EasyChair Preprint

№ 15785

Divide and Conquer: a Compositional Approach to Game-Theoretic Security

Ivana Bocevska, Anja Petković Komel, Laura Kovács, Sophie Rain and Michael Rawson

September 12, 2025

Divide and Conquer: A Compositional Approach to Game-Theoretic Security

IVANA BOCEVSKA, TU Wien, Austria
ANJA PETKOVIĆ KOMEL, Argot Collective, Switzerland
LAURA KOVÁCS, TU Wien, Austria
SOPHIE RAIN, Argot Collective, Switzerland
MICHAEL RAWSON, University of Southampton, United Kingdom

We propose a compositional approach to combine and scale automated reasoning in the static analysis of decentralized system security, such as blockchains. Our focus lies in the game-theoretic security analysis of such systems, allowing us to examine economic incentives behind user actions. In this context, it is particularly important to certify that deviating from the intended, *honest* behavior of the decentralized protocol is not beneficial: as long as users follow protocol, they cannot be financially harmed, regardless of how others behave. Such an economic analysis of blockchain protocols can be encoded as an automated reasoning problem in the first-order theory of real arithmetic, reducing game-theoretic reasoning to satisfiability modulo theories (SMT). However, analyzing an entire game-theoretic model (called game) as a single SMT instance does not scale to protocols with millions of interactions. We address this challenge and propose a *divide-and-conquer security analysis* based on compositional reasoning over games. Our compositional analysis is incremental: we divide games into subgames such that changes to one subgame do not necessitate re-analyzing the entire game, but only the ancestor nodes. Our approach is sound, complete and effective: combining the security properties of subgames yields security of the entire game. Experimental results show that compositional reasoning discovers intra-game properties and errors while scaling to games with millions of nodes, enabling security analysis of large protocols.

CCS Concepts: • **Theory of computation → Automated reasoning**; **Algorithmic game theory**; • **Security and privacy → Logic and verification**; **Distributed systems security**.

Additional Key Words and Phrases: Game Theory, Security, SMT Solving, Automated Reasoning

Reference Format:
Ivana Bocevska, Anja Petković Komel, Laura Kovács, Sophie Rain, and Michael Rawson. 2025. Divide and Conquer: A Compositional Approach to Game-Theoretic Security. In *EasyChair preprint, 2025*, 48 pages.

1 Introduction

Decentralized systems based on blockchain technology, such as cryptocurrencies [Nakamoto 2009] and decentralized finance [Buterin 2014], are in need of security guarantees. Establishing such guarantees is usually first approached by the formal static analysis of the underlying cryptographic protocols [Blanchet 2014; Kobeissi et al. 2020; Meier et al. 2013; Wang et al. 2020]. Then, game-theoretic security analysis [Rain et al. 2023; Zappalà et al. 2020] is employed to ensure economic incentives in a protocol align with intended functional outcomes. The latter ensures malicious actions and security attacks can be prevented via punishment mechanisms. This paper scales and

Authors' Contact Information: Ivana Bocevska, TU Wien, Vienna, Austria, ivana.bocevska@tuwien.ac.at; Anja Petković Komel, Argot Collective, Zug, Switzerland, anja@argot.org; Laura Kovács, TU Wien, Vienna, Austria, laura.kovacs@tuwien.ac.at; Sophie Rain, Argot Collective, Zug, Switzerland, sophie.rain@argot.org; Michael Rawson, University of Southampton, Southampton, United Kingdom, michael@rawsons.uk.

forsyte, Vienna, Austria

Publication date: August 2025.

improves game-theoretic security with satisfiability modulo theory (SMT) solving over blockchain security. We were motivated by the compositional static analysis in [Blackshear et al. 2018].

Recent work shows that automatic analysis of game-theoretic models (called games) is tractable via SMT solving in first-order real arithmetic. In particular, game-theoretic analysis is reduced to solving a single large SMT instance [Brugger et al. 2023]. However, this style of automated analysis has inherent limitations. A major problem is *scalability*: game models of large protocols are huge, yielding enormous SMT instances that cannot be solved in reasonable time. Another challenge is *game-theoretic modeling*: it is much more convenient to reason about protocol parts/phases in a modular, independent manner and compose their results into results over the entire protocol. Such convenience becomes even more pronounced in the presence of repeated phases.

This paper addresses the aforementioned challenges and introduces a *compositional approach to game-theoretic security* (Section 5). Given a protocol, we study its parts independently (as subgames), analyze the subgames' security, and combine their results to enforce security of the game that models the entire protocol. In other words, we perform a *divide-and-conquer* approach for game-theoretic security analysis, whose automation is feasible via SMT solving (Section 6). Game modeling is not in the scope of this paper, but interleaving the modeling and the analysis of a game, as shown in Section 7, makes it feasible to verify even complex real-world models with over 100 million nodes.

Compositional reasoning is, however, not trivial, as illustrated by Example 4.1, which shows that SMT queries may not be naïvely split into subqueries, as constraints in one may interact with constraints in another (Section 4). Further, a security result of a subgame cannot be simply propagated upwards, as in our experiments, we have encountered all four possible scenarios: a subgame is not secure, but the entire game is, and vice-versa; both subgame and entire game are secure; both are not secure. Our divide-and-conquer approach provides a theoretically *sound and complete* way to decompose reasoning into fine-grained SMT queries over subgames (Theorem 6.4).

To the best of our knowledge, our approach is the first compositional method for game-theoretic security. We implement our work as an extension of the CheckMate tool [Rain et al. 2024], resulting in our CheckMate2.0 framework. While our approach in this paper is illustrated via a simple example, our experiments demonstrate that divide-and-conquer reasoning in CheckMate2.0 enables game-theoretic analysis even for complex real-world protocols with millions of nodes. Compositional reasoning allows us, thus, to scale game-theoretic security analysis to large code bases, improving the efficiency of automated security analysis in general.

1.1 Setting and Wider Application

Game theory provides a rich formalism for reasoning about systems and their economics. Here we work with *extensive-form* games, which are essentially fixed finite trees where each branch has an associated player. Playing these games involves a traversal of the tree from root to leaf, with players deciding which branch they wish to take. At the end of a game, each player receives a *utility*: this could be a number or a symbolic expression like $2c + d$. *Game-theoretic security* considers a subset of possible traversals *honest* and aims to ensure that honest behavior is compatible with economic incentives present in a game. It has been shown that if a game enjoys a handful of properties, it amounts to game-theoretic security [Rain et al. 2023]. Automated reasoning tools for game-theoretic security ingest a game and attempt to (dis)prove its security.

While the main motivation of our work comes from blockchain verification, our framework can be used for arbitrary (extensive-form) game-theoretic models of any kind of turn-based interaction, such as computer security in a wider sense, correctness of concurrent and distributed systems, argumentation, negotiation, or even the design and analysis of board games. To showcase this claim we provide the following example.

Example 1.1. Suppose that a bank concurrently executes two procedures concerning the same customer. (1) A credit check requests the total amount held by the customer, who has both a savings and a current account. This requires two reads: (1a) the present value of the current account, (1b) the present value of the savings account. (2) The customer has requested regular automated transfers of money from the current account to a savings account. This requires two writes: (2a) debit the current account, (2b) credit the savings account. There are two hazards: the sequence 1a-2a-2b-1b overreports the customer's total amount (higher utility), while the sequence 2a-1a-1b-2b underreports it (lower utility). All other sequences report a correct total ("honest" (actual) utility). This can be modeled as a game, where analyzing game-theoretic security includes checking that no interleaving reports an incorrect total (utility). Here, without some kind of transaction control, this is violated and our framework can enumerate both possible serializations where this occurs.

1.2 Our contributions.

We bring the following contributions[1].

- We introduce a *compositional framework for game-theoretic security analysis* (Section 5). Our framework defines player-dependent notions of security properties, which in turn enables divide-and-conquer reasoning over games. We divide games into subgames while ensuring that the resulting reasoning is both sound and complete.
- We advocate divide-and-conquer algorithmic reasoning to automate compositional modeling and security analysis (Section 6). We interleave subgame and supergame (parent game) reasoning, by using the security result of a subgame within leaves of their respective supergames.
- Our compositional framework naturally supports the *generation of counterexamples* if security properties are violated. Moreover, we *revise game preconditions* in order to strengthen and enforce security. When security is established, we *extract a game strategy* as a proven security certificate (Sections 6.2 and 6.3).
- We implement compositional game reasoning in the CheckMate2.0 tool. Our experiments show that compositionality significantly improves runtime and supports efficient case-splitting over symbolic game utilities (Section 7), enabling verification of real-world protocols.

Outline. This paper is structured as follows. Section 2 introduces common game-theoretic concepts relevant to our work. Section 3 summarizes the notion of game-theoretic security, including security properties, quantification over utilities, and counterexamples, as largely defined by previous work [Brugger et al. 2023; Rain et al. 2023; Zappalà et al. 2020]. Section 4 briefly shows the complexity of our work before our contributions are presented in Sections 5 to 7 as listed above.

2 Preliminaries

We assume familiarity with standard first-order logic [Smullyan 1995] and real arithmetic in the context of SMT solving [Barrett and Tinelli 2018; Bjørner and Nachmanson 2024]. We next introduce common game-theoretic concepts relevant to our work.

A *game* is a finite object with finitely many *players*. Players choose from a finite set of *actions* until the game ends, whereupon they receive a *utility*. The focus is on perfect information *Extensive Form Games* (EFGs) [Osborne and Rubinstein 1994] in which the actions are chosen sequentially with full knowledge of all previous actions. Games may yield collective benefit or loss, i.e. they are not necessarily *zero-sum*.

Definition 2.1 (Extensive Form Game — EFG). An *extensive form game* $\Gamma = (N, G)$ is determined by a finite non-empty set of players N together with a finite tree $G = (V, E)$. A game path $h = (e_1, ..., e_n)$,

[1]Formal proofs of all our claims can be found in Appendices A and C.

