent communication model in which principals (including the intruder) may take actions at the same time and may receive/send several messages at the same time from/to different principals. Principals are connected via different kinds of channels: network and resilient channels. Instead of the term "resilient channel", we often use the term "secure channel". While network channels are completely controlled by the intruder, secure channels are not. In particular, the intruder may not be able to delay or modify messages sent over such a channel. We will consider two types of secure channels. Those that directly link to principals (direct secure channels) and those that are buffered (scheduled secure channels). While messages sent over direct secure channels are immediately delivered, messages sent over scheduled secure channels are first written into a buffer. The buffer is an independent agent which can follow its own strategy in delivering messages; it can for example team up with an honest principal or the intruder. Whether and with whom such a buffer collaborates depends on the security property considered, as specified by an AMC-formula.

While conceptually the model presented here and the one presented in [15] are quite similar, the presentation and level of detail varies in the following main points: First, in [15], no specific formalism for describing honest protocol participants was presented. Such participants could be arbitrary I/O components (in particular, arbitrary interactive, non-deterministic Turing machines). However,

for the complexity results shown in this paper the coloring function employed here is sufficient.

since in the present work we are interested in decidability results, we need to be more precise about the representation and the kind of computation honest protocol participants are allowed to perform. As defined below, such participants will be modeled by certain edge-labeled trees. Second, in [15] we considered a general communication model and described how a system of I/O components runs. Then, protocol runs and attacks were described in terms of such systems where every entity (honest protocol participants, the intruder, scheduled secure channels) was modeled as an I/O component. In the present work, we do not consider systems of I/O components but model protocol runs and attacks directly in terms of concurrent game structures.

In what follows, we define i) terms and messages, ii) how the intruder can derive new messages from a given set of messages, iii) principals and protocols, and iv) concurrent game structures which describe the run of a protocol along with the intruder.

3.1 Terms and Messages

Let $\mathcal{V}$ be a finite set of variables, $\mathcal{A}$ be a finite set of *atoms* (*atomic messages*), $\mathbb{K}$ be a finite set of *public* and *private keys*, and $\mathcal{A}_I$ be an infinite set of *intruder atoms*. These sets are assumed to be pairwise disjoint. Typically, the set $\mathcal{A}$ contains names of principals, atomic symmetric keys, and nonces (i.e., random numbers generated by principals). The set $\mathbb{K}$ is partitioned into a set $\mathbb{K}_{pub}$ of public keys and a set $\mathbb{K}_{priv}$ of private keys. There is a bijective mapping $\cdot^{-1} : \mathbb{K} \to \mathbb{K}$ which assigns to every public key the corresponding private key and to every private key its corresponding public key. We note that we will allow non-atomic symmetric keys as well. The atoms in $\mathcal{A}_I$ are the nonces, symmetric keys, etc. the intruder may generate.

The set $\mathcal{T}$ of *terms* is defined as follows:

$$\mathcal{T} ::= \mathcal{V} \mid \mathcal{A} \mid \mathcal{A}_I \mid \langle \mathcal{T}, \mathcal{T} \rangle \mid \{\mathcal{T}\}^s_{\mathcal{T}} \mid \{\mathcal{T}\}^a_{\mathbb{K}_{pub}} \mid \mathsf{hash}(\mathcal{T}) \mid \mathsf{sig}(\mathbb{K}_{pub}, \mathcal{T})$$

Terms without variables (i.e., ground terms) are called *messages*. The set of messages is denoted by $\mathcal{M}$. As usual, $\langle t, t' \rangle$ is the pairing of t and t', the term $\{t\}^s_{t'}$ stands for the symmetric encryption of t by t' (note that the key t' may be any term), $\{t\}^a_k$ is the asymmetric encryption of t by k, the term $\mathsf{hash}(t)$ stands for the hash of t, and $\mathsf{sig}(k, t)$ is the signature on t (generated using k^{-1}) which can be verified with the public key k. One could add further cryptographic primitives, such as *private contract signatures (PCSs)* as in [14]. While all results presented in this paper would, for example, carry over to the case with PCSs, for simplicity of presentation, this and other cryptographic primitives are not considered.

We define $\mathcal{T}_\circ = \mathcal{T} \cup \{\circ\}$ and $\mathcal{M}_\circ = \mathcal{M} \cup \{\circ\}$ where '$\circ$' is a new symbol which stands for 'no message'. This symbol will be used in case there is no message on a channel.

A *substitution* σ is a mapping from variables to terms where the domain $\mathrm{dom}(\sigma) = \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$ of σ is required to be finite and $\sigma(x) \in \mathcal{M}$ for

every $x \in \mathrm{dom}(\sigma)$. Given two substitutions σ and σ' with disjoint domains, their union $\sigma \cup \sigma'$ is defined in the obvious way. Given a term t, the term $t\sigma$ is obtained from t by simultaneously substituting each variable x occurring in t by $\sigma(x)$.

3.2 Derivation of Messages

Given a set $\mathcal{K}$ of messages, the (infinite) set $d(\mathcal{K})$ of messages the intruder can derive from $\mathcal{K}$ is the smallest set satisfying the following conditions with $m, m' \in \mathcal{M}$:

1. $\mathcal{K} \subseteq d(\mathcal{K})$.
2. $\circ \in d(\mathcal{K})$.
3. *Composition and decomposition*: If $m, m' \in d(\mathcal{K})$, then $\langle m, m' \rangle \in d(\mathcal{K})$. Conversely, if $\langle m, m' \rangle \in d(\mathcal{K})$, then $m \in d(\mathcal{K})$ and $m' \in d(\mathcal{K})$.
4. *Symmetric encryption and decryption*: If $m, m' \in d(\mathcal{K})$, then $\{m\}^s_{m'} \in d(\mathcal{K})$. Conversely, if $\{m\}^s_{m'} \in d(\mathcal{K})$ and $m' \in d(\mathcal{K})$, then $m \in d(\mathcal{K})$.
5. *Asymmetric encryption and decryption*: If $m \in d(\mathcal{K})$ and $k \in d(\mathcal{K}) \cap \mathbb{K}_{pub}$, then $\{m\}^a_k \in d(\mathcal{K})$. Conversely, if $\{m\}^a_k \in d(\mathcal{K})$ and $k^{-1} \in d(\mathcal{K}) \cap \mathbb{K}_{priv}$, then $m \in d(\mathcal{K})$.
6. *Hashing*: If $m \in d(\mathcal{K})$, then $\mathsf{hash}(m) \in d(\mathcal{K})$.
7. *Signing*: If $m \in d(\mathcal{K}), k^{-1} \in d(\mathcal{K}) \cap \mathbb{K}_{priv}$, then $\mathsf{sig}(k, m) \in \mathcal{K}$. (The signature contains the public key but can only be generated if the corresponding private key is known.)
8. *Generating fresh constants*: $\mathcal{A}_I \subseteq d(\mathcal{K})$.

3.3 Channels, Principals, and Protocols

We denote by $\mathcal{P}$ the finite set of all principals. This set is partitioned into the set $\mathcal{H}$ of *honest* and the set $\mathcal{D}$ of *dishonest* principals. All dishonest principals will be subsumed by the intruder. The behavior of honest principals will be specified by certain trees (see below). Protocols will basically be defined by a set of such trees, specifying the behavior of all honest principals participating in a protocol run. First, we have to define how principals are connected via channels.

Channels and Multi Terms. We consider three types of communication channels between principals (including the intruder): (1) *network channels*, (2) *direct secure channels*, and (3) *scheduled secure channels*. Network channels are controlled by the intruder, i.e., every message sent on a network channel by an honest principal is immediately delivered to the intruder and every message received from a network channel was sent by the intruder (who impersonates some honest or dishonest principal). A direct secure channel is a direct link between principals, i.e., every message sent on such a channel by some principal to another principal will immediately be delivered to the latter principal without intervention by the intruder. Messages sent via a scheduled secure channel will first be sent to a buffer before they are delivered to the intended recipient. Such a buffer

is an independent player in the concurrent games structures that we consider and may be controlled or may team up with (honest or dishonest) principals or other scheduled secure channels. This will be specified by AMC-formulas.

A network channel from a principal a to a principal b such that $a \neq b$ and not both a and b are dishonest will be denoted by $\mathsf{net}(a, b)$. Similarly, we use $\mathsf{dir}(a, b)$ and $\mathsf{sch}(a, b)$ to refer to direct and scheduled secure channels from a to b, respectively. The set of all the channels will be denoted by $\mathcal{C}$.

For sets $A, B \subseteq \mathcal{P}$ of principals, we define $\mathsf{Net}(A, B) = \{\mathsf{net}(a, b) \mid a \in A, b \in B, a \neq b, \text{ and } (a \in \mathcal{H} \text{ or } b \in \mathcal{H})\}$. Similarly, we define $\mathsf{Dir}(A, B)$ and $\mathsf{Sch}(A, B)$ for direct and scheduled secure channels. We define $\mathcal{C}(A, B) = \mathsf{Net}(A, B) \cup \mathsf{Dir}(A, B) \cup \mathsf{Sch}(A, B)$. We will write, for example, $\mathsf{Net}(a, B)$ instead of $\mathsf{Net}(\{a\}, B)$.

For a set $C \subseteq \mathcal{C}$, we call a mapping $\mathbf{r} : C \to \mathcal{T}_\circ$ a *multi term* and a mapping $\mathbf{r} : C \to \mathcal{M}_\circ$ a *multi message*. We denote by $\mathsf{ch}(\mathbf{m})$ and $\mathsf{ch}(\mathbf{r})$ the domain C of $\mathbf{m}$ and $\mathbf{r}$, respectively, and by $\mathcal{V}(\mathbf{r})$ the set of variables occurring in the range of $\mathbf{r}$, i.e., in the set $\{t \mid \mathbf{r}(c) = t \text{ for some } c \in C\}$. If σ is a substitution, we denote by $\mathbf{r}\sigma$ the multi term obtained by substituting every variable $x \in \mathcal{V}(\mathbf{r})$ occurring in $\mathbf{r}$ by $\sigma(x)$, i.e., $\mathbf{r}\sigma(c) = \mathbf{r}(c)\sigma$ for every $c \in C$.

Let $\mathbf{m}$ be a multi message, $\mathbf{r}$ be a multi term, and σ be a substitution with domain $\mathcal{V}(\mathbf{r})$. We say that $\mathbf{m}$ *matches with* $\boldsymbol{r}$ *by* σ, if $\mathsf{ch}(\mathbf{r}) \subseteq \mathsf{ch}(\mathbf{m})$ and $\mathbf{m}(c) = \mathbf{r}(c)\sigma$ for each $c \in \mathsf{ch}(r)$. We say that $\mathbf{m}$ *matches with* $\boldsymbol{r}$, if there is a substitution σ such that $\mathbf{m}$ matches with $\mathbf{r}$ by σ.

Honest Principals. We now define honest principals; more precisely, we should say 'instances of honest principals' since a principal might run several copies of a protocol in possibly different roles. Informally speaking, an honest principal is defined by a finite edge-labeled tree which describes the behavior of this principal in a protocol run. Each edge of such a tree is labeled by a rule which describes the receive-send action that is performed when the principal takes this edge in a run of the protocol. As mentioned above, in *one* receive-send action a principal may receive/send several messages on different channels. The trees that we consider may have self-loops. These allow a principal to stay in the same state. When a principal carries out a protocol, it traverses its tree, starting at the root. In every node, the principal takes its current input (on all channels the principal has access to or wants to read), chooses one of the edges leaving the node such that the current inputs match with the left-hand side of the rule the edge is labeled with, sends out (possibly different) messages on (possibly different) channels as determined by the right-hand side of the rule, and moves to the node the chosen edge leads to. Edges have priorities which influence which edge may be taken in case several edges are applicable. However, if several edges with the same priority can be taken, one such edge is picked non-deterministically. Formally, principals are defined as follows:

For sets $C, D \subseteq \mathcal{C}$, we call $\mathbf{r} \Rightarrow \mathbf{s}$ with $\mathbf{r} : C \to \mathcal{T}_\circ$ and $\mathbf{s} : D \to \mathcal{T}_\circ$ a (C, D)–*rule*. For an honest principal $a \in \mathcal{H}$, an a–*rule* is a (C, D)–rule with $C \subseteq \mathcal{C}(\mathcal{P}, a)$ and $D \subseteq \mathcal{C}(a, \mathcal{P})$. If σ is a substitution and $R = (\mathbf{r} \Rightarrow \mathbf{s})$ is a rule, we write $R\sigma$

to denote the rule obtained by substituting every variable x occurring in R by $\sigma(x)$, i.e., $R\sigma = (\mathbf{r}\sigma \Rightarrow \mathbf{s}\sigma)$.

Let $a \in \mathcal{H}$ be an honest principal. Its behavior is specified by what we call an a-instance (or simply principal). An *a-instance* (*principal*) is defined by a finite tree $P = (V, E, r, \ell_p, \ell)$ where V is the set of vertices, E is the set of edges, $r \in V$ is the root of the tree, and ℓ_p maps every edge $e \in E$ of P to a natural number, the *priority of this edge*. The labeling function ℓ maps every edge $e = (v, v') \in E$ of P to an a-rule $\ell(e)$ in such a way that every variable occurring in $\mathcal{V}(\mathbf{s})$ with $\ell(e) = (\mathbf{r} \Rightarrow \mathbf{s})$ also occurs on the left-hand side of $\ell(e)$, i.e., in $\mathcal{V}(\mathbf{r})$, or on the left-hand side of a rule on the path from the root r to v. In other words, every variable occurring on the right-hand side of a rule also occurs on the left-hand side of this or a preceding rule. Nodes of P may have *self-loops*, i.e., P may contain edges of the form $e = (v, v)$ for $v \in V$. In that case, we require that for $\ell(e) = (\mathbf{r} \Rightarrow \mathbf{s})$ the domains of $\mathbf{r}$ and $\mathbf{s}$ are empty, i.e., $\mathsf{ch}(\mathbf{r}) = \emptyset$ and $\mathsf{ch}(\mathbf{s}) = \emptyset$. In other words, when performing a self-loop, a principal neither reads nor writes messages from/onto a channel.

For an a-instance P, we denote by $\mathsf{ch}(P)$ the set of all the channels used by P, i.e., $\mathsf{ch}(P)$ consists of those channels c for which there exists an edge in P labeled with a rule of the form $\mathbf{r} \Rightarrow \mathbf{s}$ such that $c \in \mathsf{ch}(\mathbf{r})$ or $c \in \mathsf{ch}(\mathbf{s})$.

Protocols. A *protocol* is a tuple $Pr = (\mathcal{H}, \mathcal{D}, \mathcal{K}, \{P_a\}_{a \in \mathcal{H}})$ where $\mathcal{H}$ and $\mathcal{D}$ are sets of honest and dishonest principals, respectively, P_a is an a-instance for each $a \in \mathcal{H}$, and $\mathcal{K}$ is the *initial intruder knowledge*, i.e., a finite set of messages. W.l.o.g., we assume that the set of vertices of the trees P_a, $a \in \mathcal{H}$, are pairwise disjoint. For a protocol Pr, we denote by $\mathsf{ch}(Pr)$ the set of channels used in Pr, i.e. the set of all channels c such that $c \in \mathsf{ch}(P_a)$, for some $a \in \mathcal{H}$. The size of Pr, denoted by $|Pr|$ is defined according to some standard representation of Pr.

3.4 Example: the ASW Two-Party Contract-signing Protocol

Two illustrate the definition of honest principals and protocols introduced above, we specify the ASW protocol [4] in our model. First, we provide an informal overview of the protocol.

We write $\mathsf{sig}[k, m]$ as abbreviation for $\langle m, \mathsf{sig}(k, m)\rangle$ and sometimes write $\langle m_1, \ldots, m_n\rangle$ instead of $\langle m_1, \langle m_2, \langle \cdots \langle m_{n-1}, \rangle\rangle\rangle\rangle$. We denote the public or verification key of a principal A by k_A.

The ASW protocol enables two principals A (the originator) and B (the responder) to obtain each other's signature on a previously agreed contractual text $\mathsf{contract}$ with the help of a trusted third party (TTP) T, which however is only invoked in case of problems. In other words, the ASW protocol is an optimistic two-party contract-signing protocol.

There are two kinds of valid contracts: the standard contract

$$\langle \mathsf{sig}[k_A, m_A], N_A, \mathsf{sig}[k_B, m_B], N_B\rangle$$

and the replacement contract

$$\mathsf{sig}[k_T, \langle \mathsf{sig}[k_A, m_A], \mathsf{sig}[k_B, m_B]\rangle],$$

where $m_A = \langle k_A, k_B, k_T, \mathsf{contract}, \mathsf{hash}(N_A)\rangle$, $m_B = \langle \mathsf{sig}[k_A, m_A], \mathsf{hash}(N_B)\rangle$, and N_A and N_B are nonces.

The ASW protocol consists of three subprotocols: the exchange, abort, and resolve protocol. These subprotocols are explained next.

Exchange protocol. The basic idea of the *exchange protocol* is that A first indicates her interest to sign the contract. To this end, she sends to B the message $\mathsf{sig}[k_A, m_A]$ as defined above, where N_A is a nonce generated by A. By sending this message, A "commits" to signing the contract. Then, similarly, B indicates his interest to sign the contract by generating a nonce N_B and sending the message $\mathsf{sig}[k_B, m_B]$ to A. Finally, first A and then B reveal N_A and N_B, respectively.

Abort protocol. If, after A has sent her first message, B does not respond, A may contact T to abort, i.e., A runs the abort protocol with T. Note that A may wait as long as she wants before contacting T. (In our formal model, this is modeled as a non-deterministic action of A.) In the abort protocol, A first sends the message $a_A = \mathsf{sig}[k_A, \langle \mathsf{aborted}, \mathsf{sig}[k_A, m_A]\rangle]$. If T has not received a resolve request before (see below), then T sends back to A the *abort token* $a_T = \mathsf{sig}[k_T, \langle \mathsf{aborted}, a_A\rangle]$. Otherwise (if T received a resolve request, which in particular involves the messages $\mathsf{sig}[k_A, m_A]$ and $\mathsf{sig}[k_B, m_B]$ from above), it sends the *replacement contract* $r_T = \mathsf{sig}[k_T, r]$ to A with $r = \langle \mathsf{sig}[k_A, m_A], \mathsf{sig}[k_B, m_B]\rangle$.

Resolve protocol. If, after A has sent the nonce N_A, B does not respond, A may contact T to resolve, i.e., A runs the resolve protocol with T. Again, A may wait for as long as she wants before contacting T. In the resolve protocol, A sends the message r to T. If T has not sent out an abort token before, then T returns the replacement contract r_T, and otherwise T returns the abort token a_T. Analogously, if, after B has sent his commitment to sign the contract, A does not respond, B may contact T to resolve, i.e., B runs the resolve protocol with T similarly to the case for A. Note that contacting T is again a non-deterministic action of B.

We note that the communication with T (for both A and B) is carried out over secure channels.

Figure 1 and 2 present the formal specifications P_A and P_T of A and T, respectively. The specification of B can be defined similarly. The specification of the ASW protocol for the case that A and T are honest but B is dishonest (and hence, B's behavior is determined by the intruder) is the tuple $Pr_{ASW} = (\{A, T\}, \{B\}, \{A, B, T, k_A, k_B, k_B^{-1}, k_T\}, \{P_A, P_T\})$. In Figure 1 and 2 the communication between A and T is modeled by scheduled secure channels and the communication between B and T by direct secure channels. Note that allowing (dishonest) B direct communication with T increases his power. To check certain properties of this protocol, it is useful to add a "watch dog" W as another honest principal: W checks whether the intruder (B) has a standard or

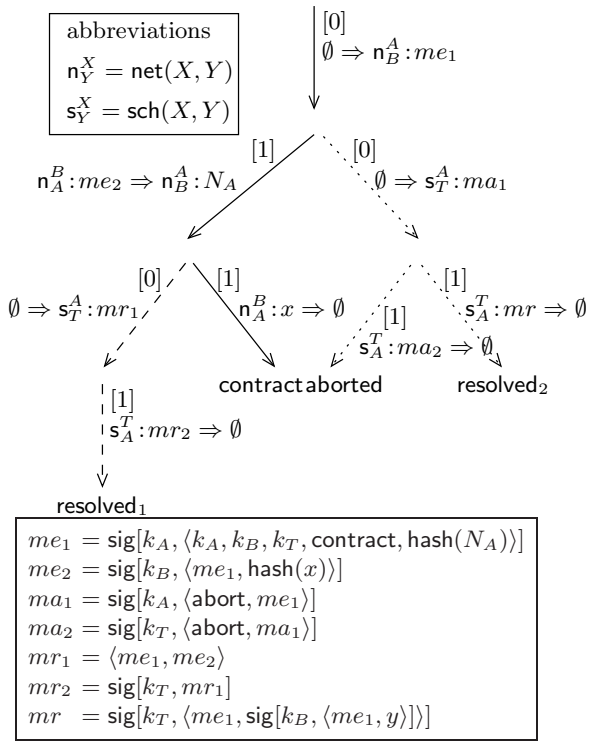
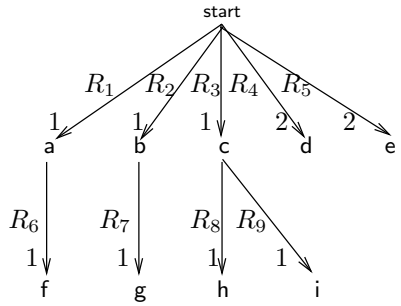


Fig. 1. Honest Alice running the ASW protocol as initiator with Bob. Even though not drawn, every vertex in the tree has a self-loop with priority 0.

replacement contract (as defined above). More precisely, W waits to receive the standard or replacement contract (as defined above) on a network channel from B and if it receives such a contract, moves to a vertex called, say `Bhascontract`; W ignores all other messages it receives.