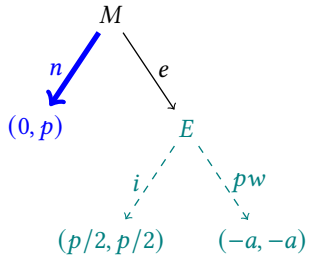

Fig. 1. Market Entry Game Γ_{me}, with $a, p > 0$. Utility tuples state M's utility first, E's second.

with $e_i \in E$, that starts from the root of G is called a *history*. We denote the set of histories $\mathscr{H}$. There is a bijection between nodes $v \in V$ and histories $h \in \mathscr{H}$ that lead to these nodes.

- A history that leads to a leaf is called *terminal* and belongs to the set of *terminal histories* $\mathscr{T} \subseteq \mathscr{H}$. Terminal histories t are associated with a *utility* for each player.
- *Non-terminal histories* are those histories that are not terminal. Non-terminal histories h have assigned a next player denoted as $P(h) \in N$. Player $P(h)$ chooses from the set $A(h)$ of possible actions following h.

Definition 2.2 (Honest History). In an EFG Γ, we call a terminal history *honest*, denoted by h^*, if it represents expected behavior in Γ. An EFG Γ can have many honest histories; security analysis over Γ is always performed relative to a chosen and fixed honest history (see Section 3.1).

Example 2.3 (Market Entry Game). Consider the Market Entry game Γ_{me} of Figure 1. This game has been chosen for its simplicity to ease readability; our approach can, however, be applied to real-world examples, as shown in Section 7. The Market Entry game Γ_{me} has two players: M representing a new company and E an established company. At the root, it is the turn of player $P(\emptyset) = M$ to choose from actions $A(\emptyset) = \{n, e\}$. Action n represents *not* entering the market, producing a terminal history (n) where M gets 0 utility and E gets all of the profits $p > 0$. Action e represents entering the market, in which case E can respond by either ignoring this move and thus splitting profits equally, or by entering a price war that damages both players.

Utilities in game theory are usually numeric constants. We generalize utilities to *symbolic* terms in real arithmetic and thus encode all possible values within given constraints. Variables and numeric constants are evaluated over the real numbers extended by a finite set of *infinitesimals*, closer to zero than any real number. Infinitesimals model subjective (in)conveniences that do not relate directly to funds, such as opportunity cost. We model infinitesimals with terms over $\mathbb{R} \times \mathbb{R}$, ordered lexicographically: the first component represents the real part, the second the infinitesimal. We write real for the first projection and avoid writing pairs, using $a, b, c \ldots$ for real variables, and $\alpha, \beta, \gamma, \ldots$ for infinitesimals. The utility term $a + \alpha - \varepsilon$ is therefore represented as $(a, 0) + (0, \alpha) - (0, \varepsilon) = (a, \alpha - \varepsilon)$.

Example 2.4. We could modify the Market Entry game from Figure 1 by adding an infinitesimal $\alpha > 0$ to the utility of player M at (e, i). Player M's new utility $\frac{p}{2} + \alpha$ at leaf (e, i), then represents half of the profit p and the additional benefit of entering the market α, as M is motivated to establish a new entity.

To formulate game-theoretic security properties, we need the following definitions for EFGs.

Definition 2.5 (EFG Properties). Let $\Gamma = (N, G)$ be an EFG.

Strategy A *strategy* σ for a group of players $S \subseteq N$ is a function mapping non-terminal histories $h \in \mathscr{H} \setminus \mathscr{T}$, where one of the players in group S has a turn $P(h) \in S$, to the possible actions $A(h)$. We write $\mathcal{S}_S$ for the set of strategies for group S, and $\mathcal{S}$ for $\mathcal{S}_N$ which we call *joint strategies*. We refer to the union of strategies with disjoint domains as a *combined strategy* and denote it as a tuple. To combine e.g. $\sigma_S \in \mathcal{S}_S$ and $\tau_{N-S} \in \mathcal{S}_{N-S}$, we write $(\sigma_S, \tau_{N-S}) \in \mathcal{S}$.

Resulting History The *resulting terminal history* $H(\sigma)$ of a joint strategy $\sigma \in \mathcal{S}$ is the unique history obtained by following chosen actions in σ from root to leaf.

Following Honest History A strategy for a player p *follows the honest history* h^* if, at every node along the honest history, where p is making a choice, the strategy chooses the action in h^*. For every other node, there is no constraint.

Utility Function The *utility function* $u_p(\sigma)$ assigns to player $p \in N$ their utility at the resulting terminal history of the joint strategy $\sigma \in \mathcal{S}$, that is $u_p(\sigma) := u_p(H(\sigma))$. We sometimes write all player utilities for a joint strategy $\sigma \in \mathcal{S}$ as $u(\sigma)$, denoting a tuple of size $|N|$.

Subgame *Subgames* $\Gamma_{|h}$ of Γ are formed from the same set N of players and a subtree of G, and are therefore identified by the history h leading to the subtree $G_{|h}$. A history $t \in \mathscr{H}_{|h}$ of $\Gamma_{|h}$ is a suffix of a history $(h, t) \in \mathscr{H}$, a strategy $\sigma \in \mathcal{S}_{|h}$ of $\Gamma_{|h}$ is a restriction of a strategy $\tau \in \mathcal{S}$ to the nodes in $G_{|h}$, that is $\tau_{|h} = \sigma$, and the utility function $u_{|h}$ of $\Gamma_{|h}$ assigns each joint strategy $\sigma \in \mathcal{S}_{|h}$ the utility of the yielded leaf $u_{|h}(\sigma) := u(h, H(\sigma))$. This includes trivial subgames: leaves or the entire tree Γ at the empty history.

Supergame If h' is a prefix of h, $\Gamma_{|h'}$ is a supergame of $\Gamma_{|h}$.

Subgame along/off Honest History Let h^* be the honest history. A subgame $\Gamma_{|h}$ is *along* the honest history iff h is a prefix of h^*; that is, there is a history $g \in \mathscr{H}_{|h}$ in the subtree such that $(h, g) = h^*$. Otherwise, $\Gamma_{|h}$ is *off* the honest history.

Intuitively, a *subgame* is the part of the game that is still to be played after some actions have been taken already. A *supergame* of a subgame is any game tree that embeds the subgame as the subtree. For the sake of readability, we use the following simplifications. From now on we use *subgame*/*subtree* and *supergame*/*supertree* interchangeably. We write $u(\sigma_S, \tau_{N-S}) := u((\sigma_S, \tau_{N-S}))$ for the combined strategy $(\sigma_S, \tau_{N-S}) \in \mathcal{S}$. For history $k \in \mathscr{H}$, we write $k_{|h}$ to express the suffix of k after h, that is $(h, k_{|h}) = k$.

Example 2.6. Consider again the Market Entry game Γ_{me} in Figure 1. A joint strategy $\tau \in \mathcal{S}$ could have M taking action n initially, and player E taking i after (e). M's (single) strategy $\tau_M \in \mathcal{S}_M$ takes action n at the root. Joint strategy τ yields utility $u_E(\tau) = p$ for player E. The history resulting from τ is (n).

The subgame $\Gamma_{me|(e)}$ after history (e) is indicated by dashed lines in Figure 1. It has players $\{M, E\}$ and a tree where E must choose between action i with utility $(\frac{p}{2}, \frac{p}{2})$ and action pw with utility $(-a, -a)$. Considering honest history (n) the subtree $\Gamma_{me|(e)}$ is *off* the honest history, whereas the trivial subtree $\Gamma_{me|(n)}$ after action n is *along* the honest history.

The Market Entry game has $2 \times 2 = 4$ joint strategies as M chooses from two possible actions, and independently also E picks one action out of two.

3 Game-Theoretic Security Properties

Our work models real-life protocols as extensive form games (EFGs). Subsequently, we reduce the security analysis of a protocol to the game-theoretic security analysis of its corresponding EFG. According to [Zappalà et al. 2020] an adversary could execute an attack in a protocol for personal gain or harming somebody. Therefore, we consider a protocol to be *game-theoretically secure* if the following properties hold:

(P1) **Byzantine Fault-Tolerance.** Even in the presence of adversaries, honest players do not suffer loss. That is, in a secure protocol an honest player will not receive negative utility, independent of others' behavior. Therefore, there are no "attacks" where somebody is harmed.
(P2) **Incentive Compatibility.** Rational agents do not deviate from the honest behavior, as it yields the best payoff. Hence, in a secure protocol, a rational "attacker" is behaving honestly and no adversary gets personal gain by deviation.

3.1 Security Properties for Subgames

As elaborated in [Rain et al. 2023], the high-level properties (P1) and (P2) can be ensured through the game-theoretic concepts *weak(er) immunity*, *collusion resilience*, and *practicality*. Property (P1) is ensured by weak(er) immunity and (P2) by collusion resilience and practicality. We take the definitions of these security properties as posed in [Brugger et al. 2023], and adapt them for any subtree of Γ, to accommodate a compositional game-theoretic approach (Section 5). Note that the definitions coincide when the entire game Γ is taken as the subtree.

Since we allow symbolic utilities, we do not necessarily know their ordering, respectively relation to zero. This knowledge is however needed to evaluate the security properties. This is why, in this subsection, we assume a total order on the symbolic utility terms. We will lift this assumption in Section 3.2.

Definition 3.1 (Weak Immunity). A subtree $\Gamma_{|h}$ of game Γ with honest history h^* is *weak immune*, if a strategy $\sigma \in \mathcal{S}_{|h}$ exists such that all players p following σ always receive non-negative utility:

$$\exists \sigma \in \mathcal{S}_{|h}. \forall p \in N\ \forall \tau \in \mathcal{S}_{|h}.\ u_p(\sigma_p, \tau_{N-p}) \geq 0\ . \qquad (\mathrm{wi}(\Gamma_{|h}))$$

If h is along h^*, additionally $H_{|h}(\sigma) = h^*_{|h}$ has to hold.

Example 3.2. The Market Entry game from Figure 1 with honest history (n) is weak immune: according to Definition 3.1 we look at the trivial subtree Γ_{me}, after the empty history $h = \emptyset$. Since the empty history is always along the honest history, following the last sentence of Definition 3.1, the desired strategy σ has to yield the honest history (n). If M behaves honestly both players get a nonnegative utility; if M deviates via e, player E can choose action i and obtains a positive utility $\frac{p}{2}$. Hence, the witness strategy σ assigns action n to the empty history: $\sigma(\emptyset) = n$, and action i to history (n): $\sigma((n)) = i$.

Sometimes, weak immunity is too restrictive and we take *weaker immunity* to ensure (P1).

Definition 3.3 (Weaker Immunity). A subtree $\Gamma_{|h}$ of game Γ with honest history h^* is *weaker immune*, if there exists a strategy $\sigma \in \mathcal{S}_{|h}$, such that all players p that follow σ always receive at least a negative infinitesimal:

$$\exists \sigma \in \mathcal{S}_{|h}. \forall p \in N\ \forall \tau \in \mathcal{S}_{|h}.\ \mathrm{real}(u_p(\sigma_p, \tau_{N-p})) \geq 0\ . \qquad (\mathrm{weri}(\Gamma_{|h}))$$

If h is along h^*, additionally $H_{|h}(\sigma) = h^*_{|h}$.

Next, the property of collusion resilience requires the honest behavior to yield the best payoff, even in the presence of collusion.

Definition 3.4 (Collusion Resilience). A subtree $\Gamma_{|h}$ of the game Γ with honest history h^* is *collusion resilient* if there exists a strategy $\sigma \in \mathcal{S}_{|h}$ such that no strict subgroup of players can deviate to receive a joint utility greater than their joint honest utility:

$$\exists \sigma \in \mathcal{S}_{|h}. \forall S \subset N\ \forall \tau \in \mathcal{S}_{|h}.\ \sum_{p \in S} u_p(h^*) \geq \sum_{p \in S} u_p(\sigma_{N-S}, \tau_S)\ . \qquad (\mathrm{cr}(\Gamma_{|h}))$$

If h is along h^*, also $H_{|h}(\sigma) = h^*_{|h}$ has to hold.

Note that the collusion resilience of a subtree according to the above definition depends on the *honest utility*, the utility resulting from the honest history in the entire game Γ. The node containing the honest utility is not necessarily part of the considered subtree. We also note that we only check for strict subgroups $S \subset N$, since if the entire group of players is colluding, no player gets harmed[2].

Example 3.5. Consider again the Market Entry game from Figure 1 with the honest history (n). This is collusion resilient: we can take actions n for player M and pw for player E. Since it is a two-player game, the colluding group of players can only be a singleton. If M deviates from the honest behavior, they get utility $-a$, which is less than 0 in the honest case. If E deviates, the history is not affected, since M chooses action n. Thus, the game is collusion resilient.

Note that this is not what one would usually do; it is just a toy example to show what collusion resilience can protect from: It guarantees that the honest participants have a way to keep others from maliciously increasing their profit. The rationality of the honest choices is ensured through the next property *practicality*.

The next property of practicality ensures that, for all player decisions, the honest behavior is also "greedy": if all players act selfishly, that is they maximize their own utilities, the honest choice yields the best utility.

Definition 3.6 (Practicality). A subtree $\Gamma_{|h}$ of the game Γ with honest history h^* is *practical*, if there exists a strategy $\sigma \in \mathcal{S}_{|h}$ such that no player can deviate in any subtree to receive a strictly greater utility in the subtree:

$$\exists \sigma \in \mathcal{S}_{|h}\ \forall g \in \mathscr{H}_{|h}\ \forall p \in N\ \forall \tau \in \mathcal{S}_{|(h,g)}.\ \ u_{|g,p}(\sigma_{|g}) \geq u_{|g,p}(\tau_p, \sigma_{|g,N-p})\ . \qquad (\mathrm{pr}(\Gamma_{|h}))$$

If h is along h^*, also $H_{|h}(\sigma) = h^*_{|h}$ has to hold.

Example 3.7. The Market Entry game from Figure 1 with the honest history (n) is not practical. Player E should choose i, as it yields a better utility. It is then not practical for M to choose n, as it yields utility 0, whereas action e yields the better utility $\frac{p}{2}$.

We finally note that every subtree $\Gamma_{|h}$ of a game Γ that is off the honest history is always practical.[3]

Example 3.8. Consider the Market Entry subgame after the non-terminal history (e), marked by teal dashed lines in Figure 1. We can always choose the action that yields the best utility for the current player E. The only way we can violate practicality is by having the best choice conflicting with the honest choice, which cannot happen when the subtree is off honest history.

The definitions weak(er) immunity, collusion resilience, and practicality can be extended to strategies and terminal histories in the following way.

Definition 3.9 (Security Properties of Strategies and Histories). A *strategy* $\sigma \in \mathcal{S}$ of a game Γ is weak(er) immune, collusion resilient, or practical, iff it can serve as the witness strategy in $(\mathrm{wi}(\Gamma_{|h}))$, $(\mathrm{weri}(\Gamma_{|h}))$, $(\mathrm{cr}(\Gamma_{|h}))$, or $(\mathrm{pr}(\Gamma_{|h}))$, respectively.

A *terminal history* $t \in \mathcal{T}$ of the game Γ is weak(er) immune, collusion resilient, or practical, iff there exists a strategy $\sigma \in \mathcal{S}$ that has the respective property and the resulting history of σ is t, that is $H(\sigma) = t$.

Together, these properties define game-theoretic security:

[2]If the entire group of players N is colluding and getting a better utility, this could happen for several reasons. There could be lower subjective values, like less opportunity costs, smaller waiting times, etc.; or the players could harm the system, for example by conjuring coins out of thin air. Since our work is checking harm to other players, this kind of a check is currently out of scope, but could be a potentially useful extension.

[3]This is also formally proven in Appendix C.

Definition 3.10 (Game-Theoretic Security). A game Γ with honest history h^* is *game-theoretically secure* if it is weak immune, collusion resilient, and practical.

In some applications, minor inconveniences can be neglected, and the notion of game-theoretic security can be weakened to *weaker immunity*, collusion resilience, and practicality.

Remark. Definition 3.10 implies that the listed security properties are not just any properties. They are *the* security properties that are sufficient to ensure game-theoretic security of game models as introduced and justified in [Rain et al. 2023].

In a protocol there can be several intended behaviors, corresponding to different honest histories. Those can be iteratively checked for security. Further, for an honest history we may get different strategies for different security properties. If a player deviated from the honest choice, the other players can choose among those strategies, depending on which attack they are defending against.

3.2 Total Orders

Similarly to [Brugger et al. 2023], in order to lift the assumption that we know how all utility terms relate, we make the security analysis relative to a finite set C of *initial constraints* on the symbolic variables appearing in the utility terms and explicitly universally quantify over the variables, as follows

$$\forall \vec{x}. \left(\bigwedge_{c \in C} c[\vec{x}] \right) \rightarrow \exists \sigma \in \mathcal{S}. H(\sigma) = h^* \wedge sp(\sigma)[\vec{x}] \,, \tag{1}$$

where $\vec{x} = (x_1, \ldots, x_\ell)$ are the real/infinitesimal variables occurring in the utility terms T_u and $sp(\sigma)$ is the formula of a security property $sp \in \{wi, weri, cr, pr\}$ after existential quantification of the strategy: E.g. for weak immunity $wi(\sigma) = \forall p \in N \; \forall \tau \in \mathcal{S}_{|h}. \; u_p(\sigma_p, \tau_{N-p}) \geq 0$, and similarly for the other properties.

Furthermore, to efficiently handle the comparison of symbolic utilities in an SMT solver, we implement an equivalent version of the above formula by considering all consistent *total orders* $\preceq$ over the set T_u of utility terms appearing in the game Γ.

Theorem 3.11 (Game-Theoretic Security with Total Orders). *For an EFG Γ with honest history h^* and a finite set of initial constraints C, property (1) is equivalent to*

$$\forall (\preceq, T_u) \; \exists \sigma \in \mathcal{S}. H(\sigma) = h^* \wedge \quad \forall \vec{x}. \left(\bigwedge_{c \in C \cup \preceq} c[\vec{x}] \right) \rightarrow sp(\sigma)[\vec{x}] \,. \tag{2}$$

3.3 Counterexamples

If a game violates a security property – that means if there is no joint strategy satisfying the security property – we can investigate why: Counterexamples serve the important purpose of providing attack vectors and thus pinpointing weaknesses of a protocol underlying the game model.

Counterexamples to Weak(er) Immunity. For the weak(er) immunity property, a counterexample is a harmed honest player p and a partial strategy of the other players $N - p$ such that no matter what honest actions player p chooses, p receives negative utility.

Definition 3.12 (Counterexamples to Weak(er) Immunity). Let Γ be an EFG and h^* the considered honest history. A *counterexample to h^* being weak(er) immune* is a player p together with a partial strategy s_{N-p} of the other players $N - p$ such that s_{N-p} combined with any strategy σ_p of player p that follows the honest history h^*, yields a terminal history $H(s_{N-p}, \sigma_p) = t_{\sigma_p}$ with $u_p(t_{\sigma_p}) < 0$ (resp. for weaker immunity $\text{real}(u_p(t_{\sigma_p})) < 0$) and s_{N-p} is *minimal* with that property.