3.5 The Concurrent Game Structure Induced by a Protocol

We now introduce the concurrent game structure induced by a protocol. The players involved are the honest principals, the scheduled secure channels, and the Dolev-Yao intruder (who subsumes the dishonest principals). The concurrent game structure describes what moves these players can take in every state and what effect these moves have. Roughly speaking, the possible moves of an honest principal are those edges (receive-send actions) that leave the current vertex and that can be applied given the current input and the priority on the edges. As a result of taking such an edge, the principal writes output on (zero, one, or more) channels. A scheduled secure channel is represented by a sequence of messages, the messages on this channel. In a move it decides whether or not to deliver the first message in this sequence. (Alternatively, in case one would like to model



rules:

R_1 : $\mathsf{sch}(A,T) : ma^a \Rightarrow \mathsf{sch}(T,A) : mak^a$
R_2 : $\mathsf{sch}(A,T) : mr^b \Rightarrow \mathsf{sch}(T,A) : mrc^b$
R_3 : $\mathsf{dir}(B,T) : mr^c \Rightarrow \mathsf{dir}(T,B) : mrc^c$
R_4 : $\mathsf{sch}(A,T) : mr^d$, $\mathsf{dir}(B,T) : mr^d \Rightarrow \mathsf{sch}(T,A) : mrc^d$, $\mathsf{dir}(T,B) : mrc^d$
R_5 : $\mathsf{sch}(A,T) : ma^e$, $\mathsf{dir}(B,T) : mr^e \Rightarrow \mathsf{sch}(T,A) : mak^e$, $\mathsf{dir}(T,B) : mak^e$
R_6 : $\mathsf{dir}(B,T) : mr^a \Rightarrow \mathsf{dir}(T,B) : mak^a$
R_7 : $\mathsf{dir}(B,T) : mr^b \Rightarrow \mathsf{dir}(T,B) : mrc^b$
R_8 : $\mathsf{sch}(A,T) : mr^c \Rightarrow \mathsf{sch}(T,A) : mrc^c$
R_9 : $\mathsf{sch}(A,T) : ma^c \Rightarrow \mathsf{sch}(T,A) : mrc^c$

messages:

$mr^u = \langle me_1^u, me_2^u \rangle$
$me_1^u = \mathsf{sig}[k_A, \langle k_A, k_B, k_T, \mathsf{contract}, x_u \rangle]$
$me_2^u = \mathsf{sig}[k_B, \langle me_1^u, y_u \rangle]$
$ma^u = \mathsf{sig}[k_A, \langle \mathsf{abort}, me_1^u \rangle]$
$mak^u = \mathsf{sig}[k_T, \langle \mathsf{abort}, me_1^u \rangle]$
$mrc^u = \mathsf{sig}[k_T, \langle me_1^u, me_2^u \rangle]$

Fig. 2. This is the tree model of participant TTP. Even though not drawn, every vertex in the tree has a self-loop with priority 0. The rules R_1 to R_5 are needed to distinguish between whether the TTP receives in one step (1) one abort message from A, (2) one resolve message from A, (3) one resolve message from B, (4) a resolve message from both A and B, or (5) an abort message from A and a resolve message from B.

secure channels that do not preserve the order of messages, one could allow secure channels to pick one of the messages in their sequence.) The intruder has an infinite number of possible moves. He can write a message on all channels that he controls, where for every such channel he can pick one of the (infinitely many possible) messages that he can derive in the current state. Direct secure channels will always immediately deliver the message written to them, and therefore, they do not need to be modeled as players. Before providing the formal definition of the concurrent game structure, we need to introduce some notation.

For a principal P and a node v of P, we write $P{\downarrow}v$ to denote the subtree of P rooted at v. If σ is a substitution, we write $P\sigma$ for the principal obtained from P by applying σ to every rule occurring in P.

For a protocol Pr, let $C = \mathcal{C}(\mathcal{P}, a) \cap \mathsf{ch}(Pr)$ and $C' = \mathcal{C}(a, \mathcal{P}) \cap \mathsf{ch}(Pr)$. For $\mathbf{m} : C \to \mathcal{M}_\circ$, $\mathbf{m}' : C' \to \mathcal{M}_\circ$, an a-instance $P = (V, E, v_0, \ell_p, \ell)$, and an

a-instance P', we write $(\mathbf{m}, P) \overset{v}{\mapsto} (\mathbf{m}', P')$ if $v \in V$, $(v_0, v) \in E$, $\ell(v_0, v)$ is of the form $\mathbf{r} \Rightarrow \mathbf{s}$, and there exists a substitution σ with $\mathrm{dom}(\sigma) = \mathcal{V}(\mathbf{r})$ such that

- $P' = (P \downarrow v)\sigma$,
- $\mathbf{m}$ matches with $\mathbf{r}$ by σ,
- for all $v' \in V$ such that $(v_0, v') \in E$, $\ell(v_0, v') = (\mathbf{r}' \Rightarrow \mathbf{s}')$ and $\mathbf{m}$ matches with $\mathbf{r}'$ we have that $\ell_p(v_0, v) \geq \ell_p(v_0, v')$, and
- $\mathbf{m}'$ matches with $\mathbf{s}$ by σ and $\mathbf{m}'(c) = \circ$ for each $c \in C' \setminus \mathsf{ch}(\mathbf{s})$.

In what follows, let $Pr = (\mathcal{H}, \mathcal{D}, \mathcal{K}_0, \{P_a^0\}_{a \in \mathcal{H}})$ be a protocol. Let $m \in \mathcal{M}_\circ$ and $t \in \mathcal{T}_\circ$. Let $\mathcal{IC} = (\mathsf{Net}(\mathcal{P}, \mathcal{H}) \cup \mathsf{Dir}(\mathcal{D}, \mathcal{H}) \cup \mathsf{Sch}(\mathcal{D}, \mathcal{H})) \cap \mathsf{ch}(Pr)$ be the set of all channels in Pr the intruder may write to and let $\mathcal{SC} = \mathsf{Sch}(\mathcal{P}, \mathcal{P}) \cap \mathsf{ch}(Pr)$ be the set of all scheduled secure channels in Pr.

We are now ready to define the concurrent game structure induced by Pr. The *concurrent game structure* $S = S_{Pr} = \langle \Sigma, Q, \mathbb{P}, \pi, \Delta, \delta \rangle$ *induced by* Pr is defined as follows:

- The set of players Σ is $\mathcal{H} \cup \mathcal{SC} \cup \{\mathcal{I}\}$.
- The set of states Q consists of tuples of the form $(\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \overline{s})$ where
  - $\mathcal{K}$ is a finite set of messages (the current intruder knowledge),
  - $\overline{P}$ is a family $\{P_a\}_{a \in \mathcal{H}}$ of a-instances P_a for every $a \in \mathcal{H}$,
  - $\overline{\mathbf{m}}$ is a family $\{\mathbf{m}_a\}_{a \in \mathcal{H}}$ of multi messages $\mathbf{m}_a : \mathcal{C}(\mathcal{P}, a) \cap \mathsf{ch}(Pr) \to \mathcal{M}_\circ$ for every $a \in \mathcal{H}$ (the current input to a).[5]
  - $\overline{s}$ is a family $\{(s_c, d_c)\}_{c \in \mathcal{SC}}$ of tuples (s_c, d_c) where $s_c \in \mathcal{M}^*$ is a sequence of messages, the messages on c, and $d_c \in \{\mathsf{delivered}, \overline{\mathsf{delivered}}\}$, indicating whether or not c delivered a message in the previous step, for every $c \in \mathcal{SC}$.
- The set $\mathbb{P}$ of propositional variables contains a propositional variable p_a for each constant $a \in \mathcal{A}$, a propositional variable p_v for each vertex v of a principal specified in Pr (recall that different principals have different sets of vertices), and propositional variables empty_c and $\mathsf{delivered}_c$ for every $c \in \mathcal{SC}$.
- The evaluation π of the propositional variables in a state $q = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \overline{s})$ is defined as follows:

$$\begin{aligned}\pi(q) = \{&p_a \mid a \in d(\mathcal{K}) \cap \mathcal{A}\} \cup \\ &\{p_v \mid v \text{ is the root of } P_a \text{ for some } a \in \mathcal{H}\} \cup \\ &\{\mathsf{empty}_c \mid c \in \mathcal{SC}, s_c = \varepsilon\} \cup \\ &\{\mathsf{delivered}_c \mid c \in \mathcal{SC}, d_c = \mathsf{delivered}\},\end{aligned}$$

  i.e., p_a is true in q if the intruder can derive a from its current knowledge, p_v is true in q if some honest principal is in vertex v, empty_c is true if c currently does not contain any messages, and $\mathsf{delivered}_c$ is true if $d_c = \mathsf{delivered}$.

[5] Messages which are to be delivered to the intruder or scheduled secure channels will immediately be added to the intruder's knowledge and the secure channel buffer, respectively.

– For $q = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \overline{s}) \in Q$ as above and a player $\alpha \in \Sigma$, we define $\Delta(q, \alpha)$ as follows:
  - If $\alpha \in \mathcal{H}$ with $P_\alpha = (V, E, v_0, \ell_p, \ell)$, the set $\Delta(q, \alpha)$ consists of all $v \in V$ such that $(\mathbf{m}_\alpha, P_\alpha) \overset{v}{\mapsto} (\mathbf{m}, P'_\alpha)$ for some $\mathbf{m}$ and P'_α, i.e., α can take one of the edges leaving the current vertex. If this set is empty, we define $\Delta(q, \alpha) = \{v_0\}$ (see below for an explanation).
  - If $\alpha = \mathcal{I}$, the set $\Delta(q, \alpha)$ consists of all $\mathbf{m} : \mathcal{IC} \to \mathcal{M}_\circ$ such that $\mathbf{m}(c) \in d(\mathcal{K})$ for every $c \in \mathcal{IC}$, i.e., the intruder can send messages on the channels that he controls, where the messages are derived from his current knowledge.
  - If $\alpha \in \mathcal{SC}$, then $\Delta(q, \alpha) = \{0\}$, in case $s_\alpha = \varepsilon$ (i.e., s_α is empty), and $\Delta(q, \alpha) = \{0, 1\}$, otherwise ($1 \mathrel{\widehat{=}}$ "α delivers the next message in the sequence" and $0 \mathrel{\widehat{=}}$ "α does not deliver a message").

– For $q = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \overline{s}) \in Q$ and a total move $\gamma \in \Delta_q^\Sigma$, we define the γ-successor $\delta(q, \gamma)$ of q to be the state $(\mathcal{K}', \overline{P}', \overline{\mathbf{m}}', \overline{s}')$ with $\overline{P}' = \{P'_a\}_{a \in \mathcal{H}}$, $\overline{\mathbf{m}}' = \{m'_a\}_{a \in \mathcal{H}}$, and $\overline{s}' = \{(s'_c, d'_c)\}_{c \in \mathcal{SC}}$ where:
  - $\mathcal{K}'$ is $\mathcal{K}$ with the following messages added: (1) the first message of s_c for every $c \in \mathsf{Sch}(\mathcal{H}, \mathcal{D}) \cap \mathsf{ch}(Pr)$ with $\gamma(c) = 1$, and (2) the message $\mathbf{m}(c)$ for every $c \in (\mathsf{Net}(a, \mathcal{P}) \cup \mathsf{Dir}(a, \mathcal{D})) \cap \mathsf{ch}(Pr)$ and every $a \in \mathcal{H}$ such that $\gamma(a) = v$ and $(\mathbf{m}_a, P_a) \overset{v}{\mapsto} (\mathbf{m}, P'_a)$, i.e., the intruder learns all messages sent by the scheduled secure channels to dishonest principals, by honest principals on the network (to honest or dishonest principals) or on direct secure channels to dishonest principals.[6]
  - P'_a is such that $(\mathbf{m}_a, P_a) \overset{v}{\mapsto} (\mathbf{m}, P'_a)$ for some $\mathbf{m}$, $v = \gamma(a)$, for every $a \in \mathcal{H}$, i.e., principal a takes edge v (and performs receive-send actions according to the rule the edge to v is labeled with).
  - $\mathbf{m}'_a(c)$ for $a \in \mathcal{H}$ is equal to:
    - $\gamma(\mathcal{I})(c)$, if $c \in (\mathsf{Net}(\mathcal{P}, a) \cup \mathsf{Dir}(\mathcal{D}, a))$,
    - the first message of s_c, if $c \in \mathsf{Sch}(\mathcal{P}, a)$ and $\gamma(c) = 1$,
    - $\circ$, if $c \in \mathsf{Sch}(\mathcal{P}, a)$ and $\gamma(c) = 0$,
    - $\mathbf{m}(c)$, if $c = \mathsf{dir}(b, a)$ for some $b \in \mathcal{H}$ such that $(\mathbf{m}_b, P_b) \overset{v'}{\mapsto} (\mathbf{m}, P'_b)$ and $\gamma(b) = v'$, i.e., b has written $\mathbf{m}(c)$ on the direct secure channel c from b to a.
  - s'_c for $c = \mathsf{sch}(a, b)$ is defined as follows: Let $s_c = m_1 \dots m_n$. If $a \in \mathcal{H}$, then let $m = \mathbf{m}(c)$ where $(\mathbf{m}_a, P_a) \overset{v}{\mapsto} (\mathbf{m}, P'_a)$ for $v = \gamma(a)$. If $a \in \mathcal{D}$, then let $m = \mathbf{m}(c)$ with $\gamma(\mathcal{I}) = \mathbf{m}$. Now, if $\gamma(c) = 0$ and $m = \circ$, then $s'_c = s_c$. If $\gamma(c) = 1$ and $m = \circ$, then $s'_c = m_2 \cdots m_n$. If $\gamma(c) = 0$ and $m \neq \circ$, then $s'_c = m_1 \cdots m_n m$. If $\gamma(c) = 1$ and $m \neq \circ$, then $s'_c = m_2 \cdots m_n m$. (Note that if $\gamma(c) = 1$, then $n \geq 1$.)

[6] In case secure channels are not supposed to be read protected, one would add all messages on direct and scheduled secure channels to the current intruder knowledge. However, here we model secure channels to be read protected, i.e., the intruder only gets to see the messages on secure channels to *dishonest* principals.

- $d'_c = \mathsf{delivered}$ if $\gamma(c) = 1$, and $d'_c = \overline{\mathsf{delivered}}$ otherwise, for every $c = \mathsf{sch}(a, b) \in \mathsf{ch}(Pr)$.

We call the state $q^0 = (\mathcal{K}_0, \{P_a^0\}_{a\in\mathcal{H}}, \overline{\mathbf{m}}^0, \overline{s}^0)$ the *initial state of* S_{Pr} where $\overline{\mathbf{m}}^0 = \{\mathbf{m}_a^0\}_{a\in\mathcal{H}}$ with $\mathbf{m}_a^0(c) = \circ$ for every $c \in \mathcal{C}(\mathcal{P}, a)$ and $\overline{s}^0 = \{(s_c^0, d_c^0)\}_{c\in\mathcal{SC}}$ with $s_c^0 = \varepsilon$ and $d_c^0 = \overline{\mathsf{delivered}}$ for every $c \in \mathcal{SC}$.

We defined $\Delta(q, \alpha)$ for $\alpha \in \mathcal{H}$ in such a way that it is never empty. More precisely, if in the current vertex none of the outgoing edges can be taken, α will stay in the current state. This, in accordance with standard specifications, models that unexpected messages are ignored and guarantees that honest principals, just like all other agents in S_{Pr}, can always take an action, and hence, computations of the overall system will never be blocked. Note that one can explicitly add edges to the specification of an honest principal that lead to error states and are taken in case of unexpected messages. The concrete examples that we consider are always complete in the sense that in every vertex there will be an edge (possibly a self-loop) that the principal can take.

4 Main Results

In this section, we summarize the main results of this paper. First, we define the general protocol induced AMC-model checking problem and some subcases. We then state our (un-)decidability and complexity-theoretic results.

Let $Pr = (\mathcal{H}, \mathcal{D}, \mathcal{K}_0, \{P_a^0\}_{a\in\mathcal{H}})$ be a protocol and $S_{Pr} = \langle \Sigma_{Pr}, Q_{Pr}, \mathbb{P}_{Pr}, \pi_{Pr}, \Delta_{Pr}, \delta_{Pr}\rangle$ the concurrent game structure induced by Pr.

We call

$$\textsc{Pamc} = \{(Pr, \varphi) \mid Pr \text{ a protocol and } \varphi \text{ an AMC-formula over } \Sigma_{Pr} \text{ and } \mathbb{P}_{Pr} \text{ such that } (S_{Pr}, q^0) \models \varphi \text{ where } q^0 \text{ is the initial state of } S_{Pr}\}$$

the *(general) protocol induced AMC-model checking problem*. The size of an instance (Pr, φ) of this problem is defined to be $|Pr| + |\varphi|$.

As we will see, this problem is undecidable. To identify the main sources of undecidability and to obtain decidable subcases, we now introduce certain classes of protocols and define certain fragments of AMC.

We call a protocol Pr *dishonest scheduled secure channel free (dssc-free)* if no honest principal uses a scheduled secure channel from a dishonest principal as input channel, i.e., $\mathsf{ch}(Pr) \cap \mathsf{Sch}(\mathcal{D}, \mathcal{H}) = \emptyset$. Otherwise, we call a protocol *dssc-containing*. Note that dssc-free protocols allow honest principals to use direct secure channels from dishonest principals as input channel. Since these channels are completely controlled by the adversary, they provide the adversary with more power than scheduled secure channels. Hence, the exclusion of scheduled secure channels from dishonest principals is not a real restriction in terms of the power of the adversary.

We also consider what we call greedy protocols, which contains only greedy honest principals. Intuitively, an honest principal is greedy if it does not ignore messages in case they conform to the protocol specification. Formally, greedy protocols are defined as follows.

An a-rule $\mathbf{r} \Rightarrow \mathbf{s}$ is *consuming* if $\mathsf{ch}(\mathbf{r}) \neq \emptyset$. Intuitively, if principal a performs a consuming rule, then the form of the incoming messages matters.

An a-instance P is *greedy* if for all vertices v of P the outgoing edges of v labeled with consuming rules have priorities strictly higher than the priority of the self-loop of v (if any). Informally speaking, when a greedy principal can read a term using some consuming rule, then he has to apply such a rule, and hence, as a result moves to another vertex.

A protocol Pr is *greedy* if all of its a-instances, for $a \in \mathcal{H}$, are. Assuming a protocol to be greedy is a realistic assumption since in typical protocol specifications honest principals will not ignore messages if these messages conform to the messages they expect to receive.

We consider a fragment of AMC, called $\mathcal{I}$-monotone formulas where $\mathcal{I}$ denotes the intruder in the concurrent game structure induced by a protocol. Formally, an AMC-formula φ is *$\mathcal{I}$-positive* if all subformulas of φ of the form $\langle\langle A\rangle\rangle\psi$ with $\mathcal{I} \in A$ fall under an even number of negations and all subformulas of φ of the form $\langle\langle A\rangle\rangle\psi$ with $\mathcal{I} \notin A$ fall under an odd number of negations. An AMC-formula φ is *$\mathcal{I}$-negative* if $\neg\varphi$ is $\mathcal{I}$-positive. A formula φ is *$\mathcal{I}$-monotone* if it is either $\mathcal{I}$-positive or $\mathcal{I}$-negative.

As we mentioned above, each AMC-formula can be written in negation normal form using the abbreviation introduced in Section 2.2. It is easy to see that if φ is an $\mathcal{I}$-positive AMC-formula and φ' is the corresponding AMC-formula in negation normal form, then (i) for each subformula of φ' of the form $\langle\langle A\rangle\rangle\bigcirc\psi'$ we have that $\mathcal{I} \in A$ and (ii) for each subformula of φ' of the form $[\![A]\!]\bigcirc\psi'$ we have that $\mathcal{I} \notin A$.

$\mathcal{I}$-positive, -negative, and -monotone ATL- and ATL*-formulas are defined in the same way. As shown by Alur et al. [2] (see also Theorem 1), every ATL and ATL*-formula can be translated into an equivalent AMC-formula. It is not hard to see that the translation preserves the property of being $\mathcal{I}$-positive/-negative, i.e., the translation of an $\mathcal{I}$-positive/-negative ATL*-formula yields an $\mathcal{I}$-positive/-negative AMC formula.

While the class of $\mathcal{I}$-monotone AMC-formulas is a proper fragment of the set of all AMC-formulas in terms of expressibility, all formulas (typically ATL or fair ATL) that we encountered in the literature for specifying security properties of cryptographic protocol are $\mathcal{I}$-monotone. Hence, the restriction to $\mathcal{I}$-monotone formulas does not seem to be a restriction from a practical point of view (see Section 5 for more details).

In what follows, we denote by

PAMC(greedy/non-greedy,dssc-containing/-free,$\mathcal{I}$-positive/-negative/-monotone)

the protocol induced AMC-model checking problem where the class of protocols is restricted to those that are i) greedy/non-greedy and ii) dssc-containing/-free and the AMC-formulas considered are $\mathcal{I}$-positive/-negative/-monotone, respectively.

Now, we can state our main results. The first theorem shows that PAMC is undecidability in case of non-greedy protocols.