Minimality of the partial strategy s_{N-p} states that, if any information point $s_{N-p}(h) = a$ is removed, there exists a strategy σ_p of player p such that (σ_p, s'_{N-p}) does not yield a terminal history, where s'_{N-p} is s_{N-p} without assigning action a to history h. That is, when following only actions of (σ_p, s'_{N-p}), we get stuck at an internal node of the tree.

Example 3.13. A counterexample to the weak immunity of the Market Entry game of Figure 1 with the (for this example considered) *honest* history (e, i) would be player M and a partial strategy for E, where they choose action pw. If M behaves honestly and chooses action e, they end up with the negative utility of $-a$ after the terminal history (e, pw).

Counterexample to Collusion Resilience. A counterexample to collusion resilience consists of a group of deviating players S and their partial strategy $s_S \in \mathcal{S}$, such that the joint utility of S is better than their joint honest utility, no matter how the other players $N - S$ react, while still following the honest history.

Definition 3.14 (Counterexamples to Collusion Resilience). Let Γ be an EFG and h^* the considered honest history. A *counterexample to h^* being collusion resilient* is a set of deviating players S together with a partial strategy s_S of players S such that s_S extended by any strategy σ_{N-S} of players $N - S$, which follows the honest history h^*, yields a terminal history $H(\sigma_{N-S}, s_S) = t_{\sigma_{N-S}}$ with

$$\sum_{p \in S} u_p(t_{\sigma_{N-S}}) > \sum_{p \in S} u_p(h^*)$$

and s_S is minimal with that property. The minimality of s_S is similar to the minimality of the partial strategy for weak(er) immunity.

Example 3.15. In the Market Entry game of Figure 1, a counterexample to the (for this example considered) *honest* history (e, pw) being collusion resilient is the deviating group $\{E\}$ with the partial strategy that takes action i after history (e). Since the honest player M can only take action e (while being honest), the deviating utility for E is $\frac{p}{2}$, which is greater than the honest one $-a$.

Counterexamples to Practicality. Intuitively, a counterexample to practicality of the honest history h^* has to provide a reason why a rational player would not follow h^*. At some point along h^* after a prefix h, there is an action a promising the current player $P(h)$ a strictly better utility than h^*. That means in the subgame $\Gamma_{|(h,a)}$ after (h, a) all practical terminal histories yield a utility for player $P(h)$ that is better than their honest one. Otherwise, other rational players would choose actions in $\Gamma_{|(h,a)}$ which disincentivize $P(h)$ to deviate from h^*.

Definition 3.16 (Counterexamples to Practicality). For an EFG Γ and honest history h^*, a *counterexample to practicality of h^** is a prefix h of h^* together with an action $a \in A(h)$, such that for all practical terminal histories t in the subgame $\Gamma_{|(h,a)}$ it holds that $u_{P(h)}(h^*) < u_{P(h)}((h, a, t))$.

Example 3.17. Recall that the Market Entry game Γ_{me} from Figure 1 with the honest history (n) is not practical. A counterexample to practicality is the empty prefix $h = \emptyset$ and the action e, as the only practical terminal history in $\Gamma_{me|(e)}$ is (i) which yields $\frac{p}{2}$ for player M, which is strictly better than the 0 in the honest case.

4 Unsound Naïve Approach to Compositionality

For a divide-and-conquer style of compositional game-theoretic security analysis, we would like to analyze a game tree by propagating security results of subtrees upwards to the parent/ancestor nodes of the supertree. However, naïvely propagating the yes/no security result of the subtree does not suffice, as shown in Example 4.1.

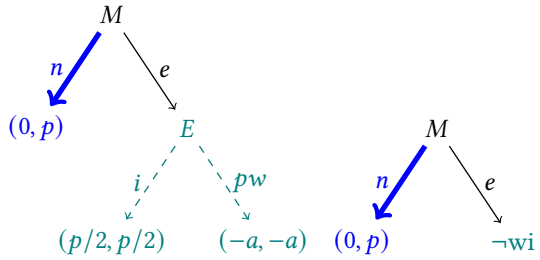

Fig. 2. Naive Compositionality of Weak Immunity for Market Entry Game Γ_{me}, $a, p > 0$.

Example 4.1. Consider the Market Entry game Γ_{me}(Example 2.3) reproduced on the left-hand side of Figure 2, with honest history (n). Example 3.2 shows this game is weak immune. Now consider a naïve compositional approach looking at the subgame after non-terminal history (e), marked by teal dashed lines. Since player E can take action pw — leading to negative utility for M — this subtree is not weak immune. To mimic a naïve compositionality approach, we replace the subtree after (e) by ¬wi, shown on the right. Asked whether this *supertree* is weak immune, one would say no, as M could deviate from the honest history via e, which leads to a subtree that is not weak immune. This is, however, an incorrect conclusion since the Market Entry game is weak immune for the honest history (n), as shown in Example 3.2.

The main reason why the naïve approach above fails is that we need more information to be able to propagate a result from a subtree to its parent, namely that the subtree is not weak immune *only for player M*. In the parent, player M can achieve weak immunity by behaving honestly and choosing action n, ensuring weak immunity of the entire game tree. Similar additional information (see Theorem 5.7) is needed for the other security properties: collusion resilience requires which colluding groups the subtree is secure against; practicality requires the utilities resulting from terminal histories that are practical in the subtree (*practical utilities*). We now show that propagating this information yields a sound and complete compositional approach to game-theoretic security.

5 Compositional Game-Theoretic Security

Our compositional framework for game-theoretic security analysis is materialized via two crucial components:

(1) Stratified analysis of security properties over players, capturing player-wise security properties (Section 5.1).
(2) Splitting player-wise security properties into subgames, enabling us to propagate subgame reasoning for deriving supergame security (Section 5.2).

For simplicity, we assume a total order $\preceq$ on the occurring utility terms T_u in order to relate symbolic game utilities. As before, this assumption is relaxed in Section 6, generalizing our approach.

5.1 Security Properties Stratified over Players

We start with the following observation. While Example 4.1 shows that there are no implications of subtree and supertree results in general, subtrees *along the honest history* can in fact soundly pass negative (not secure) results up to their parents.

Theorem 5.1 (Equivalence of Non-Secure Games). *A game Γ with honest history h^* violates one of the security properties of weak(er) immunity, collusion resilience, or practicality iff there exists a history h along the honest history h^* such that $\Gamma_{|h}$ violates the respective security property.*

Intuitively, the honest history h^* "enforces" a path down the tree Γ: when a non-secure subtree $\Gamma_{|h}$ is encountered along this path, there is no way to compensate for it. Theorem 5.1, however, only propagates non-secure properties along the honest history. To allow for all analysis results to propagate from subgames to supergames, we *stratify game-theoretic security analysis over individual players*. This means we can analyze the security properties for a player (weak immunity: Definition 5.2, practicality: Definition 5.5), or player group (collusion resilience: Definition 5.4) at a time, without interfering with results of other players or groups (Theorem 5.6).

Definition 5.2 (Weak Immunity for a Player). A subgame $\Gamma_{|h}$ with honest history h^* is *weak immune for player* $p \in N$, if there exists a strategy $\sigma \in \mathcal{S}_{|h}$ such that no matter to which strategy $\tau \in \mathcal{S}_{|h}$ other players deviate, p's utility will be non-negative and, if h is along h^*, then also $H_{|h}(\sigma) = h^*_{|h}$:

$$\exists \sigma \in \mathcal{S}_{|h}.\ (h \text{ along } h^* \rightarrow H_{|h}(\sigma) = h^*_{|h}) \wedge \qquad (\mathrm{wi}_p(\Gamma_{|h}))$$
$$\forall \tau \in \mathcal{S}_{|h}.\ u_{|h,p}(\sigma_p, \tau_{N-p}) \geq 0\ .$$

An analogous definition applies to *weaker immunity*.

Example 5.3 (Player-Wise Weak Immunity). Let us revisit the Market Entry game Γ_{me} with honest history (n) from Figure 1, considering one player at a time. The first player is M. The subgame $\Gamma_{me|(e)}$ after history (e) is not weak immune for M, since E could take action pw. Towards compositional reasoning, we try propagating this result to the supertree Γ_{me}. It leads to correctly reporting weak immunity for M: as M will honestly take action n, we avoid $\Gamma_{me|(e)}$.

For E, $\Gamma_{me|(e)}$ is weak immune as action i can always be chosen, yielding positive utility. We again try propagating this result, and also here conclude correctly that Γ_{me} is weak immune for E: all choices of M (whom we do not assume to be honest in the analysis of E), lead to either non-negative utility for E or to a subtree which is weak immune for E.

The definition for collusion resilience *against* a given player group is similar to Definition 5.2, by lifting the quantifier over the player subgroups $S \subset N$ to the front of the formula.

Definition 5.4 (Collusion Resilience against a Player Group). A subgame $\Gamma_{|h}$ of game Γ with honest history h^* is *collusion resilient against a group of players* $S \subset N$, if there exists a strategy $\sigma \in \mathcal{S}_{|h}$ such that no matter to which strategy $\tau \in \mathcal{S}_{|h}$ the players in S deviate, their joint utility will be not greater than their honest joint utility and, if h is along h^*, then also $H_{|h}(\sigma) = h^*_{|h}$:

$$\exists \sigma \in \mathcal{S}_{|h}.\ (h \text{ along } h^* \rightarrow H_{|h}(\sigma) = h^*_{|h}) \wedge \qquad (\mathrm{cr}_S(\Gamma_{|h}))$$
$$\forall \tau \in \mathcal{S}_{|h}.\ \sum_{p \in S} u_{|h,p}(\sigma) \geq u_{|h,p}(\tau_S, \sigma_{N-S})\ .$$

Defining practicality for a single player, though, requires slight changes: instead of considering an arbitrary player p, we define practicality for that player whose turn it is in the considered subtree.

Definition 5.5 (Practicality for the Current Player). A subgame $\Gamma_{|h}$ of a game Γ with honest history h^* is *practical for the current player*, if there exists a strategy $\sigma \in \mathcal{S}_{|h}$ such that in each further subtree $\Gamma_{|(h,g)}$ no matter to which strategy $\tau \in \mathcal{S}_{|(h,g)}$ the current player $P(h,g)$ deviates, the utility of $P(h,g)$ in the subtree will not increase strictly and, if h is along h^*, then also $H_{|h}(\sigma) = h^*_{|h}$:

$$\exists \sigma \in \mathcal{S}_{|h}.\quad (h \text{ along } h^* \rightarrow H_{|h}(\sigma) = h^*_{|h}) \ \wedge \forall g \in \mathscr{H}_{|h}\ \forall \tau \in \mathcal{S}_{|(h,g)}. \qquad (\mathrm{pr}_P(\Gamma_{|h}))$$
$$u_{|(h,g),P(h,g)}(\sigma_{|g}) \geq u_{|(h,g),P(h,g)}(\tau_{P(h,g)}, \sigma_{|g,N-P(h,g)})\ .$$

We now state our first crucial result towards compositionality: stratification of security analysis over players.

Theorem 5.6 (Player-Wise Security Properties). *A game Γ satisfies a security property iff it satisfies the respective security property player-wise. That is, the following equivalences hold:*

(1) Γ *weak immune* $\Leftrightarrow \forall p \in N.\ \Gamma$ *weak immune for* p.
(2) Γ *weaker immune* $\Leftrightarrow \forall p \in N.\ \Gamma$ *weaker immune for* p.
(3) Γ *collusion resilient* $\Leftrightarrow \forall S \subset N.\ \Gamma$ *collusion resilient against* S.
(4) Γ *practical* $\Leftrightarrow \Gamma$ *practical for the current player.*

5.2 Splitting and Combining Player-Wise Security Properties

Theorem 5.6 proves that the security analysis of a game can be carried out player-wise, instead of analyzing interactions between all (groups of) players. We now show that not only players can be treated individually, but *(super)game security can also be split into subgame security*. That is, the security of a supergame can be proven by proving player-wise security over subgames. This implies compositional game-theoretic security is sound and complete.

Theorem 5.7 (Compositional Game-Theoretic Security). *The game-theoretic security of an EFG Γ with honest history h^* can be computed compositionally. That is, the only information needed of a subtree $\Gamma_{|h}$, to decide whether Γ satisfies security property is*

- *for weak(er) immunity: for which players $p \in N$ the subtree $\Gamma_{|h}$ is weak(er) immune;*
- *for collusion resilience: against which player groups $S \subset N$ the subtree $\Gamma_{|h}$ is collusion resilient;*
- *for practicality:*
  - *if h is along h^*: whether $h^*_{|h}$ is practical in $\Gamma_{|h}$;*
  - *if h is not along h^*: the set $\mathbb{U}(h)$ containing all practical utilities of $\Gamma_{|h}$[4]. A utility $u(t)$ after terminal history $t \in \mathcal{T}$ is practical in subgame $\Gamma_{|h}$ iff t is practical in $\Gamma_{|h}$.*

Theorems 5.8, 5.10 and 5.12 establish how to compositionally compute player-wise security for each security property, yielding a constructive proof of Theorem 5.7.

Theorem 5.8 (Compositional Weak Immunity). *Let Γ be an EFG with honest history h^* and $p \in N$ a player. The following hold.*

(1) *A leaf of Γ is weak immune for p iff p's utility is non-negative:*

$$\forall t \in \mathcal{T}.\ wi_p(\Gamma_{|t}) \quad \Leftrightarrow \quad u_p(t) \geq 0\ .$$

(2) *A branch of Γ is weak immune for p, where p is not the current player, iff all children are weak immune for p:*

$$\forall h \in \mathcal{H} \setminus \mathcal{T}.\ p \neq P(h) \ \Rightarrow\ \big(wi_p(\Gamma_{|h}) \ \Leftrightarrow\ \forall a \in A(h).\ wi_p(\Gamma_{|(h,a)})\big)\ .$$

(3) *A branch of Γ along the honest history h^* is weak immune for the current player p, iff the child following h^* is weak immune for p. Let $a^* \in A(h)$ be the honest choice, i.e. (h, a^*) along h^*, then:*

$$\forall h \in \mathcal{H} \setminus \mathcal{T}.\ p = P(h) \wedge h \text{ along } h^* \quad \Rightarrow \quad \big(wi_p(\Gamma_{|h}) \ \Leftrightarrow\ wi_p(\Gamma_{|(h,a^*)})\big)\ .$$

(4) *A branch of Γ off the honest history h^* is weak immune for the current player p, iff there exists a child that is weak immune for p:*

$$\forall h \in \mathcal{H} \setminus \mathcal{T}.\ p = P(h) \wedge h \text{ off } h^* \quad \Rightarrow \quad \big(wi_p(\Gamma_{|h}) \ \Leftrightarrow\ \exists a \in A(h).\ wi_p(\Gamma_{|(h,a)})\big)\ .$$

Similar results to Theorem 5.8 hold for weaker immunity.

[4]The set $\mathbb{U}(h)$ is introduced properly in the paragraph before Theorem 5.12.

Example 5.9 (Compositional Weak Immunity). We revisit the Market Entry game Γ_{me} of Figure 1, with honest history (n). We compute that Γ_{me} is weak immune using our compositional approach, where we stratify over players first and then split Γ_{me} into subtrees.

We start with player M. Theorem 5.8 implies that Γ_{me} is weak immune for M iff $\Gamma_{me|(n)}$ is weak immune for M; since $\Gamma_{me|(n)}$ is a leaf, we must check that the utility of M is non-negative, i.e. $0 \geq 0$. As this is true, game Γ_{me} is weak immune for M.

Next, E. According to Theorem 5.8, the game Γ_{me} is weak immune iff $\Gamma_{me|(n)}$ and $\Gamma_{me|(e)}$ are weak immune for E. The subgame $\Gamma_{me|(n)}$ is weak immune for E if their utility is non-negative, i.e. $p \geq 0$, true by assumption. The subtree $\Gamma_{me|(e)}$ is now weak immune for E iff either $\Gamma_{me|(e,i)}$ or $\Gamma_{me|(e,pw)}$ is. E's utility at $\Gamma_{me|(e,i)}$ is $p/2 \geq 0$. Therefore Γ_{me} is weak immune for E, and from Theorem 5.6 it follows that Γ_{me} is weak immune.

Theorem 5.10 (Compositional Collusion Resilience). *Let Γ be an EFG with honest history h^* and honest utility $u^* = u(h^*)$. The following equivalences hold.*

(1) *A leaf of Γ is collusion resilient against $S \subset N$ iff the honest joint utility of the deviating players $p \in S$ is greater than or equal to their joint utility at that leaf:*

$$\forall t \in \mathscr{T}.\ cr_S(\Gamma_{|t}) \quad \Leftrightarrow \quad \sum_{p \in S} u^*_p \geq \sum_{p \in S} u_p(t)\ .$$

(2) *A branch of Γ, where the current player is in the deviating group $S \subset N$, is collusion resilient against S iff all children are collusion resilient against S:*

$$\forall h \in \mathscr{H} \setminus \mathscr{T}.\ P(h) \in S \quad \Rightarrow \quad \big(cr_S(\Gamma_{|h}) \ \Leftrightarrow\ \forall a \in A(h).\ cr_S(\Gamma_{|(h,a)})\big)\ .$$

(3) *A branch of Γ along the honest history h^*, where the current player is not in the deviating group $S \subset N$, is collusion resilient against S iff the child following h^* is collusion resilient against S. Let $a^* \in A(h)$ be the honest action, i.e. (h, a^*) along h^*, then:*

$$\forall h \in \mathscr{H} \setminus \mathscr{T}.\ P(h) \notin S \wedge h \text{ along } h^* \quad \Rightarrow \quad \big(cr_S(\Gamma_{|h}) \ \Leftrightarrow\ cr_S(\Gamma_{|(h,a^*)})\big)\ .$$

(4) *A branch of Γ off the honest history h^*, where the current player is not in the deviating group $S \subset N$, is collusion resilient against S iff there exists a child that is collusion resilient against S:*

$$\forall h \in \mathscr{H} \setminus \mathscr{T}.\ P(h) \notin S \wedge h \text{ off } h^* \quad \Rightarrow \quad \big(cr_S(\Gamma_{|h}) \ \Leftrightarrow\ \exists a \in A(h).\ cr_S(\Gamma_{|(h,a)})\big)\ .$$

Note that, if a player is in a deviating group, all child subtrees need to be collusion resilient even if we are along honest history, as the deviator might choose any action and potentially harm honest players. In contrast, for an honest player and a node off honest history, there needs to merely exist one collusion resilient child that the player can choose to defend against the deviating group.

Example 5.11 (Compositional Collusion Resilience). We compositionally compute the collusion resilience of the Market Entry game Γ_{me} (Figure 1) with honest history (n). We have two possible colluding groups, both singletons $\{M\}$ and $\{E\}$.

Consider $\{M\}$. At the root of Γ_{me}, since the player M is in the colluding group, all subtrees must be collusion resilient against $\{M\}$. Along the honest history we reach a leaf $\Gamma_{me|(n)}$, which is collusion resilient (it is the honest leaf). For subtree $\Gamma_{me|(e)}$ there needs to exist a collusion resilient child, which is the case in the leaf after (e, pw): utility $-a$ is strictly smaller than the honest utility 0.