Theorem 2. PAMC*(non-greedy,dssc-free,$\mathcal{I}$-positive) is undecidable, and hence, so is* PAMC*(non-greedy,dssc-free,$\mathcal{I}$-negative) and* PAMC*(non-greedy,dssc-free,$\mathcal{I}$-monotone).*

The proof of this theorem is presented in Section 6. It shows that the problem is undecidable even if no scheduled secure channels are used at all, i.e., neither from dishonest nor from honest principals, and if a very simple fixed $\mathcal{I}$-positive formula is used, namely, $\langle\!\langle\mathcal{I}\rangle\!\rangle\Diamond p_{\mathsf{ok}}$ where p_{ok} is a propositional variable.

The above theorem shows that to obtain decidability, one at least has to consider greedy protocols. The following theorem exhibits another source of undecidability, namely scheduled secure channels from dishonest parties.

Theorem 3. PAMC*(greedy, dssc-containing, $\mathcal{I}$-positive) is undecidable, and hence, so is* PAMC*(greedy, dssc-containing, $\mathcal{I}$-negative) and* PAMC*(greedy, dssc-containing, $\mathcal{I}$-monotone).*

The proof is presented in Section 7. Again, a fixed formula suffices for the proof, namely $\langle\!\langle\mathcal{I}, \mathsf{sch}(\mathsf{pcp}, \mathsf{test})\rangle\!\rangle\Diamond p_{\mathsf{ok}}$.

The two theorems above show that to obtain decidability, one has to restrict protocols to be greedy and dssc-free. The following theorem states that for this class of protocol and $\mathcal{I}$-monotone formulas we obtain decidability of the AMC-model checking problem.

Theorem 4. *The problem* PAMC(*greedy, dssc-free, $\mathcal{I}$-monotone*) *is decidable. More precisely, the problem* PAMC(*greedy, dssc-free, $\mathcal{I}$-positive*) *is NEXPTIME-complete, and hence,* PAMC(*greedy, dssc-free, $\mathcal{I}$-negative*) *is coNEXPTIME-complete.*

The only gap that the above theorem leaves is whether PAMC is also decidable for non $\mathcal{I}$-monotone formulas. As explained before, from a practical point of view the theorem seems to suffice since the formulas that we encountered in the literature fall into the class of $\mathcal{I}$-monotone formulas. While, as the undecidability results show, the restrictions to greedy and dssc-free protocol cannot be avoided, these restrictions are also not severe since typically protocols are greedy and requiring protocols to be dssc-free does not restrict the power of the adversary.

5 Example Properties

In this section, we illustrate the kind of properties that can be expressed in the $\mathcal{I}$-monotone fragment of AMC. Kremer and Raskin [16, 17] were the first to formulate properties of fair exchange protocols, including contract-signing and non-repudiation protocols, in terms of fair ATL, a fragment of ATL* [2], and hence, of AMC [2] (see also Theorem 1). It turns out that all properties that Kremer and Raskin have formulated fall into the $\mathcal{I}$-monotone fragment of AMC, suggesting that the $\mathcal{I}$-montone fragment of AMC suffices for most properties of interest. We demonstrate this fact by recalling some of these properties. Since, as mentioned, AMC is more expressive than ATL*, in what follows, for convenience we use ATL* as the specification language. As we will see, we will only

need $\mathcal{I}$-monotone ATL* formulas. As mentioned in Section 4, $\mathcal{I}$-monotonicity is preserved under the translation to AMC as proposed in [2], i.e., AMC formulas corresponding to $\mathcal{I}$-monotone ATL* formulas are $\mathcal{I}$-monotone.

The precise formulations of the properties stated by Kremer and Raskin typically depend on the specific protocol analyzed. For concreteness, we will therefore consider the specification of the ASW protocol Pr_{ASW}, with honest A and T and dishonest B, as presented in Section 3.4. Hence, formulas are stated w.r.t. the concurrent game structure $S_{Pr_{ASW}}$ induced by Pr_{ASW}.

As for example in [2], we use $\Diamond\varphi$ (read "eventually φ") as abbreviation for the LTL-formula $(\mathrm{true}\,\mathcal{U}\,\varphi)$ and $\Box\varphi$ (read "always φ") as abbreviation for $\neg\Diamond\neg\varphi$.

Fairness. According to Kremer and Raskin, a protocol is *un*fair for honest A if dishonest B together with all scheduled secure channels has a strategy to obtain a signed contract from A such that A does not have a strategy to receive a signed contract from B, given that the secure scheduled channels between A and T are fair, i.e., messages on these channels are eventually delivered.

For the ASW protocol as specified in Section 3.4 this property can be formalized by the following $\mathcal{I}$-positive ATL* formula where $\mathcal{SC} = \mathsf{sch}(\{A, B, T\}, \{A, B, T\}) \cap \mathsf{ch}(\mathrm{Pr}_{ASW})$ is the set of all scheduled secure channels used in Pr_{ASW}:

$$\langle\!\langle \mathcal{I}, \mathcal{SC} \rangle\!\rangle \Diamond (p_{\texttt{Bhascontract}} \wedge \neg \langle\!\langle A \rangle\!\rangle (\varphi_{FairSch} \rightarrow \Diamond \varphi_{\texttt{Ahascontract}})) \tag{1}$$

where

$$\varphi_{FairSch} = \bigwedge_{c \in \mathsf{sch}(\{A,T\},\{A,T\})} \Box\Diamond\neg\mathsf{empty}_c \rightarrow \Box\Diamond\mathsf{delivered}_c \tag{2}$$

says that the scheduled secure channels between A and T are fair[7] and

$$\varphi_{\texttt{Ahascontract}} = p_{\mathsf{contract}} \vee p_{\mathsf{resolved}_1} \vee p_{\mathsf{resolved}_2} \tag{3}$$

says that A has a standard or replacement contract, according to the protocol specification.

The property formulated in (1) requires A to use the protocol in a "smart" way in order to get a signature from B. An alternative, stronger formulation of fairness would require that if A finished the protocol, and hence, if A cannot take any further action, either both A and B or neither of the two parties has a signed contract, provided that the scheduled secure channels between A and T are fair. In other words, the protocol is *un*fair, if there exists a state in the protocol run where i) A cannot take any further action, ii) A does not have a signature from B, but iii) B has a signature from A. Formally:

$$\langle\!\langle \mathcal{I}, A, T, \mathcal{SC} \rangle\!\rangle (\varphi_{FairSch} \wedge \Diamond(\varphi_{\texttt{Afinished}} \wedge \neg\varphi_{\texttt{Ahascontract}} \wedge p_{\texttt{Bhascontract}})) \tag{4}$$

[7] Alternatively, one could require the other scheduled secure channels to be fair as well. The formulation of fairness for channels as presented is standard and, for example, also appears in [2].

where

$$\varphi_{\texttt{Afinished}} = p_{\mathsf{contract}} \vee p_{\mathsf{resolved}_1} \vee p_{\mathsf{resolved}_2} \vee p_{\mathsf{aborted}} \tag{5}$$

says that A finished her protocol run.

Timeliness. According to Kremer and Raskin, a protocol is timely for honest A if A has a strategy to finish the protocol while preserving fairness. Again, the scheduled secure channels (at least those between A and T) are required to be fair. Formally, timeliness for A is expressed by the following $\mathcal{I}$-negative ATL* formula:

$$\begin{aligned}\langle\!\langle A\rangle\!\rangle(\varphi_{FairSch} \rightarrow \Diamond(\varphi_{\texttt{Afinished}} \wedge \\ (\neg\varphi_{\texttt{Ahascontract}} \rightarrow \neg\langle\!\langle \mathcal{I}, \mathcal{SC}\rangle\!\rangle(\varphi_{FairSch} \wedge \Diamond p_{\texttt{Bhascontract}})))).\end{aligned} \tag{6}$$

Balance and Abuse-freeness. According to Kremer and Raskin, a protocol is *un*balanced for honest A if at some stage of the protocol run dishonest B has both a strategy to obtain a signature from A and a strategy to prevent A from obtaining a signature from B. For the protocol to be abusive, one additionally requires that B can convince an outside party C of this property. Whether or not B has this ability is indicated, in the model of Kremer and Raskin, by a propositional variable p_{prove2C}, which can as well be expressed in terms of propositional variables on vertices. Again, the scheduled secure channels between A and T are required to be fair. Unbalanced for A can be formulated as an $\mathcal{I}$-positive ATL* formula as follows:

$$\langle\!\langle \mathcal{I}, \mathcal{SC}, A, T\rangle\!\rangle\varphi_{FairSch} \wedge \Diamond(\varphi_{\mathsf{getcontract}} \wedge \varphi_{\mathsf{prevent}}) \tag{7}$$

where

$$\varphi_{\mathsf{getcontract}} = \langle\!\langle \mathcal{I}, \mathcal{SC}''\rangle\!\rangle\varphi_{FairSch} \rightarrow \Diamond p_{\texttt{Bhascontract}}, \tag{8}$$

$$\varphi_{\mathsf{prevent}} = \langle\!\langle \mathcal{I}, \mathcal{SC}''\rangle\!\rangle\varphi_{FairSch} \rightarrow \Diamond(\neg\langle\!\langle A\rangle\!\rangle(\varphi_{FairSch} \rightarrow \Diamond\varphi_{\texttt{Ahascontract}})) \tag{9}$$

with $\mathcal{SC}'' = \mathsf{sch}(\{B, T\}, \{B, T\}) \cap \mathsf{ch}(\mathrm{Pr}_{ASW})$.

Given p_{prove2C}, according to Kremer and Raskin abusiveness for A is formalized by the following ATL* formula:

$$\langle\!\langle \mathcal{I}, \mathcal{SC}, A, T\rangle\!\rangle\varphi_{FairSch} \wedge \Diamond(p_{\mathsf{prove2C}} \wedge \varphi_{\mathsf{getcontract}} \wedge \varphi_{\mathsf{prevent}}) \tag{10}$$

We note that the more general, protocol-independent formulation of abuse-freeness proposed in [15] is not captured by the formulation of Kremer and Raskin. The formulation in [15] defines abuse-freeness in terms of certain tests. In order to formulate this property in ATL* or AMC, one would need to augment these logics by certain predicates reflecing the tests.

Very similar formulas as the ones presented above have been stated by Kremer and Raskin for non-repudiation protocols [17]. They are $\mathcal{I}$-monotone as well.

6 Proof of Theorem 2

We prove Theorem 2 by a reduction from Post's Corresponding Problem (PCP). Let us first recall the definition of PCP.

Given an alphabet A with $|A| \geq 2$, an instance Π of PCP over A is a sequence $(u_i, v_i)_{i=1}^n = (u_1, v_1), \ldots, (u_n, v_n)$ of pairs (u_i, v_i) of words $u_i, v_i \in A^*$. A solution of such an instance is a non-empty sequence $(k_i)_{i=1}^l = k_1, \ldots, k_l$ of indices $k_i \in \{1, \ldots, n\}$, $i \in \{1, \ldots, l\}$, for some l such that $u_{k_1} \cdots u_{k_l} = v_{k_1} \cdots v_{k_l}$. Now, given an instance of PCP (over A) the question is whether it has a solution. It is well-known that this problem is undecidable.

We now prove Theorem 2 by reduction from PCP. Let Π be an instance of PCP over A as above. We (effectively) associate a protocol Pr_Π and a formula φ_Π to Π such that Π has a solution iff $(S_{Pr_\Pi}, q^0) \models \varphi_\Pi$ where q^0 is the initial state of S_{Pr_Π}.

The set of atoms of Pr_Π is $\mathcal{A}_\Pi = A \cup \{\bot, 1, \ldots, n\}$. For a word $u \in \mathcal{A}_\Pi^*$ and a term t, we define $t \cdot u$ by induction on the length of u: $t \cdot u = t$ for $u = \varepsilon$ and $t \cdot u = \langle t, a \rangle \cdot v$ for $u = av$ and $a \in \mathcal{A}_\Pi$.

We encode a solution of Π as a sequence of terms over $\mathcal{A}_\Pi = A \cup \{\bot, 1, \ldots, n\}$ as follows: A sequence $t_0, \ldots, t_l$ of terms over $\mathcal{A}_\Pi$ is called a *solution sequence for* Π if the following three conditions are satisfied:

i) $t_0 = \langle \bot, \bot, \bot \rangle$,

ii) $t_l = \langle m_1, m_2, m_2 \rangle$ for terms m_1 and m_2 over $\mathcal{A}_\Pi$, and

iii) for all $i \in \{0, \ldots, l-1\}$, if $t_i = \langle s, s', s'' \rangle$, then $t_{i+1} = \langle s \cdot j, s' \cdot u_j, s'' \cdot v_j \rangle$ for some $j \in \{1, \ldots, n\}$.

It is easy to see that Π has a solution iff there exists a solution sequence for Π. For a solution $(k_i)_{i=1}^l$ of Π we call the solution sequence $t_0, \ldots, t_l$ with $t_0 = \langle \bot, \bot, \bot \rangle$ and $t_{i+1} = \langle s_1 \cdot k_{i+1}, s_2 \cdot u_{k_{i+1}}, s_3 \cdot v_{k_{i+1}} \rangle$ for $0 \leq i < l$ and $t_i = \langle s_1, s_2, s_3 \rangle$ the *solution sequence associated with* $(k_i)_{i=1}^l$.

We define $\varphi_\Pi = \langle\langle \mathcal{I} \rangle\rangle \Diamond p_{\mathsf{ok}}$, i.e., this formula is true in those states where $\mathcal{I}$ has a strategy to obtain ok. (Note that φ_Π is presented as an ATL-formula, which by Theorem 1 can be turned into an AMC-formula).

The protocol Pr_Π is defined as follows: There is one honest principal, called test, and one dishonest principal, called pcp, in Pr_Π. The initial knowledge of the intruder is defined to be $\mathcal{K}_\Pi = \mathcal{A}_\Pi$. The honest principal test is specified by the test-instance P_{test} depicted in Figure 3 and explained next. Altogether, we define $Pr_\Pi = (\{\mathsf{test}\}, \{\mathsf{pcp}\}, \mathcal{K}_\Pi, \{P_{\mathsf{test}}\})$.

Principal test does not use any direct or scheduled secure channels. Hence, the only channel from which principal test reads is $\mathsf{net}(\mathsf{pcp}, \mathsf{test})$ and the only channel to which test writes is $\mathsf{net}(\mathsf{test}, \mathsf{pcp})$. Hence, the left-hand side of the test-rules depicted in Figure 3 are the messages test reads from $\mathsf{net}(\mathsf{pcp}, \mathsf{test})$ and the messages on the right-hand side of these rules are the messages test writes to $\mathsf{net}(\mathsf{test}, \mathsf{pcp})$. The labels [0] and [1] present the priorities of the edges. Vertices with boxes have self-loops with priority 0, those without do not have self-loops. Note that P_{test} is non-greedy; the greediness condition is violated in vertex $\mathsf{test\text{-}seq}$.

The purpose of principal test is to test whether the intruder is able to send a solution sequence. More precisely, test guarantees that the intruder has a strategy to obtain ok iff the intruder is able to send a solution sequence to test, where the intruder is supposed to send in every step of a computation in S_{Pr_Π} one element of such a sequence. In initial, once $\langle \bot, \bot, \bot \rangle$ is received, test can decide to go to test-initial or test-seq. The purpose of going to test-initial is to check whether the successor term of $\langle \bot, \bot, \bot \rangle$ is in fact a successor term according to the definition of a solution sequence. (The out-going edge from test-initial with priority 0 guarantees that the intruder has to send a new term after the initial term $\langle \bot, \bot, \bot \rangle$.) The purpose of going to test-seq is to either stay there until a term of the form $\langle x, y, y \rangle$ is received, and hence, a valid last term of the sequence, or to check at some point of the sequence whether two consecutive terms are connected according to the definition of solution sequences (which can be done by moving to test-pair).

Before we prove the correctness of our construction, we introduce some notation. By definition, see Section 3.5, a state of the concurrent game structure $\mathcal{S}_{Pr_\Pi}$ specified by the protocol Pr_Π is a tuple $p = (\mathcal{K}, \{P\}, \{\mathbf{m}\}, \bar{s})$ where $\mathcal{K}$ is the intruder knowledge, P is a test-instance, $\mathbf{m}$ is a mapping that assigns messages to the channels read by test, and $\bar{s}$ is a family of message sequences representing the states of the scheduled secure channels. Because there are no secure channels used in this protocol we will omit the last component of states when referring to them. The only channel from which participant test reads messages is $\mathsf{net}(\mathsf{pcp}, \mathsf{test})$. Thus, $\mathbf{m}$ assigns messages to $\mathsf{net}(\mathsf{pcp}, \mathsf{test})$ and we will only write the message $\mathbf{m}(\mathsf{net}(\mathsf{pcp}, \mathsf{test}))$ when specifying a state. For ease of notation, we specify a test-instance in a state simply by its current root node. Thus, by these conventions the initial state of the concurrent game structure $\mathcal{S}_{Pr_\Pi}$ is given by $q_{\mathsf{init}} = (\mathcal{K}_\Pi, \mathsf{initial}, \circ)$. For a state $p = (\mathcal{K}, s, m)$ we denote the components $\mathcal{K}$, s, and m by $\mathcal{K}(p)$, $\mathsf{state}_{\mathsf{test}}(p)$, and $\mathsf{net}(\mathsf{pcp}, \mathsf{test})(p)$, respectively. We call m the *value of channel* $\mathsf{net}(\mathsf{pcp}, \mathsf{test})$ *in state* p.

We now show that Π has a solution iff $(S_{Pr_\Pi}, q^0) \models \varphi_\Pi$:

$\Rightarrow$: First, we show that if Π has a solution, then the intruder has a strategy to obtain ok. Intuitively, the strategy of the intruder is to send a solution sequence to instance test. More specifically, let $(k_i)_{i=1}^l$ be a solution of Π and let $t_0, \ldots, t_l$ be the solution sequence associated with $(k_i)_{i=1}^l$. We may assume, w.l.o.g., that t_l is the first among the terms in the sequence of the form $\langle m_1, m_2, m_2 \rangle$ for some m_1 and m_2.

The (positional) strategy σ_{ok} of the intruder to obtain ok only depends on the message on channel $\mathsf{net}(\mathsf{pcp}, \mathsf{test})$. We define σ_{ok} by the following table: (Note that the choice of the intruder is which message he sends to instance test, i.e., what the value of $\mathsf{net}^{\mathsf{pcp}}_{\mathsf{test}}$ in the next state of the concurrent game structure $\mathcal{S}_{Pr_\Pi}$ is.)

$\mathsf{net}(\mathsf{pcp}, \mathsf{test})(q)$	$\sigma_{\mathsf{ok}}(q)$
$\circ$	t_0
t_i	t_{i+1} for $i \in \{0, \ldots, l-1\}$

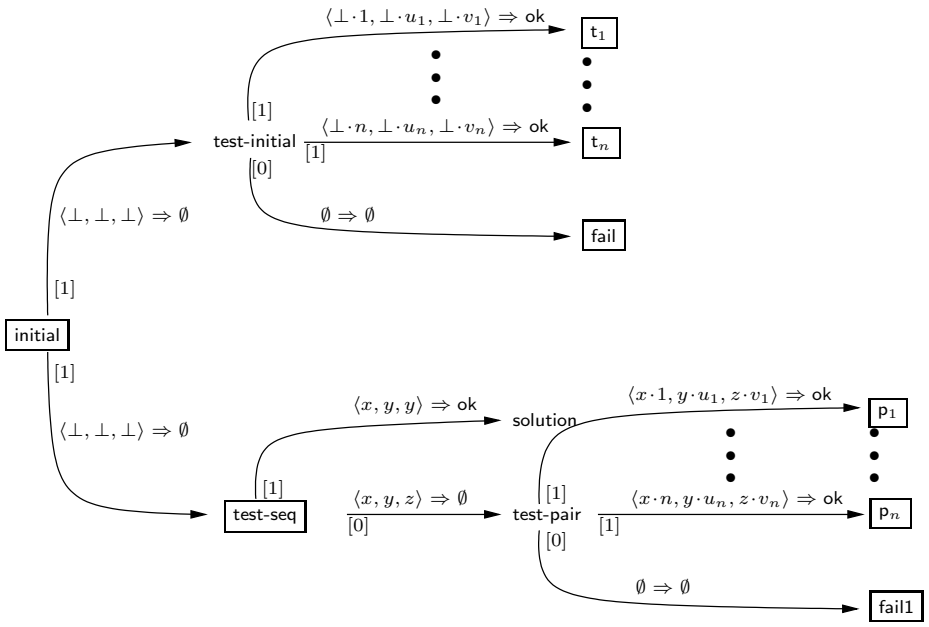


Fig. 3. Honest instance P_{test} in the proof of Theorem 2. A box around a node is an abbreviation for a self loop with priority 0.