Next, $\{E\}$. At the root, M is not in the deviating group. Hence, only the honest child $\Gamma_{me|(n)}$ need be collusion resilient against $\{E\}$, which it is, as it is the honest leaf; so the utility is equal to the honest one in part (1) of Theorem 5.10. This suffices to establish collusion resilience against $\{E\}$; checking $\Gamma_{me|(e)}$ is unnecessary.

Using Theorem 5.6, it follows that Γ_{me} is collusion resilient.

While we only have to remember boolean values of subtrees (i.e. whether it satisfies the property) to compositionally compute weak(er) immunity and collusion resilience per player(group), we need more information when reasoning about practicality: To correctly assess whether a tree is practical, one has to know the utility tuples of *all* practical terminal histories of its subtrees. For a given subtree $\Gamma_{|h}$ with practical terminal histories $\{t_1, \ldots, t_n\}$, we define the set of practical utilities $\mathbb{U}(h)$ as $\mathbb{U}(h) := \{u_{|h}(t_i) : i = 1, \ldots, n\}$.

Theorem 5.12 (Compositional Practicality). *Let Γ be an EFG with honest history h^* and $\mathbb{U}(h)$ be the set of practical utilities of subtree $\Gamma_{|h}$. Let u^* be the honest utility $u^* = u(h^*)$. Then the following identities and equivalences hold.*

(1) *In a leaf of Γ the only practical utility is that of the leaf.*

$$\forall t \in \mathscr{T}\,.\, \mathbb{U}(t) = \{u(t)\}\ .$$

(2) *The honest utility u^* is practical in a branch of Γ along h^* iff it is practical in the child following h^* and if for every other child at least one practical utility is not greater than u^* for the current player. Let $a^* \in A(h)$ be the honest action after h, then:*

$$\forall h \in \mathscr{H} \setminus \mathscr{T}\,.\, h \text{ along } h^* \quad \Rightarrow \quad \big(pr(\Gamma_{|h}) \quad \Leftrightarrow \\ pr(\Gamma_{|(h,a^*)}) \ \wedge\ \forall a \in A(h) \setminus \{a^*\}\ \exists u \in \mathbb{U}((h,a))\,.\, u^*_{P(h)} \geq u_{P(h)}\big)\ .$$

(3) *A utility is practical in a branch of Γ off the honest history h^* iff it is practical in a child and if, for every other child, at least one practical utility is not greater for the current player.*

$$\forall h \in \mathscr{H} \setminus \mathscr{T}\,.\, h \text{ off } h^* \quad \Rightarrow \quad \big(\forall t \in \mathscr{T}_{|h}.\, u(t) \in \mathbb{U}(h) \quad \Leftrightarrow \\ \exists a \in A(h).\, u(t) \in \mathbb{U}((h,a)) \wedge \\ \forall a' \in A(h) \setminus \{a\}\ \exists u' \in \mathbb{U}((h,a'))\,.\, u_{P(h)}(t) \geq u'_{P(h)}\big)\ .$$

Example 5.13 (Compositional Practicality). To compositionally compute the practicality of the Market Entry game Γ_{me} of Figure 1 with honest history (n), we start with the leaves of the tree, where the practical utilities are the utilities of the leaves. Moving upwards in the tree, we look at the subtree $\Gamma_{me|(e)}$, which is off the honest history, so we take the better utility for player E, setting $\mathbb{U}(e) = \{(\frac{p}{2}, \frac{p}{2})\}$. At the root of the tree, which is along the honest history, the practical utility of the honest subtree $(0, p)$ should be practical. Since all practical utilities of the non-honest child (there is just one) are better for player M (as $\frac{p}{2} > 0$), the honest utility is not practical. Theorems 5.6 and 5.12 then imply that Γ_{me} is not practical.

6 Automating Compositional Security Analysis

Section 5 assumed a total order $\preceq$ on game utility terms T_u. This section lifts this fixed ordering constraint and interprets the game variables in the utility terms T_u as real-valued variables $\vec{x}$, as explained in Section 3.2. Theorem 3.11 showed that quantification of the variables $\vec{x}$ can be done equivalently by grouping values of $\vec{x}$ that satisfy the same total order $(\preceq, T_u)$. In this section we combine this result with Theorem 5.6 and further pull the security property quantifications (over players, subgames and strategies) out of the variable $\vec{x}$ quantification, as they are independent. For weak immunity, for example, the formula (1) (with $sp = wi$) becomes equivalent to:

$$\forall(\preceq, T_u)\ \forall p \in N\ \exists \sigma \in \mathcal{S}\,.\, H(\sigma) = h^* \wedge \forall \tau \in \mathcal{S}\ .\\ \forall \vec{x}.\left(\bigwedge_{c \in C \cup \preceq} c[\vec{x}]\right) \rightarrow u_p(\sigma_p, \tau_{N-p})[\vec{x}] \geq 0\,. \tag{3}$$

This mapping of game-theoretic security from Theorem 3.11 to the player-wise security of Theorem 5.6 is crucial for automating compositional security: we only forward relatively small first-order expressions of the form

$$\forall \vec{x}. \left(\bigwedge_{c \in C \cup \preceq} c[\vec{x}] \right) \rightarrow ut_1[\vec{x}] \geq ut_2[\vec{x}] \,, \tag{4}$$

to an SMT solver, where ut_1 and ut_2 are term expressions over $\vec{x}$; checking such formulas is very feasible for SMT solvers.

As usual, to check whether (4) is a theorem, the property is first negated, and then an SMT solver is used to check satisfiability. This is where the simplified quantified structure of (4) becomes especially *friendly for automation*: The SMT solving of (4) happens in a purely existential fragment, for which efficient decision procedures exist [Barrett and Tinelli 2018; Bjørner and Nachmanson 2024]. The remaining reasoning in (3), about the players and the existence of strategies σ witnessing player-wise security, is performed using the compositional security results of Theorems 5.8, 5.10 and 5.12, without burdening the SMT solver. Such an interplay between SMT solving and compositional security eases automation, as illustrated below and detailed further in Section 6.1.

Example 6.1 (SMT Reasoning for Compositional Security). Revisiting the Market Entry game Γ_{me} of Figure 1, we study the SMT formulae resulting from (4), given initial constraints $C = \{a > 0, p > 0\}$. All symbolic utility terms occurring in the security properties of Γ_{me} are already totally ordered by the constraints in C. Hence, the only relevant total order $\preceq$ here that is consistent with C is $-a \prec 0 \prec p/2 \prec p$.

To analyze, for example, the weak immunity of the Market Entry game for player E compositionally, we follow the algorithm induced by Theorem 5.8. Starting at the root, we observe that we are *not* at a leaf and E is *not* the current player. Thus, by Theorem 5.8 it follows that we need to make sure that the subgames after history (n) and (e) are both weak immune for player E. The SMT reasoning is only needed when we reach a leaf, such as the one after history (n). The resulting SMT query is

$$\forall a, p.\ a > 0 \land p > 0 \rightarrow p \geq 0 \,,$$

which is trivially valid. Hence, the subtree after history (n) is weak immune for E for all allowed utility values. The analysis for the subgame after history (e) can be performed in an analogous way. Finally, we can combine the results for both subtrees to obtain the result for the supertree. In this manner, the inductive approach allows us to reduce reasoning about secure strategies to reasoning about utility terms in the leaves of the game tree.

6.1 Divide-and-Conquer Algorithms for Compositional Security

Our compositionality results from Theorem 5.6 and Theorems 5.8, 5.10 and 5.12, extended by a lazy total-order approach, induce a *divide-and-conquer approach for splitting and combining reasoning over game subtrees and supertrees*. Our overall divide-and-conquer framework for automating compositional game-theoretic reasoning is summarized in Algorithm 1, which in turn relies upon Algorithm 2 as well as upon Algorithms 3 to 5 from the appendix. We compositionally compute the game-theoretic security of a protocol, analyzing the (protocol model) game for all real-valued variables $\vec{x}$ of utitilty terms T_u, considering all total orders $\preceq$ at once. If we fail, we split the total orders into multiple cases, unless we can conclude that the respective security property cannot be satisfied even if we restrict the values of $\vec{x}$ to one total order $\preceq$. The case split we consider is induced by an SMT query as in property (4) when some but not all $\vec{x}$ satisfy the implication. We then split into total orders that enforce $ut_1[\vec{x}] \geq ut_2[\vec{x}]$, respectively $ut_1[\vec{x}] < ut_2[\vec{x}]$, in (4).

Algorithm 1: Function `SatisfiesProperty` for Compositional Game-Theoretic Security Reasoning.

input : input instance $\Pi = (\Gamma, inf, C)$, honest history h^*, the name of a security property $sp \in \{wi, weri, cr, pr\}$, and the currently analyzed case (as set of SMT constraints) `case`.
output : `true` if Π satisfies sp in case `case`, `false` otherwise

```
1  S ← ∅
2  AddConstraints (S, C ∪ case)
3  result ← true
4  split ← null
5  for pg ∈ RelevantGroups(Π, sp) do
6  |  (result_pg, split_pg) ← ComputeSP (Π,h*,S,sp,pg)
7  |  if result_pg = false then
8  |  |  result ← result_pg
9  |  |  split ← split_pg
10 |  |  break
11 |  end
12 end
13 if result = true then
14 |  return true
15 end
16 if split = null then
17 |  return false
18 end
19 for constr ∈ {split, ¬split} do
20 |  if ¬SatisfiesProperty(Π, h*, sp, case ∪ {constr}) then
21 |  |  return false
22 |  end
23 end
24 return true
```

Algorithm 1: Function `SatisfiesProperty`. In Algorithm 1 an instance Π, which contains the game tree Γ, the set of infinitesimal variables inf (as introduced in Section 2), and the set of initial constraints C, is given as input. The input to Algorithm 1 also contains the honest history h^*, the security property to be analyzed, and the currently considered case `case`.