We have to show that if the intruder follows this strategy, then every computation in $\mathcal{S}_{Pr_{\Pi}}$ starting from the initial state will reach a state q such that $\mathsf{ok} \in \mathcal{K}(q)$. Let $\rho = q_0 q_1 \ldots$ be a computation consistent with σ_{ok} and such that $q_0 = q^0$. According to the specification of instance test, see Figure 3, we know that $\mathsf{state}_{\mathsf{test}}(q_2) \in \{\mathsf{test\text{-}initial}, \mathsf{test\text{-}seq}\}$. If $\mathsf{state}_{\mathsf{test}}(q_2) = \mathsf{test\text{-}initial}$, then $\mathsf{ok} \in \mathcal{K}(q_3)$ since we have that $\mathsf{net}(\mathsf{pcp}, \mathsf{test})(q_2) = t_1$. Thus, in this case we are done. If $\mathsf{state}_{\mathsf{test}}(q_2) = \mathsf{test\text{-}seq}$, then there is a minimal $i > 2$ such that $\mathsf{state}_{\mathsf{test}}(q_i) \neq \mathsf{test\text{-}seq}$ (note that if $\mathsf{state}_{\mathsf{test}}(q_j) = \mathsf{test\text{-}seq}$ and $\mathsf{net}(\mathsf{pcp}, \mathsf{test})(q_j) = t_n$, then according to the specification of instance test, $\mathsf{ok} \in \mathcal{K}(q_{j+1})$ and $\mathsf{state}_{\mathsf{test}}(q_{j+1}) = \mathsf{solution}$). If $\mathsf{state}_{\mathsf{test}}(q_i) = \mathsf{solution}$, then we are obviously done. If $\mathsf{state}_{\mathsf{test}}(q_i) = \mathsf{test\text{-}pair}$, then we know that $\mathsf{net}(\mathsf{pcp}, \mathsf{test})(q_{i-1}) \neq t_n$ (note that if the intruder has already sent t_n in q_{i-1} then $\mathsf{state}_{\mathsf{test}}(q_i)$ would be $\mathsf{solution}$). By the specification of instance test and by the definition of σ_{ok} we know that $\mathsf{state}_{\mathsf{test}}(q_{i+1}) \in \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ and thus, we are done.

$\Leftarrow$: Now, we prove the other direction, i.e., if the intruder has a strategy to obtain ok, then Π has a solution. It suffices to show that there is a solution sequence for instance Π. Let σ_{ok} be a strategy of the intruder such that in each computation of $\mathcal{S}_{Pr_{\Pi}}$ starting from the initial state and consistent with σ_{ok} the intruder obtains ok. We want to show that the intruder has to send a solution sequence $t_0, t_1, \ldots, t_l$ for Π to instance test. More specifically, we want to show

that there is a computation $\rho = q_0 q_1 \ldots$ of $\mathcal{S}_{Pr_\Pi}$, $q_0 = q^0$, and an index i such that $\sigma_{\mathsf{ok}}(q_0), \sigma_{\mathsf{ok}}(q_0 q_1), \ldots, \sigma_{\mathsf{ok}}(q_0 \cdots q_i)$ is a solution sequence for Π.

It is easy to see that it is w.l.o.g. to assume that (*) $\sigma_{\mathsf{ok}}(q^0) = \langle \bot, \bot, \bot \rangle$. The successor state q_1 of q^0 is $(\mathcal{K}_\Pi, \mathsf{initial}, \langle \bot, \bot, \bot \rangle)$. Since one possible choice of instance test in state q_1 is to proceed to state $\mathsf{test\text{-}initial}$ one possible successor state of q_1 is $(\mathcal{K}_\Pi, \mathsf{test\text{-}initial}, \circ)$, see Figure 3. Thus, if $\sigma_{\mathsf{ok}}(q_1) \neq \langle \bot \cdot j, \bot \cdot u_j, \bot \cdot v_j \rangle$ for all $j = 1, \ldots, n$, then instance test will proceed to state fail and the intruder will not obtain ok. Since σ_{ok} is a strategy for the intruder to obtain ok we can conclude that (**) $\sigma_{\mathsf{ok}}(q_1)$ is of the form $\langle \bot \cdot j, \bot \cdot u_j, \bot \cdot v_j \rangle$ for some $j \in \{1, \ldots, n\}$.

If the choice of instance test in state q_1 is to proceed to state $\mathsf{test\text{-}seq}$, then the successor state of q_1 is $(\mathcal{K}_\Pi, \mathsf{test\text{-}seq}, \sigma_{\mathsf{ok}}(q_1))$. By an inductive argument it is easy to see that for a run $\rho = q_0, q_1, q_2, \ldots$ of $\mathcal{S}_{Pr_\Pi}$ that is consistent with σ_{ok} we have that: (***) if $\mathsf{state}_{\mathsf{test}}(q_i) = \mathsf{test\text{-}seq}$, $\mathsf{net}(\mathsf{pcp}, \mathsf{test})(q_i) = \langle m_1, m_2, m_3 \rangle$, where $m_2 \neq m_3$, then $\sigma_{\mathsf{ok}}(q_i) = \langle m_1 \cdot j, m_2 \cdot u_j, m_3 \cdot v_j \rangle$ for some $j \in \{1, \ldots, n\}$.

Since σ_{ok} is a strategy for the intruder to obtain ok there is no computation $\rho = q_0, q_1, \ldots$ of $\mathcal{S}_{Pr_\Pi}$ that is consistent with σ_{ok} such that $\mathsf{state}_{\mathsf{test}}(q_j) = \mathsf{test\text{-}seq}$ from some point on, i.e., $\mathsf{state}_{\mathsf{test}}(q_j) = \mathsf{test\text{-}seq}$ for all $j > i$ for some $i > 0$. Since instance test can decide to stay in state $\mathsf{test\text{-}seq}$ if $\mathsf{net}(\mathsf{pcp}, \mathsf{test})$ is not of the form $\langle m_1, m_2, m_2 \rangle$ for some terms m_1 and m_2 we can conclude that there is a computation $\rho' = q'_0, q'_1, \ldots$ that is consistent with σ_{ok} such that there is some minimal i such that $\mathsf{state}_{\mathsf{test}}(q'_i) = \mathsf{solution}$. Thus, $\mathsf{net}(\mathsf{pcp}, \mathsf{test})(q'_{i-1}) = \langle m_1, m_2, m_2 \rangle$ for some terms m_1, m_2. Together with (*), (**), and (***) we can conclude that the sequence $\sigma_{\mathsf{ok}}(q'_0), \sigma_{\mathsf{ok}}(q'_0 q'_1), \ldots, \sigma_{\mathsf{ok}}(q'_0 \cdots q'_{i-2})$ is a solution sequence for Π. □

7 Proof of Theorem 3

The proof of Theorem 3 is very similar to the one of Theorem 2.

In Section 6, we used the non-greediness of instance P_{test} in node $\mathsf{test\text{-}seq}$ to check property iii) of solution sequences: From node $\mathsf{test\text{-}seq}$ of instance P_{test} there is a self-loop with priority 0 and a consuming edge with priority 0 (see Figure 3). Applying the self-loop means "do not check" and applying the outgoing edge means "check" whether the next two messages fulfill property iii) of solution sequences. In other words, in node $\mathsf{test\text{-}seq}$, P_{test} can non-deterministically decide when to check property iii) of solution sequences for two consecutive messages. However, now we are not allowed to use non-greediness anymore. Nevertheless, we can simulate this non-deterministic behavior by introducing a new instance, which we will call P_{check}. Basically, this instance will send a "check" message to P_{test} in order to tell P_{test} when to check. More precisely, in the very first step of the protocol run P_{check} will send the "check" message on a scheduled secure channel $\mathsf{sch}(\mathsf{check}, \mathsf{test})$ to P_{test}. Then, this scheduled secure channel will non-deterministically decide when it delivers the "check" message. (Alternatively, P_{check} could be defined to be non-deterministic and decide when to send the "check" message to P_{test}, now on a direct secure channel. However,

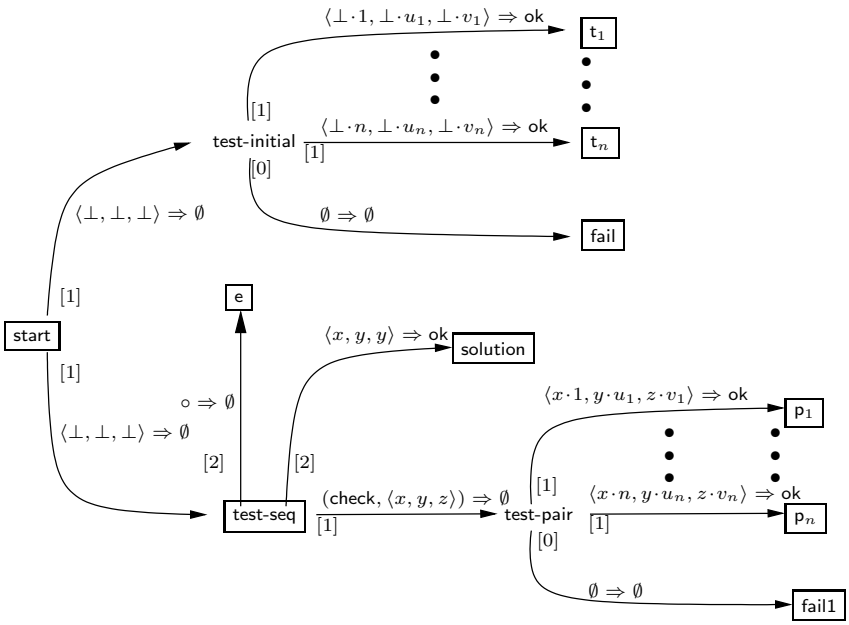


Fig. 4. Honest instance P_{test} in the proof of Theorem 3. A box around a node is an abbreviation for a self-loop with priority 0.

this instance seems to be less natural than the first one.) The only problem is that if in one step of the protocol run the $\mathsf{sch}(\mathsf{check}, \mathsf{test})$ decides to deliver the "check" message, then in the next state the intruder knows that a check will be performed in the next step. Hence, he could produce a message that together with the previous message sent passes the test, even though the rest of the sequence that the intruder is sending does not satisfy the conditions on solution sequences. In other words, the intruder knows one step in advance, i.e., before sending the second message of the two messages to be checked, when a check is going to be performed. (Note that in the proof of Theorem 2 this was not the case since by the time P_{test} changed its internal state to perform the check, the intruder must have sent the second message already.) We therefore let the intruder communicate with P_{test} only over a scheduled secure channel $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$. Now, by the time the intruder gets to know that a check is going to be performed, he must already have sent the second message of the two messages to be checked to $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$. In other words, he must already have committed to the second message, and hence, cannot change it anymore. Thus, this second message must have been valid in the first place.

We now present the reduction formally and prove its correctness. Let $\Pi = (u_i, v_i)_{i=1}^n$ be an instance of PCP over the alphabet A. As in Section 6, we define a protocol Pr_Π and an ATL-formula φ_Π such that Π has a solution iff $(S_{Pr_\Pi}, q^0) \models \varphi_\Pi$ where q^0 is the initial state of S_{Pr_Π}.

Protocol Pr_{Π} contains two honest principals, P_{test} and P_{check} and one dishonest principal pcp. The initial knowledge of the intruder is defined by $\mathcal{K}_{\Pi} = A$. Altogether, $P_{\Pi} = (\{\mathsf{test}, P_{\mathsf{check}}\}, \{\mathsf{pcp}\}, \mathcal{K}_{\Pi}, \{P_{\mathsf{test}}, P_{P_{\mathsf{check}}}\})$ where the test-instance P_{test} is specified by the tree given in Figure 4, explained below. The instance P_{check} consists of only two edges: one edge (from the root to another node, say v) is labeled with a check-rule of the form $\emptyset \Rightarrow \mathsf{check}$ where check is sent via $\mathsf{sch}(\mathsf{check}, \mathsf{test})$ to P_{test}. The second edge is a self-loop at v. Hence, the only action that P_{check} takes is to send, in the first protocol step, check to $\mathsf{sch}(\mathsf{check}, \mathsf{test})$.

The definition of P_{test} (see Figure 4) is similar to the one of P_{test} in the proof of Theorem 2. However, now P_{test} may receive messages from two scheduled secure channels, $\mathsf{sch}(\mathsf{check}, \mathsf{test})$ and $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$. The convention in Figure 4 is that if the left-hand side of the rule consists only of one message, then this message comes from $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$. If the left-hand side is a tuple with two components (this is only the case for the rule the edge from $\mathsf{test\text{-}seq}$ to $\mathsf{test\text{-}pair}$ is labeled with), then the first component is the message coming from $\mathsf{sch}(\mathsf{check}, \mathsf{test})$ and the other one from $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$.

The intuition behind P_{test} as presented in Figure 4 is the same as the one in Figure 3. The edge from $\mathsf{test\text{-}seq}$ to e is needed to guarantee that $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ always delivers messages. The priority of the edge from $\mathsf{test\text{-}seq}$ to $\mathsf{test\text{-}pair}$ is now 1, instead of 0, and hence, $\mathsf{test\text{-}seq}$ satisfies the greediness condition. However, because of the message check coming from $\mathsf{sch}(\mathsf{check}, \mathsf{test})$, this edge will only be applied if $\mathsf{sch}(\mathsf{check}, \mathsf{test})$ has delivered check.

The formula φ_{Π} is defined as $\varphi_{\Pi} = \langle\!\langle \mathcal{I}, \mathsf{sch}(\mathsf{pcp}, \mathsf{test}) \rangle\!\rangle F p_{\mathsf{ok}}$. (Note that, as in Section 6, we define, w.l.o.g., φ_{Π} as an ATL-formula.)

We now prove that our construction is correct, i.e., we prove that Π has a solution iff $(S_{Pr_{\Pi}}, q^0) \models \varphi_{\Pi}$:

$\Rightarrow$: First we show that if Π has a solution, then $\mathcal{I}$ together with the scheduled secure channel $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ has a strategy such that $\mathcal{I}$ obtains ok. Similar to the proof of Theorem 2, the strategy of $\mathcal{I}$ is to send a solution sequence to test via $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ and the strategy of $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ is to deliver a message whenever possible. More precisely, let $(k_i)_{i=1}^{l}$ be a solution of Π and let $t_0, \ldots, t_l$ be the solution sequence associated with $(k_i)_{i=1}^{l}$ (as defined in Section 6).

It is easy to see that if $\mathcal{I}$ and $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ follow their strategy, then after two steps participant test is in state start and message $\langle \bot, \bot, \bot \rangle$ is on channel $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ ready to be read by test. Also, P_{check} has written check on $\mathsf{sch}(\mathsf{check}, \mathsf{test})$, which in turn may or may not have delivered this message. At this point, participant test has two alternatives to proceed: 1) advance to $\mathsf{test\text{-}initial}$ or 2) advance to $\mathsf{test\text{-}seq}$ (see Figure 4). If test advances to $\mathsf{test\text{-}initial}$, then in the next step test advances to one of the states $\mathsf{t}_1, \ldots, \mathsf{t}_n$ and the intruder will obtain ok. If test advances to $\mathsf{test\text{-}seq}$, we have to distinguish between two cases: 2a) message check is not delivered to test by $\mathsf{sch}(\mathsf{check}, \mathsf{test})$ before the last message of the solution sequence sent by the intruder is delivered to test by $\mathsf{sch}(\mathsf{pcp}, \mathsf{test})$ and 2b) message check is delivered to test before the last message of the solution sequence sent by the intruder is delivered to test. In

case 2a) participant test advances to solution and the intruder will obtain ok. In case 2b) participant test advances to test-pair and since the intruder has sent a solution sequence to test and sch(pcp, test) immediately delivers all messages, test advances to one of the states $\mathsf{p}_1, \ldots, \mathsf{p}_n$ and the intruder obtains ok.

$\Leftarrow$: Now, we have to show that if the intruder together with the scheduled secure channel sch(pcp, test) has a strategy σ_{ok} such that the intruder obtains ok, then the PCP-instance Π has a solution. It is easy to see when playing according to strategy σ_{ok} that the intruder at some point must send $\langle \bot, \bot, \bot \rangle$ to test over the scheduled secure channel sch(pcp, test) and that at some point sch(pcp, test) delivers this message. Thus, at some point participant test is in state start and $\langle \bot, \bot, \bot \rangle$ is ready to be read on channel sch(pcp, test). There are two possible ways of how participant test may proceed: 1) advancing to state test-initial and 2) advancing to state test-seq. If 1) participant test advances to test-initial, then we have that (*) there has to be a message of the form $\langle \bot \cdot i, \bot \cdot u_i, \bot \cdot v_i \rangle$ stored on channel sch(pcp, test) for some $i \in \{1, \ldots, n\}$ since otherwise in the next step test would advance to state fail and the intruder would not obtain ok, in contradiction to the assumption. If 2) participant test advances to state test-seq, then because of the edge from test-seq to e, we know that in each step sch(pcp, test) has to deliver a message to test, since otherwise participant test would advance to e and the intruder could not obtain ok anymore. We now distinguish between two cases: 2a) message check is never delivered to test by the scheduled secure channel sch(check, test) and 2b) message check is delivered to test eventually. In case of 2a), we have that (**) at some point sch(pcp, test) must deliver a message of the form $\langle m_1, m_2, m_2 \rangle$ to test since this is the only way for the intruder to obtain ok. In case of 2b), similar to the proof of Theorem 2, we can conclude that (***) the sequence of messages delivered by sch(pcp, test) to test satisfy property iii) of the properties of solution sequences. At this point we use that the intruder sends messages to test via a scheduled secure channel. This guarantees that the intruder must have sent the next message in a sequence to sch(pcp, test) before he knows that a check is going to be performed. Now, from (*), (**), and (***) we can conclude that the intruder has to send a solution sequence to test, and thus, Π has a solution.

8 Proof of Theorem 4

To prove Theorem 4, it obviously suffices to show that PAMC(greedy, dssc-containing, $\mathcal{I}$-positive) is NEXPTIME-complete. In the following subsections, this statement is proved. The proof of the complexity upper bound is presented in Section 8.1, with the proofs of key lemmas postponed to Section 8.2 to 8.6. The complexity lower bound is shown in Section 8.7.

8.1 Poof of Theorem 4 Using Key Lemmas

In this section, we prove Theorem 4 using three key lemmas. The proofs of these lemmas are postponed to subsequent sections.

Let $Pr = (\mathcal{H}, \mathcal{D}, \mathcal{K}, \{P_a\}_{a\in\mathcal{H}})$ be a greedy and dssc-free protocol and φ be an $\mathcal{I}$-positive AMC-formula over Σ_{Pr} and $\mathbb{P}_{Pr}$. Since every AMC-formula can (in polynomial-time) be turned into an AMC-formula in negation normal form, we may, w.l.o.g., assume that φ is in negation normal form. Let $S = S_{Pr} = \langle \Sigma, Q_S, \mathbb{P}, \pi, \Delta, \delta\rangle$ be the concurrent game structure induced by Pr.

We define an equivalence relation $\sim$ on Q_S as follows. For $q, q' \in Q_S$, we write $q \sim q'$ if q and q' are equal up to the messages on input ports of honest participants, i.e., $\mathcal{K}_q = \mathcal{K}_{q'}$, $\overline{P}_q = \overline{P}_{q'}$, and $\overline{s}_q = \overline{s}_{q'}$. We will write $q \prec q'$, if $q \not\sim q'$ and q' is a descendant of q in S, i.e., there exists $q_1, \ldots, q_n$ such that $q_1 = q$, $q_n = q'$, and q_{i+1} is a successor of q_i for every $i = 1, \ldots, n-1$. We extend these relations to states of the parity game $G^{\varphi}_{(S,q^0)}$ where q^0 is the initial state of S: We write $(q, \psi) \sim (q', \psi')$, if $q \sim q'$, and we write $(q, \psi) \prec (q', \psi')$, if $q \prec q'$.

We call a state q of S consuming, if on the input port of some honest participant a there is a message which can be read by some consuming rule. Since, by assumption a is greedy, this implies that a's state will change in the next step, i.e., a moves to a new vertex. Formally, a state $q = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \overline{s})$ is *consuming*, if there exists $a \in \mathcal{H}$ with $P_a = (V, E, v_0, \ell_p, \ell)$ and if there exists $v \in V$ such that $\ell(v_0, v) = (\mathbf{r} \Rightarrow \mathbf{s})$ is consuming and $\mathbf{m}_a$ matches with $\mathbf{r}$. Otherwise, q is called *non-consuming*. Note that in non-consuming states, honest principals can only take edges with non-consuming rules (including self-loops). In particular, any two equivalent, non-consuming states have the same set of successors. We call a vertex $v = (q, \psi)$ in $G^{\varphi}_{(S,q^0)}$ *non-consuming* if q is non-consuming.

The following definition says that a strategy is $\sim$-uniform if the intruder chooses the same messages whenever he is in certain non-consuming, equivalent states.

Definition 1. Consider the game $G^{\varphi}_{(S,q^0)}$ as above. A strategy f for Player 0 is *$\sim$-uniform*, if it is memoryless and moreover, for all non-consuming states v, v' with $v \sim v'$ we have that:

(a) If $v = (q, \langle\!\langle A\rangle\!\rangle \circ \psi)$, $v' = (q', \langle\!\langle A\rangle\!\rangle \circ \psi)$ with $f(v) = (q, \Box_c \psi)$ and $f(v') = (q', \Box_{c'} \psi)$, then $c = c'$.

(b) If $v = (q, \Diamond_c \psi)$ and $v' = (q', \Diamond_c \psi)$ then $f(v) = f(v')$.

The following lemma says that it suffices to consider $\sim$-uniform strategies.

Lemma 1. *If there exists a winning strategy of Player 0 in $G^{\varphi}_{(S,q^0)}$, then there exists a $\sim$-uniform winning strategy for this player.*

The proof of this lemma is provided in Section 8.4. We call a strategy graph induced by a $\sim$-uniform strategy a *$\sim$-uniform strategy graph.*

Lemma 2. *If F is a $\sim$-uniform strategy graph for Player 0 in $G^{\varphi}_{(S,q^0)}$, then the length of every path in F starting from the initial vertex and without repetitions has length polynomially bounded in $|Pr| + |\varphi|$. Also, the number of reachable vertices of F is exponentially bounded in $|Pr| + |\varphi|$.*

The proof of this lemma is provided in the Section 8.5. Lemma 1 and 2 imply that if Player 0 wins the game $G^{\varphi}_{(S,q^0)}$, then one can witness this fact by a strategy graph F with an exponentially bounded number of vertices. However, this does not necessarily mean that the representation of F is bounded exponentially in $|Pr| + |\varphi|$ since the size of states in F, in particular the size of messages in such states, might be big. Fortunately, it is possible to show that the overall size of F can be bounded exponentially, where the size of a strategy graph is defined in the obvious way according to some standard representation, where the set of all messages occurring in F are represented by a single DAG.

Lemma 3. *If F is a winning strategy graph for Player 0 in $G^{\varphi}_{(S,q^0)}$ as described in Lemma 2, then there exists a winning strategy graph F' of (overall) size exponentially bounded in $|Pr| + |\varphi|$.*

The proof of this lemma is provided in Section 8.6.

Using the three lemmas just stated, it immediately follows that PAMC(greedy, dssc-containing, $\mathcal{I}$-positive) is in NEXPTIME: By the lemmas, we know that Player 0 wins $G^{\varphi}_{(S,q^0)}$ iff there exists a winning strategy graph for Player 0 of size exponentially bounded in $|Pr| + |\varphi|$. So, we first guess such a graph and then check whether it represents a winning strategy graph for Player 0. The last step can be checked in polynomial time in the size of the graph (see, e.g., [11]). Hence, we have a non-deterministic exponential-time algorithm for the problem PAMC(greedy, dssc-containing, $\mathcal{I}$-positive).

In the following sections, we present the proofs of Lemma 1 to 3. However, first, in Section 8.2 and 8.3 we summarize some useful properties of concurrent game structures and parity games induced by protocols.

8.2 Properties of Concurrent Game Structures for Protocols

The following lemma says that $\prec$ as defined above is transitive, where for an instance P we write $\mathsf{root}(P)$ to denote the root of P.

Lemma 4. *Let S be defined as above and let q, q', q'' be states of S. If $q \prec q'$ and $q' \prec q''$, then $q \prec q''$.*

Proof. Let $q = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \overline{s}), q' = (\mathcal{K}', \overline{P}', \overline{\mathbf{m}}', \overline{s}'), q'' = (\mathcal{K}'', \overline{P}'', \overline{\mathbf{m}}'', \overline{s}'')$ be states of $S = \langle \Sigma, Q, \mathbb{P}, \pi, \Delta, \delta \rangle$ such that $q \prec q'$ and $q' \prec q''$. We have to show that $q \prec q''$, i.e., $q \not\sim q''$ and q'' is a descendant of q. Obviously, q'' is a descendant of q. Since $q \not\sim q'$ and $q' \not\sim q''$, one of the following cases must occur. From every case we can conclude that $q \not\sim q''$:

- $P_a \neq P'_a$ or $P'_a \neq P''_a$ for some $a \in \mathcal{H}$: By definition of S and since q' is a descendant of q and q'' is a descendant of q', it follows that $\mathsf{root}(P''_a)$ is a (proper) descendant of $\mathsf{root}(P_a)$ in P_a. In particular, $\mathsf{root}(P_a) \neq \mathsf{root}(P''_a)$. Hence, $q \not\sim q''$.
- $\mathcal{K} \neq \mathcal{K}'$ or $\mathcal{K}' \neq \mathcal{K}''$: Since q' is a descendant of q and q'' is a descendant of q' we know that $\mathcal{K} \subseteq \mathcal{K}' \subseteq \mathcal{K}''$. Thus, we can conclude that $\mathcal{K}$ is a strict subset of $\mathcal{K}''$. Thus, $q \not\sim q''$.

– $(s_c, d_c) \neq (s'_c, d'_c)$ or $(s'_c, d'_c) \neq (s''_c, d''_c)$ for some $c \in \mathsf{Sch}(\mathcal{H}, \mathcal{P}) \cap \mathsf{ch}(Pr)$ (note that $\mathsf{ch}(Pr) \cap \mathsf{Sch}(\mathcal{D}, \mathcal{P}) = \emptyset$ since protocols are dssc-free) and $P_a = P'_a = P''_a$ for all $a \in \mathcal{H}$: Since $c = \mathsf{sch}(a, b)$ for some honest $a \in \mathcal{H}$ and some $b \in \mathcal{P}$, and $P_a = P'_a = P''_a$ for all $a \in \mathcal{H}$, we know that nothing is written to any scheduled secure channel by a, and hence, $|s_c| \geq |s'_c| \geq |s''_c|$. If $|s_c| > |s''_c|$, we immediately obtain that $q \not\prec q''$. Otherwise, we have $|s_c = s''_c|$ which implies that $d'_c = d''_c = \overline{\mathsf{delivered}}$ and thus $d_c = \mathsf{delivered}$. So, $d_c \neq d''_c$. Thus, $q \not\prec q''$. □

While a computation in the concurrent game structure for a protocol can be infinite, "real progress", which is captured by relation $\prec$, can only be made a bounded number of times during such a computation. This fact is formally stated in the following lemma.

Lemma 5. *If $q_1 \prec \cdots \prec q_n$, then n is bounded polynomially w.r.t. the size of Pr.*

Proof. If $q \prec q'$, then, by definition, one of the following cases holds: q and q' differ on the state of (a) some honest participant, (b) a scheduled secure channel in $\mathsf{Sch}(\mathcal{H}, \mathcal{P})$, or (c) the intruder knowledge.

The lemma follows from the following observations: Each honest participant can change his state at most n times, where n is the size of the protocol description, so case (a) can happen only n^2 times. Moreover, each scheduled secure channel in $\mathsf{Sch}(\mathcal{H}, \mathcal{P})$ receives only n messages (from honest participants) during the course of a protocol execution, so its state can be changed at most $2n$ times. It means that case (b) can happen at most $2n^2$ times. Now, whenever a state of the intruder is changed, the state of some honest participant or some secure channel has to be changed as well. So, if (c) happens, then so must (a) or (b). □

The following lemma follows immediately from the definition of a consuming state and greedy principals (recall that principals considered here are greedy).

Lemma 6. *If a state q is consuming and q' is a successor of q, then $q' \prec q$.*

The next lemma formalizes the already mentioned intuition behind non-consuming states (see Section 8.1): When in a non-consuming state, an instance ignores incoming messages, and hence, two equivalent, non-consuming states have the same set of successors.

Lemma 7. *Let q_1, q_2 be non-consuming states with $q_1 \sim q_2$. If a state q is a c-successor of q_1, for $c \in \Delta^A(q_1)$ and some $A \subseteq \Sigma$, then $c \in \Delta^A(q_2)$ and q is also a c-successor of q_2.*

Proof. Since $q_1 \sim q_2$, we have $q_1 = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}_1, \overline{s})$ and $q_2 = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}_2, \overline{s})$. It is enough to show that if $c \in \Delta^\Sigma(q_1)$, then $c \in \Delta^\Sigma(q_2)$ and $\delta(q_1, c) = \delta(q_2, c)$.

The set of moves of $\mathcal{I}$ depends only on the intruder knowledge ($\mathcal{K}$) which is the same in q_1 and q_2 and the result of applying these moves has the same consequences.

The set of moves of the secure channels depends only on $\overline{s}$ which, again, is the same in q_1 and q_2. Thus, these players have the same moves in q_1 and q_2, and the result of applying these moves has again the same consequences.

The states of an honest participant in q_1 and q_2 are the same, and, because the states are non-consuming, only non-consuming rules (and exactly those) can be applied. But the set of these rules is the same in q_1 and q_2 and the application of these rules does not depend on the current input. Hence, the result of applying these moves has the same consequences. □

8.3 Some Properties of Parity Games for Protocols

Given Pr, $S = S_{Pr}$, q^0 (the initial state of S), and an $\mathcal{I}$-positive AMC-formula φ in negation normal form as above, in this section we study the induced parity game $G^{\varphi}_{(S,q^0)}$. We also state some general properties of parity games. We first note:

Remark 1. For each subformula $\langle\!\langle A \rangle\!\rangle \bigcirc \psi$ of φ, we have that $\mathcal{I} \in A$ and for each subformula $[\![A]\!] \bigcirc \psi$ of φ, we have that $\mathcal{I} \notin A$. Also, for each subformula $\Box_c \psi$ of φ, the domain of c contains $\mathcal{I}$ and for each subformula $\Diamond_c \psi$ of φ, the domain of c does not contain $\mathcal{I}$.

The following lemma is a consequence of Lemma 7.

Lemma 8. *If $(q_1, \Diamond_c \psi)$ and $(q_2, \Diamond_c \psi)$ are vertices of $G^{\varphi}_{(S,q^0)}$ where q_1 and q_2 are non-consuming, $q_1 \sim q_2$, and such that (q, ψ) is a successor of $(q_1, \Diamond_c \psi)$, then (q, ψ) is also a successor of $(q_2, \Diamond_c \psi)$. Similarly, if $(q_1, \Box_c \psi)$ and $(q_2, \Box_c \psi)$ are vertices of $G^{\varphi}_{(S,q^0)}$ with non-consuming states q_1 and q_2, $q_1 \sim q_2$, and (q, ψ) is a successor of $(q_1, \Box_c \psi)$, then (q, ψ) is also a successor of $(q_2, \Box_c \psi)$.*

From Lemma 4 and 5, we obtain:

Lemma 9. *If λ is a play in $G^{\varphi}_{(S,q^0)}$, then λ can be written as a concatenation $\lambda_1 \dots \lambda_n$ where*

- *λ_j for $j = 1, \dots, n-1$ is a finite sequence of states of $G^{\varphi}_{(S,q^0)}$ and λ_n is an infinite sequence of states of $G^{\varphi}_{(S,q^0)}$,*
- *n can be polynomially bounded in the size of Pr,*
- *for all i and states v, u in λ_i we have that $v \sim u$, and*
- *for all $i < j$, u in λ_i, and v in λ_j, we have $u \prec v$.*

Proof. First observe that if $(q_1, \psi_1), (q_2, \psi_2), \dots$ is a play, then, for $i < j$, the state q_j is a descendant of q_i, and so, either $q_i \sim q_j$ or $q_i \prec q_j$. Now, the lemma easily follows from Lemma 4 and 5. □

We now summarize some useful properties of parity games, independent of the particular game $G^{\varphi}_{(S,q^0)}$.

Lemma 10. *Let λ be an infinite play in some parity game G. Assume that λ is the concatenation $\lambda_1\lambda_2\dots$ where each λ_i is a non-empty sequence of vertices such that, for each index i, the maximal color occurring in λ_i is even (odd). Then the maximal color occurring in λ infinitely often is even (odd).*

Proof. Let a be the maximal color occurring infinitely often in λ. Because the set of colors is finite, there is an index i_0 such that, for each $i > i_0$, no color $a' > a$ occurs in λ_i. Clearly, there is $j > i_0$ such that a occurs in λ_j. So, a is the maximal color occurring in λ_j which, by assumption, is even (odd). □

Before we proceed, we need to introduce some terminology.

Let $G = (V, V_0, V_1, E, v_I, l)$ be a parity game. A *(finite) path* π in G is a finite sequence of the form $v_1, \dots, v_n$ such that $(v_i, v_{i+1}) \in E$ for every $i = 1, \dots, n-1$. We say that a vertex $v \in V$ is *reachable* in G if there exists a path from v_I to v in G. For $U \subseteq V$, we call $\pi = v_1, \dots, v_n$ a *(finite) U-path* if π is a path and $v_i \in U$ for every $i = 1, \dots, n$. An *infinite path* λ in G is a finite sequence of the form $v_1, v_2, \dots$ such that $(v_i, v_{i+1}) \in E$ for every $i \geq 1$. We call λ winning for Player 0 if the maximum color occurring infinitely often in λ is even; otherwise λ is winning for Player 1. We call λ an *(infinite) U-path* if λ is an infinite path and $v_i \in U$ for every $i \geq 1$.

Let $U \subseteq V$ and let $f : U \to V$ be a function. We call f *consistent* with a path $v_1, \dots, v_n$ in G if $v_{i+1} = f(v_i)$ for every $i = 1, \dots, n-1$ with $v_i \in U$; analogously for infinite paths.

Definition 2. *Consider a parity game (V, V_0, V_1, E, v_I, l). For a set $U \subseteq V$, a* U-strategy for Player 0 *is a function $f : U \cap V_0 \to V$ such that if $f(v) = v'$, then $(v, v') \in E$, and each infinite U-path consistent with f and starting with a state reachable from v_I is winning for Player 0.*

Definition 3. *Let f be a U-strategy for Player 0, $U \subseteq V$, and $D \subseteq \mathrm{dom}(f)$. We say that f* gives a good choice for D in a vertex $v \in D$, *if for each U-path $\pi = v_1, \dots, v_n$ in G consistent with f such that $v_1 = f(v)$, $v_n \in D$, and $v_i \in U \setminus D$, for every $1 \leq i < n$, the maximal color occurring in π is even.*

Lemma 11. *Let (V, V_0, V_1, E, v_I, l) be a parity game, $U \subseteq V$, f be a U-strategy for Player 0, and D be a non-empty subset of* $\mathrm{dom}(f)$. *Then, there exists a vertex $v \in D$ such that f gives a good choice for D in v.*

Proof. For the sake of contradiction, suppose that there is no vertex $v \in D$ such that f gives a good choice for D in v. Let v_0 be some element of D (recall that D is non-empty).

Because f does not give a good choice in v_0, there is a U-path π_1 consistent with f, starting in $f(v_0)$, and ending in some vertex $v_1 \in D$ such that the maximal color occurring in π_1 is odd. Because f does not give a good choice in v_1 we can repeat the argument and obtain a U-path π_2 consistent with f, starting with $f(v_1)$, ending in some vertex $v_2 \in D$ such that the maximal color occurring in π_2 is odd, and so on. In this way, we obtain an infinite path $\lambda = \pi_1\pi_2 \cdots \in U^\omega$ consistent with f such that the maximal color on λ_i is odd, for every i. By Lemma

10, it follows that λ is winning for Player 1, in contradiction to the assumption that f is a U-strategy for Player 0. $\square$

8.4 Proof of Lemma 1

Assume that there exists a winning strategy f in $G^{\varphi}_{(S,q^0)}$ for Player 0. By Fact 1, we may assume that this winning strategy is memoryless. Starting with f, we will construct a $\sim$-uniform winning strategy for Player 0.

For $v \in V$, let $[v] = [v]_\sim = \{v' \in V \mid v \sim v'\}$ denote the equivalence class of v (w.r.t. $\sim$). We define $V/_\sim = \{[v]_\sim \mid v \in V\}$ to be the set of equivalence classes of V. For each $\rho \in V/_\sim$, let $f_\rho : \rho \cap V_0 \to V$ be the restriction of f to ρ, i.e., $f_\rho(v) = f(v)$, for each $v \in \mathrm{dom}(f_\rho) = \rho \cap V_0$. (Note that f_ρ is a ρ-strategy for Player 0.)

The outline of the proof is as follows: For each $\rho \in V/_\sim$, we will construct $f'_\rho : \rho \cap V_0 \to V$ such that the function f' defined by $f'(v) = f'_{[v]}(v)$, for every $v \in V_0$, is a $\sim$-uniform winning strategy for Player 0.

For a (possibly non-standard) subformula ψ of φ, we define the set $D_\rho(\psi) = \{v \in \rho \mid v = (q, \psi) \text{ for some non-consuming } q\} \subseteq \rho$.

We say that a function $g_\rho : \rho \cap V_0 \to V$ is $\sim$-uniform w.r.t. a set $D \subseteq \rho \cap V_0$, if for each $v, v' \in D$, (a) and (b) of Definition 1 hold true for g_ρ.

Now, we construct f'_ρ from f_ρ by iteratively performing the two steps described below, until this is not possible anymore. More precisely, let $f^0_\rho = f_\rho$. Given f^i_ρ, we obtain $f^{i+1}_\rho : \rho \cap V_0 \to V$ either by applying step A or step B below. In the construction we use the fact that each f^i_ρ is a ρ-strategy for Player 0 (see Definition 2), which will be proven later.

A. Pick a subformula of φ of the form $\Diamond_c \psi$ such that f^i_ρ is not $\sim$–uniform w.r.t. $D = D_\rho(\Diamond_c \psi)$. Note that f^i_ρ is an ρ-strategy for Player 0. Note also that D is a non-empty subset of $\mathrm{dom}(f^i_\rho)$. Thus, by Lemma 11, there exists a vertex $\tilde{v} \in D$ such that f^i_ρ gives a good choice for D in $\tilde{v}$. We define f^{i+1}_ρ as follows: $f^{i+1}_\rho(v) = f^i_\rho(\tilde{v})$, if $v \in D$, and $f^{i+1}_\rho(v) = f^i_\rho(v)$, otherwise.

B. Pick a subformula of φ of the form $\langle\!\langle A \rangle\!\rangle \bigcirc \psi$ such that f^i_ρ is not $\sim$–uniform w.r.t. $D = D_\rho(\langle\!\langle A \rangle\!\rangle \bigcirc \psi)$. Note that f^i_ρ is an ρ-strategy for Player 0. Note also that D is a non-empty subset of $\mathrm{dom}(f^i_\rho)$. Thus, by Lemma 11, there exists a node $\tilde{v} = (\tilde{q}, \langle\!\langle A \rangle\!\rangle \bigcirc \psi) \in D$ such that f^i_ρ gives a good choice for D in $\tilde{v}$. We define f^{i+1}_ρ as follows: Let $(\tilde{q}, \Box_{\tilde{c}} \psi) = f^i_\rho(\tilde{v})$. If $v = (q, \langle\!\langle A \rangle\!\rangle \bigcirc \psi) \in D$, we set $f^{i+1}_\rho(v) = (q, \Box_{\tilde{c}} \psi)$, and if $v \notin D$, we set $f^{i+1}_\rho(v) = f^i_\rho(v)$.

It is easy to show that if f^i_ρ is $\sim$–uniform w.r.t. $D_\rho(\langle\!\langle A \rangle\!\rangle \bigcirc \psi)$ (or $D_\rho(\Diamond_c \psi)$), then f^{i+1}_ρ is also $\sim$-uniform w.r.t. this set. Moreover, if f^{i+1}_ρ is obtained by step B, for some $\langle\!\langle A \rangle\!\rangle \bigcirc \psi$, then f^{i+1}_ρ is $\sim$–uniform w.r.t. the set $D_\rho(\langle\!\langle A \rangle\!\rangle \bigcirc \psi)$. Hence, because the number of distinct subformulas of φ of this form is bounded by $|\varphi|$, this step can be done at most $|\varphi|$ times. Similarly, if f^{i+1}_ρ is obtained by step A, for some $\Diamond_c \psi$, then f^{i+1}_ρ is $\sim$–uniform w.r.t. the set $D_\rho(\Diamond_c \psi)$. The number of subformulas of φ of the form $\Diamond_c \psi$ is bounded by $O(|\varphi| \cdot 2^n)$, where n is the size of

Pr. Here we use that φ is $\mathcal{I}$-positive, and hence, $\mathcal{I} \notin \mathrm{dom}(c)$ (see Remark 1), and each participant, except for I, has a bounded number of available moves in each state. Consequently, this step can also only be performed a bounded number of times. Thus, after a bounded number of steps, say k steps, we obtain f^k_ρ which is $\sim$–uniform w.r.t $D_\rho(\psi)$, for each ψ of the form $\langle\!\langle A \rangle\!\rangle \bigcirc \psi'$ or $\Diamond_c \psi'$. We define $f'_\rho = f^k_\rho$.

Now, we prove that each f^i_ρ is a ρ-strategy for Player 0. We proceed by induction on i. In case $i = 0$, we have $f^0_\rho = f_\rho$ and from the definition of f_ρ it follows that f^0_ρ is a ρ-strategy for Player 0. So, suppose that f^i_ρ is a ρ-strategy for Player 0. We will show that f^{i+1}_ρ is a ρ-strategy for Player 0 as well.

First, it is easy to show that if $f^{i+1}(v) = v'$, then $(v, v') \in E$ (if f^{i+1} is obtained by Step B, then it follows from Lemma 7 and the definition of $G^{\varphi}_{(S,q^0)}$; if f^{i+1} is obtained by Step A, then we use Lemma 8).

Now, suppose that λ is an infinite ρ-path consistent with f^{i+1}_ρ and starting with a vertex reachable from the initial state of $G^{\varphi}_{(S,q^0)}$. We consider two cases: (1) There is a suffix λ' of λ such that λ' does not contain vertices in D. In this case, λ' is consistent with f^i_ρ. Thus λ' it is winning for Player 0 and so is λ. (2) λ contains an infinite number of elements in D. In this case we can split λ into $\lambda_0\lambda_1\ldots$ such that the last element of λ_i is the only one in λ_i belonging to D. Let $k > 0$. Let $\lambda_k = v_1, \ldots, v_n$ and v_0 be the last element of λ_{k-1}. So, $v_0, v_n \in D$ and $v_i \notin D$, for $0 < i < n$. Now we consider two cases, depending on whether f^{i+1}_ρ was obtain by step A or B:

- If f^{i+1}_ρ was obtained by step A, then $v_1 = f^i_\rho(\tilde{v})$ is the successor of $\tilde{v}$ for which f^i_ρ gives a good choice. By definition of a good choice for D, the maximal color occurring in $\lambda_k = v_1, \ldots, v_n$ is even.
- If f^{i+1}_ρ was obtained by step B, then v_1 is of the form $(q, \Box_{\tilde{c}}\psi)$, where $(\tilde{q}, \Box_{\tilde{c}}\psi) = f^i_\rho(\tilde{v})$ is the successor of $\tilde{v}$ for which f^i_ρ gives a good choice for D. Because v_2 is a successor of $(q, \Box_{\tilde{c}}\psi)$ and $q, \tilde{q} \in \rho$ are non-consuming, by Lemma 8, we have that v_2 is also a successor of $(\tilde{q}, \Box_{\tilde{c}}\psi)$. Hence, the path $\tilde{v}, (\tilde{q}, \Box_{\tilde{c}}\psi), v_2, \ldots, v_n$ is consistent with f^i_ρ. Note also that $(\tilde{q}, \Box_{\tilde{c}}\psi)) \notin D$. So, by the definition of a good choice, the maximal color on $(\tilde{q}, \Box_{\tilde{c}}\psi), v_2, \ldots, v_n$ is even. Because the colors of $(\tilde{q}, \Box_{\tilde{c}}\psi)$ and $(q, \Box_{\tilde{c}}\psi)$ are the same, the maximal color on $\lambda_k = v_1, \ldots, v_n$ is also even.