The function `SatisfiesProperty` in Algorithm 1 is called initially with the empty case to analyze all total orders. This case can be refined throughout Algorithm 1, using case splits. Hence, in the first call of the function, the set S, representing the constraints handed to an SMT solver, contains only the initial constraints C. The relevant player groups `RelevantGroups` of security property sp are set according to the stratified definitions of sp from Section 5.1: N for $w(er)i$, as we stratify over players; $2^N \setminus \{\emptyset, N\}$ for cr, as we stratify over deviating subgroups; and {"none"} for pr.

The function `ComputeSP` in line 6 of Algorithm 1 stands for `ComputeWI`, `ComputeCR` or `ComputePR` (Algorithms 2, 4 and 5), depending on the security property sp, as summarized in Algorithm 3. The result of `ComputeSP` depends on whether Γ with honest history h^* satisfies property sp for/against pg, given the constraints in S. Here, we also keep track of utility comparisons we cannot decide.

Importantly, the constraint $ut_1[\vec{x}] \geq ut_2[\vec{x}]$ to whether Γ satisfies sp in `case` is returned as split_{pg}, if it exists.

The loop in lines 5–12 of Algorithm 1 incorporates player-wise security from Theorem 5.6. It additionally provides a necessary case split if the security property is violated for a player group. Subsequently, the respective results are returned: `true` for all groups yields `true`; `false` but nothing to split on for at least one group yields `false`; and `false` together with a split leads to further case splits (lines 19–24). If we split the total orders into multiple cases, all the cases have to return `true` for the property to be satisfied.

Example 6.2. Let us revisit the Market Entry game from Example 2.3, but this time let us assume only $p > 0$ is the initial constraint and $a \in \mathbb{R}$ can take any value. We check whether the honest history (n) is collusion resilient. Algorithm 1 will in line 5 consider each singleton player group individually, suppose we start with $\{M\}$. The function `ComputeCR`, specified in Algorithm 4, returns $(\texttt{false}, 0 \geq -a)$, as the comparison between 0 and $-a$ is missing to determine collusion resilience. So the result is set to `false` and split to $0 \geq -a$. For the colluding group $\{E\}$ the function `ComputeCR` returns $(\texttt{true}, \texttt{null})$, as the honest player M can choose a collusion resilient honest action. Algorithm 1 then proceeds with line 19, refining the constraints by first adding $0 \geq -a$ to the case. The function `SafisfiesProperty` will return `true` (and empty split); and then adding the negated constraint $0 < -a$ to the case, at which point `SafisfiesProperty` will return `false`, since M can profit by deviating from the honest action (n), as both $\frac{p}{2}$ and $-a$ are better utilities than 0. Algorithm 1 thus terminates by returning `false`.

The security-property-specific function variants of `ComputeSP` recursively apply the compositional results of Theorem 5.7. To illustrate case splitting of total orders, we only describe function `ComputeWI` of Algorithm 2 below.

Algorithm 2: Function `ComputeWI`. The function `ComputeWI` of Algorithm 2 is initially called with the entire game tree Γ from function `SatisfiesProperty` of Algorithm 1. We then proceed recursively, according to Theorem 5.8. Note that the player group pg is just one player.

In a leaf, `GetUtility` in Algorithm 2 returns $ut(\vec{x})$ as the utility of player pg. We then – in line 2 of Algorithm 2 – check whether the constraints in S together with $ut(\vec{x}) < 0$ are `unsat`. This is equivalent to the constraints in S implying $ut(\vec{x}) \geq 0$, which is an instance of property (4), except that we do not (necessarily) have one total order $\preceq$ at hand, only some constraints from `case`. If the implication holds, we return `true`. Otherwise, we check the opposite condition, by asking in line 5 of Algorithm 2 whether

$$\forall \vec{x}. \bigwedge_{c \in C \cup \text{case}} c[\vec{x}] \rightarrow ut[\vec{x}] < 0 \tag{5}$$

holds. If it does (line 6), the leaf is not weak immune. Otherwise (line 8), the leaf's weak immunity depends on the total order, which induces a case split on $ut[\vec{x}] \geq 0$.

At a branch (lines 10–32 of Algorithm 2), we check in which of the cases of Theorem 5.8 we are. We then call the function `ComputeWI` recursively on immediate subgames $\Gamma_{|(a)}$ and propagate the result accordingly. Note that, for simplicity, in line 13 of Algorithm 2 we do not wait for a `null` split that would immediately return `false`, but rather proceed with a split. However, as there are only finitely many possible case splits, we will eventually see the `null` split for a `false` subtree if it exists and return it to reach the correct result.

Example 6.3. We mimic the execution of the function `ComputeWI` from Algorithm 2 on the Market Entry game from Example 2.3, but this time only assume $a > 0$ is the initial constraint, and $p \in \mathbb{R}$ can take any value. Suppose we enter Algorithm 2 with the entire tree Γ_{me}, honest history (e, i)

Algorithm 2: Function `ComputeWI` for Weak Immunity.

input : game tree Γ, honest history h^*, set S containing initial constraints and current case, player group pg.

output : (result, split), where result states whether Γ is weak immune for pg, given S, and split a crucial utility comparison we cannot decide.

```
1  if isLeaf(Γ) then
2      if Check(S, GetUtility(Γ, pg) < 0) = unsat then
3          return (true, null)
4      end
5      if Check(S, GetUtility(Γ, pg) ≥ 0) = unsat then
6          return (false, null)
7      end
8      return (false, GetUtility(Γ, pg) ≥ 0)
9  end
10 if CurrentPlayer(Γ) ≠ pg then
11     for a ∈ Actions(Γ) do
12         (result, split) ← ComputeWI(Γ_|(a), h*, S, pg)
13         if result = false then
14             return (result, split)
15         end
16     end
17     return (true, null)
18 end
19 if AlongHonest(Γ, h*) then
20     a* ← HonestAction(Γ, h*)
21     return ComputeWI(Γ_|(a*), h*, S, pg)
22 end
23 newsplit ← null
24 for a ∈ Actions(Γ) do
25     (result, split) ← ComputeWI(Γ_|(a), h*, S, pg)
26     if result = true then
27         return (true, null)
28     else if split ≠ null then
29         newsplit ← split
30     end
31 end
32 return (false, newsplit)
```

and player $pg = M$. Since the root of the tree is along the honest history, the function will jump to line 19, and recursively call `ComputeWI` for the honest subtree $\Gamma_{me|(e)}$. Then the current player E is not pg, so we proceed with line 10, and iterate through the actions pw and i. Suppose we first look at the action pw and from line 12 recursively compute the weak immunity for the leaf after (e, pw). Algorithm 2 will execute lines 1 and 2, and since the utility of player M is 0, which is a non-negative number, the check in line 2 will be unsat, so the function returns (`true`, null). For the other action i, we recursively compute (line 12 of the algorithm) the weak immunity for the leaf after (e, i). The function `GetUtility`$(\Gamma_{me|(e,i)}, M)$ will return $\frac{p}{2}$, for which we cannot decide

whether it is non-negative (there are no initial constraints on p). Both conditions from lines 2 and 5 are thus false and we return the pair $(\texttt{false}, \frac{p}{2} \geq 0)$ in line 8. Proceeding from the supertree $\Gamma_{me|(e)}$ in line 12, with the result being `false`, we return in line 14 the pair $(\texttt{false}, \frac{p}{2} \geq 0)$.

Theorem 6.4 (Correctness of Algorithm 1). *The compositional approach to compute the game-theoretic security of an input instance* Π *for honest history* h^* *described in Algorithm 1 is sound and complete. That is,* $\texttt{SatisfiesProperty}(\Pi, h^*, sp, \emptyset) = \texttt{true}$ *iff* Π *with honest history* h^* *satisfies the property* sp*. Otherwise, it returns* `false`.

In addition to compositional security via Algorithm 1, our work supports additional features to debug a protocol and better understand its structure. Those include (i) strategy extraction in case the considered security property was satisfied (Section 6.2), (ii) finding counterexamples (Section 6.3), and (iii) providing weakest preconditions to make the game secure otherwise. Computing preconditions in our compositional setting can be done via collecting all cases, where the security property is violated, and then conjoin and negate them afterwards.

6.2 Extracting Compositional Strategies

The way compositional security analysis in Algorithm 1 works, unfortunately, does not immediately provide witness strategies. However, Algorithm 1 can still carry around enough information to compute witnesses.

Theorem 6.5 (Weak(er) Immune Strategies). *For a weak(er) immune game* Γ*, with honest history* h^* *and total order* $\preceq$*, strategy* σ *is honest and weak(er) immune for all* $\vec{x}$ *satisfying* $\preceq$*, where*

$$\sigma := (\sigma^{p_1}, \ldots, \sigma^{p_{|N|}}) ,$$

and $\sigma^{p_i} \in \mathcal{S}_{p_i}$ *is a strategy for player* p_i*. Strategy* σ^{p_i} *picks the honest choice along the honest history, whereas at other nodes, where it is* p_i*'s turn, it picks an arbitrary action* a *that yields a weak(er) immune for* p_i *subtree after action* a*.*

Theorem 6.5 is constructive in nature, yielding thus an algorithmic approach for extracting a weak(er) immune strategy. For each player pg, function `ComputeWI` (and `ComputeWERI`) proceeds as follows. If it is their turn after history h, h off h^*, and we found a weak(er) immune choice, we store this action as the choice of a possible weak(er) immune and honest strategy σ. If the game is weak(er) immune for all players, we can simply compute σ by collecting all the stored choices throughout the tree.

Example 6.6. We compute the weak immune strategy of the Market Entry game from Example 2.3 with honest history (n), which was analyzed as in Example 5.9. The strategy σ^M for player M has to choose the honest action n at the root, which is the only choice point for M. The strategy σ^E for player E needs to choose one weaker immune subtree after history (e). Since the subtree after history (e, i) is the only candidate, we set $\sigma^E(e) = i$. The strategy $\sigma = (\sigma^M, \sigma^E)$ is the desired weak immune strategy.