So, in the both cases, we have proven that the maximal color on λ_k is even. Thus, by Lemma 10, the path $\lambda_1\lambda_2\ldots$ is winning for Player 0 and so is λ.

This shows that f^{i+1}_ρ is a ρ-strategy for Player 0. In particular, f'_ρ is a ρ-strategy for Player 0. We define $f'(v) = f'_{[v]}(v)$ for every $v \in V_0$. It remains to show that f' is a $\sim$-uniform winning strategy for Player 0.

We know that f'_ρ is $\sim$-uniform w.r.t $D_\rho(\psi)$, for each ψ of the form $\langle\!\langle A \rangle\!\rangle \bigcirc \psi'$ or $\Diamond_c \psi'$. From this it is easy to conclude that f' is $\sim$-uniform.

To prove that f' is winning for Player 0, let us consider some play λ consistent with f'. By Lemma 9, there is a suffix λ' of λ such that λ' is an infinite ρ-path for some equivalence class ρ. Because λ' is consistent with f' and, for each $v \in \rho$,

we have that $f'(v) = f'_\rho(v)$, the infinite path λ' is consistent with f'_ρ as well. Since f'_ρ is an ρ-strategy, we can conclude that λ' is winning for Player 0, and hence, so is λ. □

8.5 Proof of Lemma 2

Given Pr, $S = S_{Pr}$, q^0 (the initial state of S), and an $\mathcal{I}$-positive AMC-formula in negation normal form as above, and $G^{\varphi}_{(S,q^0)}$ in this section we proof Lemma 2.

We first show that the branching degree of $G^{\varphi}_{(S,q^0)}$ is bounded exponentially. This is true independent of whether or not the underlying strategy is $\sim$-uniform.

Lemma 12. *The branching degree of a strategy graph of Player* 0 *of* $G^{\varphi}_{(S,q^0)}$ *is exponentially bounded* $|Pr|$.

Proof. By the definition of $G^{\varphi}_{(S,q^0)}$, the only vertices of $G^{\varphi}_{(S,q^0)}$ which can have more than two successors are of the form $(q, \langle\langle A\rangle\rangle \bigcirc \psi)$, $(q, [\![A]\!] \bigcirc \psi)$, $(q, \Diamond_c \psi)$, and $(q, \Box_c \psi)$. Vertices of the form $(q, \langle\langle A\rangle\rangle \bigcirc \psi)$ and $(q, \Diamond_c \psi)$ belong to V_0, so in any strategy graph for Player 0 these vertices have exactly one successor. Hence, it suffices to check that the number of successors in $G^{\varphi}_{(S,q^0)}$ of vertices of the form $(q, [\![A]\!] \bigcirc \psi)$ and $(q, \Box_c \psi)$ is exponentially bounded in $|Pr|$.

Let us first consider the case of $v = (q, [\![A]\!] \bigcirc \psi)$. Because φ is $\mathcal{I}$-positive, we know that $\mathcal{I} \notin A$. Hence, A contains only some honest principals and some secure channels. Each successor of v corresponds to some combination of moves of players in A, so the number of successors of v is $|\Delta^A_q|$. Since the number of moves available to each honest principal and to each secure channel is linear in $|Pr|$, $|\Delta^A_q|$ is exponentially bounded in $|Pr|$.

Now, let us consider the case of $v = (q, \Box_c \psi)$. Because φ is $\mathcal{I}$-positive, we have that $\mathcal{I} \in \mathrm{dom}(c)$. Each successor of v corresponds to some c-successor of q, i.e. to some combination of moves of players in $B = \Sigma \setminus \mathrm{dom}(c)$. Hence, one can identify every such successor with exactly one element of Δ^B_q. From the previous case, we know already that $|\Delta^B_q|$ is exponentially bounded in $|Pr|$. □

The following lemmas states that in a path in $G^{\varphi}_{(S,q^0)}$ one cannot stay long in a state with the same first component, i.e., the same state of S, without repeating the second component, i.e., the subformula of φ.

Lemma 13. *For every path* $(q, \psi_1), \ldots, (q, \psi_n)$ *in* $G^{\varphi}_{(S,q^0)}$ *with* $\psi_i \neq \psi_j$, *for every* $i, j \in \{1, \ldots, n\}$, $i \neq j$, *it holds that* $n \leq 2|\varphi| + 1$.

Proof. If ψ_i is non-standard, then ψ_{i+1} is standard (by the definition of $\mathrm{Sub}^q_S(\varphi)$, symbols $\Box_c$ and $\Diamond_c$ are not nested). Thus, at least $\llcorner n/2 \lrcorner$ formulas amongst $\psi_1 \ldots, \psi_n$ are standard. Because there is at most $|\varphi|$ standard subformulas of φ, we can conclude that $n \leq 2|\varphi| + 1$. □

The following lemma states properties of paths consisting of equivalent states.

Lemma 14. *Let $v_1, \ldots, v_n$ be a path in $G^{\varphi}_{(S,q^0)}$ such that $v_i \sim v_j$, for every $i, j \in \{1, \ldots, n\}$. Then:*

1. *For each $i, j \in \{1, \ldots, n-1\}$, if $v_i = (q_i, \Box_c \psi)$ and $v_j = (q_j, \Box_c \psi)$, then $v_{i+1} = v_{j+1}$.*
2. *For each $i, j \in \{1, \ldots, n-2\}$, if $v_i = (q_i, [\![A]\!] \circ \psi)$ and $v_j = (q_j, [\![A]\!] \circ \psi)$, then $v_{i+1} = (q_i, \Diamond_c \psi)$ and $v_{j+1} = (q_j, \Diamond_c \psi)$, for some c.*

Proof. In both cases, the choices made in v_i and v_j represent choices of some honest principals and some scheduled secure channels, but not choices of the intruder. These choices cannot change the state of these agents: in case 1, this is because $v_i \sim v_{i+1}$ and $v_j \sim v_{j+1}$, and in case 2, it is because $v_i \sim v_{i+2}$ and $v_j \sim v_{j+2}$. So, the choices are uniquely determined and correspond to the choice of staying in the same state for honest players, and to the choice of not delivering any message by the scheduled secure channels (recall that since the protocol is assumed to be dssc-free, these scheduled secure channels only get their messages from honest principals). □

In what follows, we call a vertex v of $G^{\varphi}_{(S,q^0)}$ *modal* if it is of the form $(q, \langle\!\langle A \rangle\!\rangle \circ \psi)$ or $(q, [\![A]\!] \circ \psi)$.

Lemma 15. *Let $v_1, \ldots, v_n$ be a path in a $\sim$-uniform strategy graph for Player 0 in the game $G^{\varphi}_{(S,q^0)}$ such that $v_i \sim v_j$, for every $i, j \in \{1, \ldots, n\}$, and the vertices $v_1, \ldots, v_n$ are non-consuming. Let, for some $i, j \in \{1, \ldots, n-2\}$, the vertices v_i and v_j be modal and of the form $v_i = (q_i, \psi)$ and $v_j = (q_j, \psi)$. Then, we have $v_{i+2} = v_{j+2}$.*

Proof. We consider two cases:

Case 1: $\psi = \langle\!\langle A \rangle\!\rangle \circ \psi'$. By the definition of $\sim$-uniform strategy, there exists c such that $v_{i+1} = (q_i, \Box_c \psi)$ and $v_{j+1} = (q_j, \Box_c \psi)$. Thus, by Lemma 14, we obtain $v_{i+2} = v_{j+2}$.

Case 2: $\psi = [\![A]\!] \circ \psi'$. By Lemma 14, there exists c such that $v_{i+1} = (q_i, \Diamond_c \psi)$ and $v_{j+1} = (q_j, \Diamond_c \psi)$. Thus, we obtain $v_{i+2} = v_{j+2}$ by the definition of $\sim$-uniform strategy. □

Let H be a $\sim$-uniform strategy graph of Player 0 in $G^{\varphi}_{(S,q^0)}$. A path $v_1, \ldots, v_n$ in H is *conservative*, if, for all $i \neq j$ we have that $v_i \neq v_j$ and $v_i \sim v_j$.

Lemma 16. *Let $\pi = v_1, \ldots, v_n$ be a conservative path in a $\sim$–uniform strategy graph of Player 0 in the game $G^{\varphi}_{(S,q^0)}$. If v_1 is consuming, then $n \leq 2|\varphi| + 1$.*

Proof. Assume that v_i is of the form (q_i, ψ_i) for all $i = 1, \ldots, n$. Let k be maximal such that $q_i = q_1$ for all $i \leq k$. By Lemma 13, $k \leq 2|\varphi| + 1$. We will show that $n = k$, which gives $n \leq 2|\varphi| + 1$. By the sake of contradiction, suppose that $k < n$. So, $q_{k+1} \neq q_k$, and thus q_{k+1} must be a successor of q_k. By the assumption, the state $q_k = q_1$ is consuming, so by Lemma 6, we have $q_k \prec q_{k+1}$, which contradicts the assumption that π is conservative. □

Lemma 17. *Let $v_1, \ldots, v_n$ be a conservative path in a $\sim$–uniform strategy graph for Player 0 in the game $G^{\varphi}_{(S,q^0)}$. If $v_1, \ldots, v_n$ are non-consuming, then the number of modal vertices in $v_1, \ldots, v_n$ is bounded by $|\varphi| + 2$.*

Proof. We will show that a modal subformula of φ cannot occur twice in the path $v_1, \ldots, v_{n-2}$, which means that the number of modal vertices in $v_1, \ldots, v_{n-2}$ is bounded by the number of distinct modal subformulas of φ, and hence, is $\leq |\varphi|$. Consequently, the number of modal vertices in $v_1, \ldots, v_n$ is bounded by $|\varphi| + 2$.

Suppose that, for $i, j \leq n-2$ we have $v_i = (q_i, \psi)$ and $v_j = (q_j, \psi)$, for a modal formula ψ. By Lemma 15, $v_{i+2} = v_{j+2}$, which contradicts the assumption that $v_1, \ldots, v_n$ is conservative. □

Lemma 18. *Let $\pi = v_1, \ldots, v_n$ be a conservative path in a $\sim$–uniform strategy graph of Player 0 in the game $G^{\varphi}_{(S,q^0)}$. If $v_1, \ldots, v_n$ are non-consuming, then $n \leq p(|\varphi|)$ for some fixed polynomial p in $|\varphi|$.*

Proof. We split π into $\pi_0, \ldots, \pi_m$ such that for each u, v in π_i the first components of u and v are the same and, if $v = (q_v, \psi_v)$ is the last element of π_i and $u = (q_u, \psi_u)$ is the first element of π_{i+1}, then $q_v \neq q_u$.

For $0 \leq i < m$, the second component of the last element of π_i has to be a non-standard formula. Moreover, for $0 < i \leq m$, the second component of the first element of π_i is standard. Hence, for $0 < i < m$, π_i has at least two elements. A predecessor of a vertex with a non-standard formula is modal, so each π_i, for $0 < i < m$, contains a modal element. Hence, by Lemma 17, we obtain $m \leq |\varphi| + 4$. By Lemma 13, the length of each π_i is bounded by $2|\varphi| + 1$, so we conclude that $n \leq (2|\varphi| + 1)(|\varphi| + 4) =: p(|\varphi|)$. □

Now, we are ready to prove Lemma 2. First, by Lemma 12, the branching degree of a strategy graph of Player 0 is exponentially bounded. Below, we show that every path in $G^{\varphi}_{(S,q^0)}$ starting from the initial state of $G^{\varphi}_{(S,q^0)}$ without repetitions has length polynomially bounded in $|Pr| + |\varphi|$. This shows the first part of Lemma 2 and from this, together with the bounded branching degree, the lemma follows.

Let π be a path without repetitions in a $\sim$–uniform strategy graph of Player 0. Let $v_1, \ldots, v_n$ be a subsequence of π such that $v_i \sim v_j$, for $1 \leq i, j \leq n$. By Lemma 9, the number of maximal subsequences of π of this type is at most polynomial in $|Pr|$. Thus, to prove a polynomial bound on the length of π, it is enough to show that n is polynomially bounded in $|Pr| + |\varphi|$.

Observe that $v_1, \ldots, v_n$ is conservative. Thus, if $v_1, \ldots, v_n$ does not contain any consuming vertex, then, by Lemma 18, n is polynomially bounded. Otherwise, let k be the smallest index such that v_k is consuming. By Lemma 18, k is polynomially bounded in $|\varphi|$ and, by Lemma 16, $n - k$ is polynomially bounded in $|\varphi|$ as well. Hence, we obtain a polynomial bound on n as desired. □

8.6 Proof of Lemma 3

Given Pr, $S = S_{Pr}$, q^0 (the initial state of S), and an $\mathcal{I}$-positive AMC-formula φ in negation normal form as above, and $G^{\varphi}_{(S,q^0)}$ in this section we proof Lemma 3.

To do so, we need to bound the size of messages used in a winning strategy graph. The main idea is to (iteratively) replace certain (unnecessarily big) messages by new atoms in such a way that the resulting graph is still a winning strategy graph. For this purpose, we first characterize the set $d(E)$ of terms derivable from E in terms of what we call intruder rules and study how the replacement of terms by other terms effects the derivability of terms.

For a term t the set $\mathrm{Sub}(t)$ of *subterms* of t is defined as usual. We extend $\mathrm{Sub}(\cdot)$ to sets of terms, multi terms, a-rules and a-instance for $a \in \mathcal{H}$, and protocols as expected.

The intruder rules that we use include those introduced in [22]. In addition, we need rules for hashing, signatures, and generating new atoms. In what follows, we often write E, m and m, m' instead of $E \cup \{m\}$ and $\{m, m'\}$, respectively.

An intruder rule L is of the form $E \to m$ where E is a finite set of messages and m is a message. A rule of this form is also called m-rule since m is generated. Given a set E', L can be *applied to* E' if $E \subseteq E'$. The rule L induces a binary relation $\to_L$ on finite sets of messages: $\to_L = \{(E', E' \cup \{m\}) \mid L \text{ can be applied to } E'\}$. If $\mathcal{L}$ is a set of intruder rules, then $\to_{\mathcal{L}} = \bigcup_{L \in \mathcal{L}} \to_L$. For a binary relation $\to$ we write $E \to E'$ instead of $\to (E, E')$. The reflexive and transitive closure of $\to$ is denoted by $\to^*$.

To characterize $d(E)$, we consider the following set of intruder rules. In what follows, the notion "intruder rule" will always refer to the rules introduced below. This set is partitioned into decomposition and composition rules. Accordingly, we call a rule decomposition and composition rule, respectively.

Decomposition rules are of one of the following forms, where m and m' are messages and $k \in \mathbb{K}$ (and thus, $k^{-1} \in \mathbb{K}$):

1. $\langle m, m' \rangle \to m$ and $\langle m, m' \rangle \to m'$.
2. $\{m\}^s_{m'}, m' \to m$.
3. $\{m\}^a_k, k^{-1} \to m$.

Composition rules are of one of the following forms, where m, m' are some messages, $k, k_0, k_1, k_2 \in \mathbb{K}$, and $a_I \in \mathcal{A}_I$:

1. $m, m' \to \langle m, m' \rangle$.
2. $m, m' \to \{m\}^s_{m'}$.
3. $m, k \to \{m\}^a_k$.
4. $m \to \mathsf{hash}(m)$.
5. $m, k^{-1} \to \mathsf{sig}(k, m)$.
6. $\to a_I$.

Let $\mathcal{L}$ denote the set of all (composition and decomposition) rules. It is easy to see that

$$d(E) = \bigcup \{E' \mid E \to^*_{\mathcal{L}} E'\}.$$

A *derivation* is of the form $E_0 \to_{L_0} E_1 \to_{L_1} E_2 \to_{L_2} \cdots \to_{L_{n-1}} E_n$ where $E_i \to_{L_i} E_{i+1}$ for every i. We call n the length of the derivation. We know that for every $m \in d(E)$ there exists n, intruder rules $L_0, \ldots, L_{n-1}$, and sets $E_0, \ldots, E_n$ such that $E_0 = E$, $m \in E_n$, and there is a derivation from E_0 to E_n

as above. We call such a derivation a derivation for m of length n. The derivation is *minimal* if no step can be removed such that the resulting sequence is still a derivation for m. Clearly, for every $m \in d(E)$ there exists a minimal derivation. We write $m \in d^c(E)$ if there exists a minimal derivation of m where the last rule is a composition rule. The following fact is well-known (see, e.g., [22]).

Lemma 19. *Let $m \in d(E)$ and let D be a minimal derivation of m from E such that D ends with a decomposition rule. Then, $m \in Sub(E)$.*

From this lemma, we obtain:

Lemma 20. *Let E be a set of messages and let τ be a message such that $\tau \notin Sub(E)$ and $\tau \notin d^c(E)$. Then for all $m \in d(E)$ we have that $\tau \notin Sub(m)$.*

Proof. Let $D = E_0 \to_{L_1} E_1 \cdots \to_{L_n} E_n$ be a derivation of m from $E_0 = E$. Assume, for the purpose of contradiction, that $\tau \in \mathrm{Sub}(m)$. Then, there exists a minimal $i \neq 0$ such that $\tau \in \mathrm{Sub}(E_i)$ since τ is a subterm of E_n. Assume that L_i is an s-rule for some s. Then, τ is a subterm of s. If τ is a proper subterm of s, by the definition of intruder rules, it follows that τ is a subterm of E_{i-1}, in contradiction to the minimality of i. Thus, $\tau = s$ and therefore, $\tau \in d(E)$. Since $\tau \notin d^c(E)$ it follows that all derivations of τ end with a decomposition rule. Hence, by Lemma 19, $\tau \in \mathrm{Sub}(E)$, in contradiction to the assumption that $\tau \notin \mathrm{Sub}(E)$. Hence, $\tau \notin \mathrm{Sub}(m)$.

Let

$$\textsc{Derive} = \{(E, m) \mid m \in d(E)\}$$

where E and m are given as DAGs be the *derivation problem*. The following is well-known (see, e.g., [8]):

Lemma 21. Derive *can be decided in polynomial time.*

We now study which messages can be derived from a set of messages if certain terms are replaced by other terms.

Definition 4. *Let t, t' and t'' be terms. By $t_{|t' \to t''}$ we denote the term obtained from t by simultaneously replacing any occurrence of t' in t by t''.*

For a set T of terms we define $T_{|t' \to t''} = \{t_{|t' \to t''} \mid t \in T\}$. For a sequence $s = t_1 \cdots t_n$ of terms the sequence $s_{|t' \to t''}$ is defined by $s_{|t' \to t''} = t_{1|t' \to t''} \cdots t_{n|t' \to t''}$. For a substitution σ we denote by $\sigma_{|t' \to t''}$ the substitution σ' with the same domain as σ and $\sigma'(x) = \sigma(x)_{|t' \to t''}$ for all $x \in \mathrm{dom}(\sigma)$. For a multi message $\mathbf{m} : A \to \mathcal{M}_\circ$ for some $A \subseteq \mathcal{C}$, we denote by $\mathbf{m}_{|t' \to t''}$ the multi message $\mathbf{m}' : A \to \mathcal{M}_\circ$ with $\mathbf{m}'(c) = \mathbf{m}(c)_{|t' \to t''}$ for all $c \in A$. For $C, D \subseteq \mathcal{C}$ and a (C, D)-rule $R = \mathbf{r} \Rightarrow \mathbf{s}$ the rule $R_{|t' \to t''}$ is defined by $R_{|t' \to t''} = \mathbf{r}_{|t' \to t''} \Rightarrow \mathbf{s}_{|t' \to t''}$. For a principal $P = (V, E, r, \lambda, l)$ the principal $P_{|t' \to t''}$ is defined by $P_{|t' \to t''} = (V, E, r, \lambda, l')$ where for $(v, v') \in E$ the label $l'(v, v')$ is defined by $l'(v, v') = l(v, v')_{|t' \to t''}$. For a state $q = (\mathcal{K}, \overline{P}, \overline{\mathbf{m}}, \{(s_c, d_c)\}_{c \in \mathcal{SC}})$ of S_{Pr} we define

$$q_{|t' \to t''} = (\mathcal{K}_{|t' \to t''}, \{P_{a|t' \to t''}\}_{a \in \mathcal{H}}, \{\mathbf{m}_{a|t' \to t''}\}_{a \in \mathcal{H}}, \{(s_{c|t' \to t''}, d_c)\}_{c \in \mathcal{SC}}) \ .$$

For a formula ψ of the form $\Diamond_c\psi'$ or $\Box_c\psi'$, we set $\psi_{|t'\to t''} = \Diamond_{c'}\psi'$ or $\psi_{|t'\to t''} = \Box_{c'}\psi'$, respectively, where $c'(\mathcal{I}) = c(\mathcal{I})_{|t'\to t''}$, in case $\mathcal{I} \in \mathrm{dom}(c)$, and $c'(a) = c(a)$ for $a \in \mathrm{dom}(c) \setminus \{\mathcal{I}\}$. If ψ is not of the form $\Diamond_c\psi'$ or $\Box_c\psi'$, we define $\psi_{|t'\to t''} = \psi$. For a state $\alpha = (q, \psi)$ of $G^{\varphi}_{(S,q^0)}$ we set $\alpha_{|t'\to t''} = (q_{|t'\to t''}, \psi_{|t'\to t''})$. For a subgraph F of $G^{\varphi}_{(S,q^0)}$ the graph $F_{|t'\to t''}$ is defined in the obvious way.

The following lemma was proved in [13].

Lemma 22. *Let E be a set of messages and τ, τ' be messages. Then, $\tau \in d^c(E \setminus \{\tau\})$ implies $d(E)_{|\tau\to\tau'} \subseteq d(E_{|\tau\to\tau'} \cup \{\tau'\})$.*

In order to define messages that can be replaced by an intruder atom from $\mathcal{A}_I$, we need to know how variables are substituted in instances. Therefore, we now define substitutions that keep track of this information. More specifically, let

$$\rho = q_0, q_1, \ldots, q_l$$

be a rooted path in the concurrent game structure $S = S_{Pr}$ (induced by protocol Pr), i.e., $q_0 = q^0$ is the initial state of S and q_{i+1} is a successor of q_i as defined in Section 2.1. For $i \in \{0, \ldots, l\}$ let

$$q_i = (\mathcal{K}^i, \overline{P}^i, \overline{\mathbf{m}}^i, \overline{s}^i) \ .$$

For $a \in \mathcal{H}$ let

$$P^i_a = (V^i_a, E^i_a, r^i_a, \lambda^i_a, l^i_a) \ .$$

For $i \in \{0, \ldots, l-1\}$ and $a \in \mathcal{H}$ let v^i_a such that

$$(\mathbf{m}^i_a, P^i_a) \overset{v^i_a}{\mapsto} (\mathbf{m}, P^{i+1}_a) \ .$$

Let $l^i_a(r^i_a, v^i_a) = \mathbf{r}^i_a \Rightarrow \mathbf{s}^i_a$. Let $\tau^{\rho}_{i,a}$ be the substitution with domain $\mathcal{V}(\mathbf{r}^i_a)$ such that for all $c \in \mathrm{dom}(\mathbf{r}^i_a) \cap \mathrm{dom}(\mathbf{m}^i_a)$ we have

$$\mathbf{r}^i_a(c)\tau^{\rho}_{i,a} = \mathbf{m}^i_a(c) \ .$$

We inductively define σ^{ρ}_i by

$$\begin{aligned} \sigma^{\rho}_0 &= \emptyset \\ \sigma^{\rho}_{i+1} &= \sigma^{\rho}_i \cup \textstyle\bigcup_{a\in\mathcal{H}} \tau^{\rho}_{i,a} \qquad \text{for } i \in \{0, \ldots, l-1\} \ . \end{aligned}$$

We set $\sigma_\rho = \sigma^{\rho}_l$. For a substitution σ and terms t and t' we say that t is a *σ-match of t'* ($t \sqsubseteq_\sigma t'$) if t is not a variable and $t\sigma = t'$. Now we define for a message m what it means that m does not match with a rooted path in S or $G^{\varphi}_{(S,q^0)}$.

Definition 5. *Let $\rho = q_0, q_1, \ldots, q_l$ be a rooted path in S. A message m* does not match with *ρ if $t \not\sqsubseteq_{\sigma_\rho} m$ for all $t \in Sub(Pr) \cup \mathcal{A}_I$.*

Let $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_l$ be a rooted path in $G^{\varphi}_{(S,q^0)}$ where $\alpha_i = (q_i, \psi_i)$. Let $0 = i_0 < i_1 < \cdots < i_k \leq l$ such that

– $q_{i_k} = q_l$,
– $q_{i_s} \neq q_{i_{s+1}}$ for all $s \in \{0, \ldots, k-1\}$, and
– $q_{i_s} = q_{i_s+1} = \cdots = q_{i_{s+1}-1}$ for all $s \in \{0, \ldots, k-1\}$.

Then $\rho = q_{i_0}, q_{i_1}, \ldots, q_{i_k}$ is a rooted path in S and we call ρ the *S-projection of* α. A message m *does not match with* α if m does not match with the S-projection of α.

The following lemma states that a message that does not match with a rooted path ρ in S can be replaced by a new intruder atom from $\mathcal{A}_I$ and after this replacement one still has a rooted path in S with essentially the same properties. In particular, at the end of the rooted path the intruder can derive exactly the same constants as he could before the replacement.

Lemma 23. *Let*

$$\rho = q_0, q_1, \ldots, q_l$$

be a rooted path in S. Let τ be a message that does not match with ρ. Furthermore, let $a_I \in \mathcal{A}_\mathcal{I}$ be a constant that does not occur anywhere in ρ or Pr, and define

$$\rho' = q'_0, q'_1, \ldots, q'_l$$

where $q'_j = q_{j|_{\tau \to a_I}}$. Then, the following is true:

1) ρ' is a rooted path in S.
2) For each $j \in \{0, \ldots, l\}$ and $a \in \mathcal{H}$ we have $\Delta(q'_j, a) = \Delta(q_j, a)$.
3) For each $j \in \{0, \ldots, l\}$ we have that $\Delta(q_j, \mathcal{I})_{|\tau \to a_I} \subseteq \Delta(q'_j, \mathcal{I})$.
4) For each $j \in \{0, \ldots, l\}$, $a \in \mathcal{H}$, and $b \in \mathcal{P}$ we have that $\Delta(q'_j, \mathsf{sch}(a, b)) = \Delta(q_j, \mathsf{sch}(a, b))$.
5) We have that $\pi(q_l) = \pi(q'_l)$.

Proof. First, assume that τ does not occur as a subterm anywhere in ρ. Then $\rho' = \rho$ and nothing is to show. In what follows, we show the properties claimed for ρ' under the assumption that τ occurs in ρ. The proof is organized in three steps. First, we will prove that the intruder can derive message τ when it first occurs in ρ. Second, with this proved we show some auxiliary claims for the states q'_i. Third, with these auxiliary claims we show claims 1) to 5) from above.

Before starting with the first step described above we introduce some notation. For $i \in \{0, \ldots, l\}$ let

$$q_i = (\mathcal{K}^i, \overline{P}^i, \overline{\mathbf{m}}^i, \overline{s}^i).$$

For $a \in \mathcal{H}$ let

$$P^i_a = (V^i_a, E^i_a, r^i_a, \lambda^i_a, l^i_a) \ .$$

For $i \in \{0, \ldots, l-1\}$ and $a \in \mathcal{H}$ let v^i_a such that

$$(\mathbf{m}^i_a, P^i_a) \overset{v^i_a}{\mapsto} (\mathbf{m}, P^{i+1}_a) \ .$$

Let $l^i_a(r^i_a, v^i_a) = \mathbf{r}^i_a \Rightarrow \mathbf{s}^i_a$. We also introduce primed versions of these symbols, for example, $q'_i = (\mathcal{K}'^i, \overline{P}'^i, \overline{\mathbf{m}}'^i, \overline{s}'^i)$.

First step. Now we show that the intruder can derive τ, even with a composition rule at the end of a minimal derivation, when it first occurs in ρ. We call (*) the property of τ that $t \not\sqsubseteq_{\sigma_l} \tau$ for all $t \in \mathrm{Sub}(Pr) \cup \mathcal{A}_I$.

We know that there is $i \in \{0, \ldots, l\}$ such that τ occurs in q_i. Let $p \in \{0, \ldots, l\}$ be minimal such that τ occurs in q_p. We first show the following three claims:

i) $p > 0$,

ii) there is a channel c of the form $\mathsf{net}(e, a)$ or $\mathsf{dir}(d, a)$ where $a \in \mathcal{H}$, $d \in \mathcal{D}$, and $e \in \mathcal{P}$ such that $\tau \in \mathrm{Sub}(\mathbf{m}_a^p(c))$, and

iii) τ does not occur in components of q_p other then those described in ii).

Claim i): From (*) it follows that $\tau \notin \mathrm{Sub}(Pr) \cup \mathcal{A}_I$. Since initially there are no messages on secure channels and no messages on channels to honest instances τ does not occur in q_0.

Claim ii) and iii): We show that τ can not occur in components of q_p other than those described in ii). First assume that τ occurs in some scheduled secure channel, i.e., there is $c \in \mathcal{SC}$ such that s_c^p contains τ. Let m be a message in s_c^p that contains τ. Thus, m has to be sent in a step before step p by an honest instance, i.e., there is $s < p$ such that $m = \mathbf{s}_a^s \sigma_s$. From (*) it follows that there is a variable x in the domain of σ_s such that $\tau \in \mathrm{Sub}(\sigma_s(x))$. By definition of a-instances x occurs in step s or before in ρ. From this we get that τ occurs in a step before step p. This contradicts the minimality of p. Second, assume that τ occurs in $\mathcal{K}_p$. Since all messages in $\mathcal{K}_p$ are in $\mathcal{K}_0$ or are sent by honest principals to the intruder by a similar argument to the one used in the first case we get a contradiction to the minimality of p. Third, assume that τ occurs in an instance of q_p. By a similar argument to the one used in the first case we get a contradiction to the minimality of p. This concludes the proof of Claim ii) and iii).

Now we can show that (**) $\tau \in d^c(\mathcal{K}_p)$. From ii) and iii) it follows that $\tau \notin \mathrm{Sub}(\mathcal{K}_p)$ and $\tau \in \mathrm{Sub}(d(\mathcal{K}_p))$. By Lemma 20 we get that $\tau \in d^c(\mathcal{K}_p)$.

Second step. We now do the second step of the proof, i.e., we show auxiliary claims about q_i' needed to prove claims 1) to 5). More precisely, by induction on $0 \leq j \leq l$ using (**) from above we will show that the following claims hold:

a) q_j' is a state of S,

b) for each $a \in \mathcal{H}$ and $j < l$ we have $\Delta(q_j', a) = \Delta(q_j, a)$,

c) $d(\mathcal{K}_j)_{|\tau \to a_I} \subseteq d(\mathcal{K}_j')$,

d) for each $a \in \mathcal{H}$ and $b \in \mathcal{P}$, with $\mathsf{sch}(a, b) \in \mathsf{ch}(Pr)$, we have that $\Delta(q_j', \mathsf{sch}(a, b)) = \Delta(q_j, \mathsf{sch}(a, b))$, and

e) if $j < l$ then q_{j+1}' is a successor of q_j'.

First, assume $j = 0$. Claims a),b),d) are obviously fulfilled since $q_0' = q_0$. To show claim c) we distinguish between two cases:

– $\tau \in d(\mathcal{K}_0)$: By (*) we know that $\tau \notin \mathrm{Sub}(\mathcal{K}_0)$. By Lemma 20 we get that $\tau \in d^c(\mathcal{K}_0) (= d^c(\mathcal{K}_0 \setminus \{\tau\}))$. By Lemma 22 we get c).

– $\tau \notin d(\mathcal{K}_0)$: Since we have that $\tau \notin \mathrm{Sub}(\mathcal{K}_0)$, by Lemma 20, we know that for all $m \in d(\mathcal{K}_0)$ we have that $\tau \notin \mathrm{Sub}(m)$. Thus, we have $d(\mathcal{K}_0)_{|\tau \to a_I} = d(\mathcal{K}_0) = d(\mathcal{K}_{0|\tau \to a_I}) = d(\mathcal{K}'_0)$.

To show claim e) we distinguish between two cases:

– $p > 1$: We have $q'_1 = q_1$ and thus we have that q'_1 is a successor of $q'_0 = q_0$.
– $p = 1$: Choose the ports which carry messages that contain τ. By the points above we have that the intruder can derive these messages and τ does not occur in other components of q_1. Thus, we have that q'_1 is a successor of $q'_0 = q_0$.

For the induction step assume that a) to e) are true for a $j-1$. We want to proof the statements for $j > 0$. Claim a) is fulfilled by induction and claim e) for $j-1$.

To prove claim b) we have to show that $\Delta(q_j, a) = \Delta(q'_j, a)$. For this it suffices to show that if a message m matches with a term t occurring in the left-hand side of a rule in q_j for $a \in \mathcal{H}$, then $m_{|\tau \to a_I}$ matches with $t_{|\tau \to a_I}$ and vice versa. More precisely, let $v \in V_a^j$ be a successor of r_a^j and let $l_a^j(r_a^j, v) = \mathbf{r} \Rightarrow \mathbf{s}$. Let $t = \mathbf{r}(c)$ for some $c \in \mathrm{dom}(\mathbf{r})$.

First, suppose that $m = \mathbf{m}_a^j(c)$ matches with t, i.e., $m = t\sigma$ for some substitution σ. We have to show that $m_{|\tau \to a_I}$ matches with $t_{|\tau \to a_I}$, i.e., we have to show that there is a substitution σ' such that $m_{|\tau \to a_I} = t_{|\tau \to a_I}\sigma'$. Let $t^0 = l_a^0(r_a^j, v)$, i.e., t^0 is the term in $P_a^0 = P_a$ that corresponds to t. We have that

$$\begin{aligned}
m_{|\tau \to a_I} &= t\sigma_{|\tau \to a_I} \\
&= (t^0(\sigma_j \cup \sigma))_{|\tau \to a_I} \\
&= (t^0\sigma_l)_{|\tau \to a_I} \\
&\overset{\oplus}{=} t^0(\sigma_{l|\tau \to a_I}) \\
&= t^0(\sigma_j \cup \sigma)_{|\tau \to a_I} \\
&= t^0(\sigma_{j|\tau \to a_I} \cup \sigma_{|\tau \to a_I}) \\
&= (t^0(\sigma_{j|\tau \to a_I}))\sigma_{|\tau \to a_I} \\
&\overset{\oplus}{=} (t^0\sigma_j)_{|\tau \to a_I}\sigma_{|\tau \to a_I} \qquad = t_{|\tau \to a_I}\sigma_{|\tau \to a_I}
\end{aligned}$$

where all steps are obviously fulfilled by definition, except for the steps marked with $\oplus$: For these steps, we use property (*) from above. Thus, $m_{|\tau \to a_I}$ matches with $t_{|\tau \to a_I}$.

Now, conversely, suppose that $m_{|\tau \to a_I}$ matches with $t_{|\tau \to a_I}$, i.e., $m_{|\tau \to a_I} = t_{|\tau \to a_I}\sigma$ for some substitution σ. We have to show that m matches with t, i.e., $m = t\sigma'$ for some substitution σ'. Using the fact that a_I is a new intruder atom we have that

$$\begin{aligned}
m &= (m_{|\tau \to a_I})_{|a_I \to \tau} \\
&= (t_{|\tau \to a_I}\sigma)_{|a_I \to \tau} \\
&= (t_{|\tau \to a_I|a_I \to \tau})\sigma_{|a_I \to \tau} = t\sigma_{|a_I \to \tau} \ .
\end{aligned}$$

Thus, m matches with t.

To show claim c) we distinguish between two cases:

– $\tau \in d(\mathcal{K}_j)$: First, assume that $j \geq p$. By (**), we have that $\tau \in d^c(\mathcal{K}_j \setminus \{\tau\})$. By Lemma 22 we get the desired fact. Second, assume that $j < p$. We have that $\tau \notin \mathrm{Sub}(\mathcal{K}_j)$. Thus, by Lemma 20, we have that $\tau \in d^c(\mathcal{K}_j \setminus \{\tau\})$. By Lemma 22 we get the desired fact.
– $\tau \notin d(\mathcal{K}_j)$: In this case we know that $j < p-1$. Thus, we have that $\tau \notin \mathrm{Sub}(\mathcal{K}_j)$. From this, by Lemma 20, it follows that for every $m \in d(\mathcal{K}_j)$ we have $\tau \notin \mathrm{Sub}(m)$ and therefore

$$d(\mathcal{K}_j)_{|\tau \to a_I} = d(\mathcal{K}_j) = d(\mathcal{K}_{j\,|\tau \to a_I}) = d(\mathcal{K}'_j) \ .$$

To prove claim d) we have to show that $\Delta(q'_j, \mathsf{sch}(a,b)) = \Delta(q_j, \mathsf{sch}(a,b))$. By definition of q'_j we know that each scheduled secure channel $\mathsf{sch}(a,b)$ contains as many messages as in q_j. Thus, we have $\Delta(q'_j, \mathsf{sch}(a,b)) = \Delta(q_j, \mathsf{sch}(a,b))$.

To prove claim e) we have to show that if $j < l$, then q'_{j+1} is a successor of q'_j. If $j < l$ then we know that q_{j+1} is a successor of q_j since ρ is a path in S. Let $\gamma \in \Delta^{\Sigma}_{q_j}$ be a total move such that $\delta(q_j, \gamma) = q_{j+1}$. Let γ' be defined by $\gamma'(a) = \gamma(a)$ for $a \in \mathcal{H} \cup \mathcal{SC}$ and $\gamma'(\mathcal{I}) = \gamma(\mathcal{I})_{|\tau \to a_I}$. By claims b),c), and d) we get that $\gamma' \in \Delta^{\Sigma}_{q'_j}$. Now we have to show that $\delta(q'_j, \gamma') = q'_{j+1}$, i.e., we have to show that $\delta(q'_j, \gamma') = \delta(q_j, \gamma)_{|\tau \to a_I}$. With similar arguments as used for claim b) one shows that if $(\mathbf{m}^j_a, P^j_a) \overset{v}{\mapsto} (\mathbf{m}, P)$ for some v, $\mathbf{m}$, and P, then $(\mathbf{m}^j_{a\,|\tau \to a_I}, P^j_{a\,|\tau \to a_I}) \overset{v}{\mapsto} (\mathbf{m}_{|\tau \to a_I}, P_{|\tau \to a_I})$. With this it is easy to see that $\delta(q'_j, \gamma') = \delta(q_j, \gamma)_{|\tau \to a_I}$.

Third step. Now we are ready to prove claims 1) to 5) using claims a) to e) from above. Claim 1) is a direct consequence of points a) and e). Claims 2), 3), and 4) follow directly from b), c), and d), respectively. To show claim 5) we first show that for each atom c we have that $c \in d(\mathcal{K}_l)$ iff $c \in d(\mathcal{K}'_l)$. The implication from left to right follows from c) and the fact that $c \neq \tau$. The implication in the other direction, follows from Lemma 22 if we set E to be $\mathcal{K}_{q'_l}$, set τ (from Lemma 22) to be a_I, and τ' to be τ (from the lemma proved here). For all other propositional variables p it is obvious that $p \in \pi(q_l)$ iff $p \in \pi(q'_l)$. □

The following lemma states that in paths α of $G^{\varphi}_{(S,q^0)}$ starting in the initial vertex of $G^{\varphi}_{(S,q^0)}$ messages that do not match with α can be replaced by new constants and after this replacement one still has a path in $G^{\varphi}_{(S,q^0)}$.

Lemma 24. *Let*

$$\alpha = \alpha_0, \alpha_1, \ldots, \alpha_l$$

be a rooted path in $G^{\varphi}_{(S,q^0)}$. Let τ be a message that does not match with α. Furthermore, let $a_I \in \mathcal{A}_{\mathcal{I}}$ be a constant that does not occur anywhere in α and Pr, and define

$$\alpha' = \alpha'_0, \alpha'_1, \ldots, \alpha'_l$$

where $\alpha'_j = \alpha_{j\,|\tau \to a_I}$. Then we have that α' is a rooted path in $G^{\varphi}_{(S,q^0)}$.

Proof. First, assume that τ does not occur as a subterm anywhere in α. Then $\alpha' = \alpha$ and nothing is to show. Now, assume that τ occurs in α. For $i \in \{0, \ldots, l\}$ let $\alpha'_i = (q'_i, \psi'_i)$. Let $\rho = q_{i_0}, q_{i_1}, \ldots, q_{i_k}$ be the S-projection of α. By Lemma 23 we know that $\rho' = q'_{i_0}, q'_{i_1}, \ldots, q'_{i_k}$ is a rooted path in S. With statements 2),3), and 4) of Lemma 23 we can conclude that α' is a rooted path in $G^{\varphi}_{(S,q^0)}$. □

Let F be a finite subgraph of $G^{\varphi}_{(S,q^0)}$ such that the initial vertex α_0 of $G^{\varphi}_{(S,q^0)}$ is present in F and all vertices α in F are reachable from α_0 in F. A vertex $\alpha \in F$ is called *S-maximal* if there is no descendant α' of α in F such that an a-instance in α' differs from an a-instance in α, for some $a \in \mathcal{H}$.

We call a path in $G^{\varphi}_{(S,q^0)}$ *simple* if it is repetition free, i.e., all vertices in this path are pairwise distinct.

Definition 6. *Let F be a finite subgraph of $G^{\varphi}_{(S,q^0)}$ such that the initial vertex α_0 of $G^{\varphi}_{(S,q^0)}$ is present in F and all vertices α of F are reachable from α_0 in F. Let M be the set of S-maximal vertices in F. Let R be the set of all simple paths in F from α_0 to some vertex in M. Let R' be the set of all S-projections of paths in R. Let T be the set of all substitutions σ_ρ with $\rho \in R'$. A message m* does not match with *F if for all substitutions $\sigma \in T$ and all $t \in Sub(Pr) \cup \mathcal{A}_I$ we have that $m \not\sqsubseteq_\sigma t$.*

We now can extend Lemma 24 to subgraphs of $G^{\varphi}_{(S,q^0)}$.

Lemma 25. *Let F be a winning strategy graph for Player 0 in $G^{\varphi}_{(S,q^0)}$. Let τ be a message that does not match with F. Let $a_I \in \mathcal{A}_\mathcal{I}$ be a constant that does not occur anywhere in F and Pr. Then $F_{|\tau \to a_I}$ is a winning strategy graph for Player 0 in $G^{\varphi}_{(S,q^0)}$.*

Proof. Let $F' = F_{|\tau \to a_I}$. We have to show the following two points:

1) F' is a strategy graph for Player 0 in $G^{\varphi}_{(S,q^0)}$.

2) F' is winning for Player 0.

1) By claim 1) of Lemma 24 we get that F' is a subgraph of $G^{\varphi}_{(S,q^0)}$. Thus, it suffices to show that for all vertices α of Player 1 in F' all successors of $\alpha_{|\tau \to a_I}$ in $G^{\varphi}_{(S,q^0)}$ are present in F'.

Let $\alpha = (q_{|\tau \to a_I}, \psi_{|\tau \to a_I})$ be a vertex of Player 1 in F'. Then, by definition, (q, ψ) is a vertex of Player 1 in F. We distinguish between the different forms of formula ψ. First, if ψ is of the form

$$\psi_1 \wedge \psi_2,\ p,\ \neg p,\ X,\ \mu X.\psi \text{ or } \nu X.\psi,$$

then we have that the only successor of (q, ψ) in $G^{\varphi}_{(S,q^0)}$ is of the form (q, ψ') for some ψ'. Since F is a strategy graph for Player 0 we know that (q, ψ') is present in F. Thus, $(q_{|\tau \to a_I}, {\psi'}_{|\tau \to a_I})$ is present in F'. By definition of $G^{\varphi}_{(S,q^0)}$, we know that $(q_{|\tau \to a_I}, {\psi'}_{|\tau \to a_I})$ is the only successor of α in F'.

Second, if ψ is of the form $\Box_c \psi'$ or $[\![A]\!] \circ \psi'$, then we know that the choices of players that have to be specified are choices of honest participants of the

protocol Pr or scheduled secure channels, because φ is $\mathcal{I}$-positive. Since F is a strategy graph for Player 0 we know that each such choice, there is a unique successor (q', ψ') of (q, ψ) in F. Now, by condition 2) and 4) of Lemma 23, we can conclude that all successors of α are present in F'.

2) Obviously, it suffices to check that for all vertices $(q, \psi) \in F$ the evaluation of propositional variables in (q, ψ) and $(q_{|\tau \to a_I}, \psi_{|\tau \to a_I})$ is the same. This follows directly from point 5) of Lemma 23. □

Now we can prove Lemma 3. The idea of the proof is to repeatedly apply Lemma 25 to a given winning strategy graph F for Player 0 with an exponential number of vertices to obtain a winning strategy graph F' for Player 0 in which all messages occurring as a subterm in F' match with F'. By this fact and the exponential number of vertices in F' we obtain the exponential bound of the size of F' as desired.

Proof (Lemma 3). Let F be a winning strategy graph for Player 0 in the game $G^{\varphi}_{(S,q^0)}$ such that the number of vertices of F is exponentially bounded and for each vertex $\alpha \in F$ the length of any simple path from the initial vertex to α in F is polynomially bounded. Thus, the number of simple paths in F from the initial vertex in F to S-maximal vertices in F is exponentially bounded. By Lemma 25, we may assume that all messages occurring as subterms in F match with F. Since the number of substitutions as described in Definition 5 is exponentially bounded, it is easy to see that F can be represented in size exponentially bounded in $|Pr| + |\varphi|$ by representing the set of all messages occurring in F by a single DAG. □

8.7 Lower Bound

In this section, we prove that the problem PAMC(greedy, dssc-containing, $\mathcal{I}$-positive) is NEXPTIME-hard. The proof is by reduction from the the exponentially bounded tiling problem, a known NEXPTIME-hard problem.

The *exponentially bounded tiling problem* is defined as follows (see, e.g., [6]): Given is a finite set U of *tiles*, two relations $H, V \subseteq U \times U$, two tiles $u_0, u_f \in U$, and an integer (encoded in unary) $m > 0$. The question is whether it is possible to tile a $(2^m \times 2^m)$-square so that the horizontal neighbors belong to H, vertical neighbors belong to V, the left-top tile is u_0, and the left-bottom tile is u_f. More formally, the question is whether there exists a function $t : \{0, \ldots, 2^m - 1\}^2 \to U$ such that

(i) $\langle t(i,j), t(i+1,j)\rangle \in H$, for all $0 \leq i < 2^m - 1$, and $0 \leq j \leq 2^m - 1$,
(ii) $\langle t(i,j), t(i,j+1)\rangle \in V$, for all $0 \leq i \leq 2^m - 1$, and $0 \leq j < 2^m - 1$,
(iii) $t(0,0) = u_0$ and $t(0, 2^m - 1) = u_f$.

The function t is called a *solution* of the given tiling problem.

Given an instance $\mathcal{T}$ of this problem, i.e., given U, H, V, m, u_0, and u_f as above, we now (efficiently) construct a protocol Pr and an AMC-formula φ such that $(S_{Pr}, q^0) \models \varphi$ where q^0 is the initial state of φ iff $\mathcal{T}$ has a solution.

The formula (presented as an ATL-formula) is

$$\varphi = \langle\langle \mathcal{I} \rangle\rangle \Diamond p_c$$

for some propositional variable p_c. Note that φ is independent of $\mathcal{T}$, and hence, is fixed. Therefore and using Theorem 1, stating φ as an ATL-formula is w.l.o.g.

We now define Pr (which depends on $\mathcal{T}$). The constants used in Pr are c, e, h, v and the elements of U, where the constants e, h, v stand for "equal", "horizontal", and "vertical", and will be used as keys.

Potential solutions of tiling problems will be represented by messages that encode binary trees of depth $2 \cdot m$, using the pairing operator, with elements of U as their leafs. So, every path in such a tree has length $2 \cdot m$. The first m steps of such a path represent an integer i (encoded as bit string of length m) which stands for a column in the $(2^m \times 2^m)$-square. Analogously, the remaining m steps of the path represent an integer j which stands for a row in the $(2^m \times 2^m)$-square. The node the path is leading to represents the tile at position (i, j) in the square.

Following this intuition, we introduce the following notation: For a term s and a sequence $a \in \{0,1\}^*$, we recursively define $s[a]$ as follows: $s[\epsilon] = s$; $s[0a'] = s'[a']$, if $s = \langle s', s'' \rangle$, and otherwise $s[0a']$ is undefined; $s[1a'] = s''[a']$, if $s = \langle s', s'' \rangle$, and otherwise $s[1a']$ is undefined. Furthermore, for a term s and integers $i, j \in \{0, \ldots, 2^m - 1\}$, we write $s[i, j]$ for $s[ab]$, where $a \in \{0,1\}^m$ is the binary representation of i (with leading zeros, if necessary), $b \in \{0,1\}^m$ is the binary representation of j, and ab stands for the concatenation of the m-bit string a and b. Now, a function $t : \{0, \ldots, 2^m - 1\}^2 \to U$ (thus a potential solution of a tiling problem) can be represented by a term s such that, for each $0 \leq i, j < 2^m$, the expression $s[i, j]$ is defined and $s[i, j] = t(i, j)$. In that case, s is called *the term representation of* t. We call the term $s[i, j]$ (if defined) a *cell of* t.

The honest principals of Pr are $A_0, \ldots A_{2m+1}$. We also have one dishonest principal B, which we call the *the initiator*. (Recall that B will be played by the intruder.) The initial intruder knowledge is U.

The idea is that the initiator guess a solution of $\mathcal{T}$ (encoded by a message) and then the principals $A_0, \ldots A_{2m+1}$ are used to check whether the message is in fact a solution. More precisely, the message is sent by the initiator to A_0, converted in some way by A_0 and sent to A_1 over a direct secure channel, then converted again by A_1 and then sent to A_2 over a direct secure channel, and so on, until the message reaches A_{2m+1}, who will possibly output c to the initiator. The principals $A_0, \ldots, A_{2m+1}$ are defined in such a way that if the message given by the initiator to A_0 is in fact a solution, then no matter what the choices of the A_i are, at the end A_{2m+1} will output c. Otherwise, if the initiator did not send a solution, there will be at least one choice of the A_i such that A_{2m+1} does not output c.

We now describe the behavior of the honest principals in detail: While we do not formally define these principals in terms of trees, doing this is straightforward. We abbreviate messages of the form $\langle m_1, \langle m_2, \langle \cdots \langle m_{n-1}, m_n \rangle \cdots \rangle \rangle \rangle$ by $\langle m_1, \ldots, m_n \rangle$.

– A_0 waits to receive some message m_0 over a network channel from B, the initiator (and hence, the intruder). As a response, A_0 outputs $\{\langle m_0, m_0, m_0, m_0\rangle\}_e^s$ to A_1 over a direct secure channel.
Intuitively, m_0 represents a potential solution of $\mathcal{T}$. The purpose of $A_1, \ldots, A_m$ will then be to pick four bit strings $a^1 = a_1^1 \ldots a_m^1$, $a^2 = a_1^2 \ldots a_m^2$, $a^3 = a_1^3 \ldots a_m^3$, and $a^4 = a_1^4 \ldots a_m^4$, where $a_i^j \in \{0,1\}$ is picked by A_i for $j = 1, \ldots, 4$. Analogously, the purpose of $A_{m+1}, \ldots, A_{2m}$ will be to pick four bit strings $b^1 = b_1^1 \ldots b_m^1$, $b^2 = b_1^2 \ldots b_m^2$, $b^3 = b_1^3 \ldots b_m^3$, and $b^4 = b_1^4 \ldots b_m^4$, where $b_i^j \in \{0,1\}$ is picked by A_{i+m} for $j = 1, \ldots, 4$. Hence, $A_1, \ldots, A_{2m}$ pick for positions, namely $m_0[a^1b^1]$, $m_0[a^2b^2]$, $m_0[a^3b^3]$, and $m_0[a^4b^4]$ in the potential solution m_0. The principals are defined in such a way that $a^1 = b^1 = 0^m$, $a^2 = 0$, and $b^2 = 1^m$, i.e., the first two positions considered in m_0 are $(0,0)$ and $(0, 2^m - 1)$. Principal A_{2m+1} will check for these positions whether $m_0[0,0] = u_0$ and $m_0[0, 2^m - 1] = u_f$. Moreover, we either have that $a^4 = a^3 + 1$ (interpreted as integers) and $b^3 = b^4$, or that $a^3 = a^4$ and $b^4 = b^3 + 1$. In other words, the third and fourth position correspond to two positions in m_0 that are adjacent horizontally or vertically, respectively. Principal A_{2m+1} will use these positions to check whether the tilings at these positions are in a relationship in H or V, respectively.
– Principal A_i, $0 < i \leq m$, in response to the message from A_{i-1}, received over a direct secure channel, sends a message to A_{i+1} over a direct secure channel according to one of the following rules which all have the same priority, say 1, and are explained below:

$$\{\langle\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \langle x_3, y_3\rangle, \langle x_4, y_4\rangle\rangle\}_e^s \to \{\langle x_1, x_2, x_3, x_4\rangle\}_e^s,$$
$$\{\langle\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \langle x_3, y_3\rangle, \langle x_4, y_4\rangle\rangle\}_e^s \to \{\langle x_1, x_2, y_3, y_4\rangle\}_e^s,$$
$$\{\langle\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \langle x_3, y_3\rangle, \langle x_4, y_4\rangle\rangle\}_e^s \to \{\langle x_1, x_2, x_3, y_4\rangle\}_h^s,$$
$$\{\langle\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \langle x_3, y_3\rangle, \langle x_4, y_4\rangle\rangle\}_h^s \to \{\langle x_1, x_2, y_3, x_4\rangle\}_h^s$$

If A_i does not receive a message from A_{i-1} in the current round, then A_i stays in the same state by performing a self-loop which is defined to have priority 0.
As explained above, we want that $a_i^1 = a_i^2 = 0$. Therefore, for the first two messages ($\langle x_1, y_1\rangle$ and $\langle x_2, y_2\rangle$), all rules pick the left components, x_1 and x_2. As for the last two messages, the first two rules pick the same component. This corresponds to choosing $a_i^3 = a_i^4$. In the third rule, the first component is picked for the third message and the second component for the fourth message. This corresponds to choosing $a_i^3 = 0$ and $a_i^4 = 1$. Note that now the encryption key is h (instead of e). In particular, all A_j, with $i{+}1 \leq j \leq m$, can then only choose the last rule which corresponds to picking $a_j^3 = 1$ and $a_j^4 = 0$. Hence, $a^4 = a^3 + 1$.
– Principal A_i, $m < i \leq 2m$, in response to the message from A_{i-1}, received over a direct secure channel, sends a message to A_{i+1} over a direct secure channel according to one of the following rules which all have the same priority, say 1, and are explained below:

$$\{\langle\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \langle x_3, y_3\rangle, \langle x_4, y_4\rangle\rangle\}^s_e \to \{\langle x_1, x_2, x_3, x_4\rangle\}^s_e,$$
$$\{\langle\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \langle x_3, y_3\rangle, \langle x_4, y_4\rangle\rangle\}^s_e \to \{\langle x_1, x_2, y_3, y_4\rangle\}^s_e,$$
$$\{\langle\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \langle x_3, y_3\rangle, \langle x_4, y_4\rangle\rangle\}^s_h \to \{\langle x_1, x_2, x_3, x_4\rangle\}^s_h,$$
$$\{\langle\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \langle x_3, y_3\rangle, \langle x_4, y_4\rangle\rangle\}^s_h \to \{\langle x_1, x_2, y_3, y_4\rangle\}^s_h,$$
$$\{\langle\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \langle x_3, y_3\rangle, \langle x_4, y_4\rangle\rangle\}^s_e \to \{\langle x_1, x_2, x_3, y_4\rangle\}^s_v,$$
$$\{\langle\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \langle x_3, y_3\rangle, \langle x_4, y_4\rangle\rangle\}^s_v \to \{\langle x_1, x_2, y_3, x_4\rangle\}^s_v,$$

If A_i does not receive a message from A_{i-1} in the current round, then A_i stays in the same state by performing a self-loop which is defined to have priority 0.

The intuition behind the rules is similar to the previous case. Here, A_i chooses the bits $b_i^1, \ldots, b_i^4$. If the message received is encrypted by h, then this means that in the previous case two (horizontally) adjacent positions in m_0 were chosen already. So, b_i^3 has to be equal to b_i^4. Otherwise, if the message is encrypted by e, two vertically adjacent positions can be chosen. This is done analogously to the previous case.

– A_{2m+1} receives a message from A_{2m} over a direct secure channel and sends a message to the initiator (and thus, to the intruder) over a network channel according to one of the following rules:

$$\{\langle u_0, u_f, a, b\rangle\}^s_h \to c \qquad \text{for each } (a, b) \in H,$$
$$\{\langle u_0, u_f, a, b\rangle\}^s_v \to c \qquad \text{for each } (a, b) \in V,$$
$$\{\langle u_0, u_f, a, a\rangle\}^s_e \to c \qquad \text{for each } a \in U.$$

From the explanation given above it should now be clear that the intruder has a strategy to obtain c iff m_0 encodes a solution of $\mathcal{T}$, and hence, iff $\mathcal{T}$ has a solution: Clearly, if $\mathcal{T}$ has, then the intruder can send this solution (encoded as a message) to A_0 and in any case will receive c at the end. Conversely, if m_0 does not have the correct format, i.e., does not encode a binary tree as explained above, then one of the A_i will not be able to apply a rule, and hence, the intruder will not obtain c. If m_0 is a binary tree as required but it nevertheless does not represent a solution of $\mathcal{T}$, then one of the conditions (i) to (iii) will be violated and then there exists a choice of the $A_1, \ldots, A_{2m}$ such that A_{2m+1} will not be able to apply any of the rules available.

We finally note that instead of direct secure channels one could as well use only network channels. In this case, more keys would be used to enforce the intruder to forward messages from one principal to the next as desired.

9 Conclusion

In this paper, we studied the AMC-model checking problem over infinite-state concurrent games structures induced by protocols and the Dolev-Yao intruder.

We proved that to obtain decidability it is necessary to restrict to greedy and dssc-free protocols, which seems to be a reasonable class of protocols from a practical point of view. For this class of protocols and the $\mathcal{I}$-monotone fragment of AMC, which contains all game-theoretic properties formulated, for example, by Kremer and Raskin, we obtained decidability of the model checking problem with tight complexity bounds. The complexity upper bounds were obtained by combining techniques from the theory of infinite games and cryptographic protocol analysis in a novel and quite modular way, and hence, it is quite likely that results for reachability properties, e.g., taking algebraic properties into account, also carry over to our setting. The main technical question left open by our result is whether the model checking problem is decidable also for full AMC. To obtain practical implementations, it might be possible to employ constraint solving techniques similar to those for reachability properties [3, 19]. Given the succinctness of ATL* compared to AMC, it might also be useful to find implementations particularly tailored to (fragments) of ATL* or fair ATL.

References

1. R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS'97)*, pages 100–109. IEEE Computer Society Press, 1997.
2. R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. To appear in Journal of the ACM. Available from http://www.cis.upenn.edu/ alur/Jacm02.pdf, 2002.
3. A. Armando, D.A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P.H. Drielsma, P.-C. Héam, 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 K. Etessami and S.K. Rajamani, editors, *Computer Aided Verification, 17th International Conference (CAV 2005)*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer-Verlag, 2005.
4. N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99. IEEE Computer Society, 1998.
5. M. Boreale. Symbolic trace analysis of cryptographic protocols. In F. Orejas, P.G. Spirakis, and J. van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium (ICALP 2001)*, volume 2076 of *Lecture Notes in Computer Science*, pages 667–681. Springer-Verlag, 2001.
6. D. Calvanese, G .De Giacomo, M. Lenzerini, and M.Y. Vardi. View-based query containment. In *PODS 2003*, pages 56–67. ACM Press, 2003.
7. R. Chadha, M.I. Kanovich, and A.Scedrov. Inductive methods and contract-signing protocols. In P. Samarati, editor, *8-th ACM Conference on Computer and Communications Security (CCS 2001)*, pages 176–185. ACM Press, 2001.
8. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 261–270. IEEE, Computer Society Press, 2003.

9. R. Corin, S. Etalle, and A. Saptawijaya. A Logic for Constraint-based Security Protocol Analysis. In *IEEE Symposium on Security and Privacy (SP 2006)*, pages 155–168. IEEE Computer Society, 2006.
10. E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science (FOCS '91)*, pages 368–377. IEEE Computer Society Press, 1991.
11. E. Grädel, W. Thomas, and Th. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.
12. D. Kähler and R. Küsters. Constraint Solving for Contract-Signing Protocols. In M. Abadi and L. de Alfaro, editors, *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR 2005)*, volume 3653 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2005.
13. D. Kähler, R. Küsters, and Th. Wilke. Deciding Properties of Contract-Signing Protocols. Technical Report 0409, Institut für Informatik und Praktische Mathematik, CAU Kiel, Germany, 2004.
14. D. Kähler, R. Küsters, and Th. Wilke. Deciding Properties of Contract-Signing Protocols. In Volker Diekert and Bruno Durand, editors, *Proceedings of the 22nd Symposium on Theoretical Aspects of Computer Science (STACS 2005)*, volume 3404 of *Lecture Notes in Computer Science*, pages 158–169. Springer-Verlag, 2005.
15. D. Kähler, R. Küsters, and Th. Wilke. A Dolev-Yao-based Definition of Abuse-free Protocols. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, *Proceedings of the 33rd International Colloqium on Automata, Languages, and Programming (ICALP 2006)*, volume 4052 of *Lecture Notes in Computer Science*, pages 95–106. Springer, 2006.
16. S. Kremer and J.-F. Raskin. Game analysis of abuse-free contract signing. In *Computer Security Foundations Workshop 2002 (CSFW 2002)*, pages 206–220. IEEE Computer Society, 2002.
17. Steve Kremer and Jean-François Raskin. A game-based verification of non-repudiation and fair exchange protocols. In *12th International Conference on Concurrency Theory (CONCUR 2001)*, volume 2154 of *Lecture Notes in Computer Science*, pages 551–565. Springer-Verlag, 2001.
18. D.A. Martin. Borel Determinacy. *Annals of Mathematics*, 102:363–371, 1975.
19. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 166–175. ACM Press, 2001.
20. A.W. Mostowski. Games with forbidden positions. Technical Report 78, Uniwersytet Gdański, Instytut Matematyki, 1991.
21. M. Rusinowitch and M. Turuani. Protocol Insecurity with Finite Number of Sessions is NP-complete. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 174–190. IEEE Computer Society, 2001.
22. M. Rusinowitch and M. Turuani. Protocol insecurity with a finite number of sessions, composed keys is NP-complete. *Theoretical Computer Science*, 299(1–3):451–475, 2003.
23. S. Schewe and B. Finkbeiner. Satisfiability and Finite Model Property for the Alternating-Time mu-Calculus. In *CSL 2006*, volume 4207 of *Lecture Notes in Computer Science*, pages 591–605. Springer, 2006.
24. V. Shmatikov and J.C. Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science (TCS), special issue on Theoretical Foundations of Security Analysis and Design*, 283(2):419–450, 2002.

25. T. Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bull. Soc. Math. Belg.*, 8(2), May 2001.
26. Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.