Collusion resilience and practicality also admit elegant methods for deriving compositional strategies, as detailed in Appendix D.

6.3 Finding Compositional Counterexamples

Counterexamples to the security properties, as defined in Definitions 3.12, 3.14 and 3.16, serve the important purpose of providing attack vectors and thus pinpointing the weaknesses of a protocol underlying the considered game model. We use the following *pseudo-algorithms* to compute counterexamples compositionally.

Compositional Counterexamples to Weak(er) Immunity. We first store information during Algorithm 2: When analyzing the weak(er) immunity for a player p, whenever it is not p's turn and there exists an action leading to a not weak(er) immune subtree (line 14 with split = null in Algorithm 2), we store the action, the current history and the player p.

Secondly, after the analysis terminated and the result was not weak(er) immune, we generate a counterexample to the weak(er) immunity of player p by walking through the tree again. Assume the current history is h and we proceed from the root as follows.

- If p is the current player and h is along the honest history, we follow the honest action to a subtree. This is sufficient since an honest p follows the honest history.
- If it is p's turn but h is not along the honest history, all choices had to lead to not weak(er) immune for p subtrees for the current tree to be not weak(er) immune for p. We, therefore, have to follow all choices to compute a counterexample.
- Otherwise, if it is not p's turn, we check our stored data for a not weak(er) immune for p choice a. By construction and using Theorem 5.8, it has to exist. We add it to our partial strategy s_{N-p}, i.e. $s_{N-p}(h) = a$. Then, we continue at history (h, a).
- At a leaf nothing has to be considered. A not weak(er) immune for p leaf leads to a negative (real) utility for p.

According to Theorem 5.8, the steps outlined above provide a player p and a partial strategy s_{N-p} for the other players $N-p$, no matter how the honest p behaves off the honest history. It also yields only negative utilities for p and it thus provides a counterexample to the weak(er) immunity of p and, therefore, a counterexample to the weak immunity of the game with the considered honest history.

It is also possible to compute *all* counterexamples to weak(er) immunity. This can be done by simply storing *all* actions that lead to not weak(er) immune subtrees.

Example 6.7. Let us adapt the Market Entry game from Example 2.3 by changing the initial constraint on the variable p to $p < 0$. The honest history (n) is not weak immune for player E, as they get a negative utility $p < 0$ in the honest leaf. We can thus construct the counterexample as follows: starting from the root, it is not E's turn and the not weak immune choice is (n), so we add the action n to the partial strategy for player M. We then continue at history (n), which is a leaf, so we are done.

Counterexamples to collusion resilience and practicality can also be computed, which is detailed in Appendix D.

7 Experimental Evaluation

We implemented the compositional security approach of Section 5 by exploiting its divide-and-conquer reasoning nature from Section 6. Our implementation is available online in the CheckMate2.0 tool [5].

Experimental Setup. We evaluated our tool using a machine with 2 AMD EPYC 7502 CPUs clocked at 2.5GHz with 32 cores and 1TB RAM using 16 game-theoretic security benchmarks. Our dataset contains the 15 benchmarks from [Rain et al. 2024], listed in Table 3, which include realistic models of real-world blockchain protocols along with game scenarios of various sizes, such as models of an auction (*Auction*) or tic-tac-toe (*Tic Tac Toe*, which is modeled infeasibly on purpose). The games *Closing*, *3-Player Routing*, *Unlocking Routing* are detailed models with up to 36,000 tree nodes of different phases of Bitcoin's Lightning protocol [Poon and Dryja 2016]. Additionally, we

[5] https://github.com/apre-group/checkmate/releases/tag/OOPSLA25

detail later in this section one large example (over 100 million nodes), named *4-Player Routing*, in order to showcase the impact of interleaved sub- and supertree reasoning. 4-Player Routing is yet another realistic model of a phase of the Lightning protocol.

To the best of our knowledge, the only other automated approach for game-theoretic security is the CheckMate framework [Rain et al. 2024]. Our experiments also compare CheckMate2.0 to CheckMate.

Experimental Results. Tables 1 and 2 summarize our experiments, with further details in Appendix E. We report both on the results of CheckMate2.0 and CheckMate; the respective columns on times, nodes, and calls in Tables 1 and 2 detail these comparisons. In particular, the columns "Nodes evaluated" and "Nodes evaluated (reps)" of Table 1 indicate the number of game tree nodes visited during the security analysis without and, respectively, with repetitions. The "Calls" column of Table 1 shows the number of calls made to the SMT solver while proving the security property listed in column 4.

	Game	Nodes	Players	Property	Secure yes/no	Time	Nodes evaluated CheckMate2.0/CheckMate	Nodes evaluated (reps)	Calls
medium-sized games	Pirate	79	4	wi	n	0.010 / 0.015	10 / 79	10 / 316	5 / 1
	(y, n, n			cr	n	0.041 / 0.029	79 / 79	622 / 1,106	368 / 4
	n, y, y)			pr	n	0.036 / 0.049	79 / 79	482 / 79	554 / 8
	Auction	92	4	wi	n	0.012 / 0.033	16 / 92	16 / 368	9 / 1
	(E, E, I, I)			cr	n	0.018 / 0.030	66 / 92	128 / 1,288	103 / 1
real-world models	Closing	221	2	wi	y	0.011 / 0.024	20 / 221	22 / 442	16 / 1
	(H)			weri	y	0.010 / 0.021	20 / 221	22 / 442	16 / 1
				cr	y	0.012 / 0.023	44 / 221	46 / 442	36 / 1
				pr	n	0.097 / 0.346	221 / 221	568 / 221	1454 / 1
	(C_h, S)			wi	y	0.011 / 0.024	33 / 221	36 / 442	25 / 1
				weri	y	0.011 / 0.020	33 / 221	36 / 442	25 / 1
				cr	y	0.013 / 0.023	60 / 221	63 / 442	48 / 1
				pr	y	2.144 / 0.345	221 / 221	14353 / 221	38220 / 1
	3-Player	21,688	3	wi	n	0.248 / 0.984	16 / 21,688	16 / 65,064	9 / 1
	Routing			weri	y	0.514 / 1.008	7,084 / 21,688	7,570 / 65,064	5,441 / 1
	($S_H, L, L,$			cr	n	0.272 / 1.886	430 / 21,688	474 / 130,128	299 / 1
	U, U)			pr	n	33.162 / 34.717	21,688 / 21,688	416,156 / 21,688	569,418 / 13
	Tic Tac Toe	549,946	2	wi	y	5.276 / 255.368	18,026 / 549,946	18,036 / 1,099,892	10,694 / 1
	($CM, RU, LU,$			weri	y	5.256 / 255.600	18,026 / 549,946	18,036 / 1,099,892	10,694 / 1
	$RD, RM, LM,$			cr	y	5.302 / 286.574	18,026 / 549,946	18,036 / 1,099,892	10,694 / 1
	CU, CD, LD)			pr	y	36.530 / TO	549946 / TO	549946 / TO	527198 / TO

Table 1. Selected experimental results of game-theoretic security, using the compositional CheckMate2.0 approach and the non-compositional CheckMate setting of [Rain et al. 2024]. A full summary of our experiments is in Appendix E. Runtimes are given in seconds, with timeout (TO) after 8 hours. For each game, columns 2–3 list the size (tree nodes and game players) of the game from column 1. Column 4 shows the game-theoretic security property we analyzed and (dis)proved, as indicated in column 5. Columns 6–9 present the results of CheckMate2.0 compared to CheckMate, using the slash / sign.

Experimental Analysis. Table 1 demonstrates that the compositional approach of CheckMate2.0 significantly outperforms the non-compositional CheckMate setting in execution time across nearly all benchmarks. The scalability of CheckMate2.0 is especially evident in the Tic Tac Toe benchmark, which involves a substantial 548,946 nodes. In this example, for the properties weak immunity (`wi`), weaker immunity (`weri`), and collusion resilience (`cr`), CheckMate2.0 completes the security analysis in approximately 5 seconds, whereas CheckMate requires between 255 and 287 seconds. When proving practicality (`pr`) of Tic Tac Toe, the conventional CheckMate fails to terminate within 8 hours while CheckMate2.0 succeeds in less than 37 seconds.

In some benchmarks, where a security property is not satisfied, CheckMate2.0 explores significantly fewer nodes, see 3-Player Routing for weak immunity and collusion resilience, the Pirate game for weak immunity, and Auction for weak immunity and collusion resilience.

We note that CheckMate2.0 requires considerably more SMT-solving calls. Notable examples include the Closing Game (38,220 CheckMate2.0 calls vs. 1 CheckMate call for practicality), 3-Player Routing (546,418 vs. 13 calls for practicality), and Tic Tac Toe (10,694 vs. 1 call for weak(er) immunity and collusion resilience). Despite the higher number of SMT calls in CheckMate2.0, the SMT queries generated by CheckMate2.0 are considerably smaller than the ones of CheckMate; moreover, CheckMate2.0 calls inhabit a quantifier-free fragment, easing reasoning significantly as reflected in the improved execution times.

In general, CheckMate2.0 analysis may occasionally result also in suboptimal splits, leading to longer execution times. This issue is exemplified in the Closing game when analyzing practicality of the honest history (C_h, S). Additionally, analyzing collusion resilience can sometimes take longer, particularly when more players are involved, for example in the Pirate game. This might be explained by the very large number of colluding groups combined with a small game resulting in many trivial SMT calls compared to CheckMate.

Game	Property	Time (one CE) CheckMate2.0/CheckMate	Time (all CEs) CheckMate2.0/CheckMate
Pirate (y, n, n, n, y, y)	cr	0.041 / 0.039	3.232 / 79.839
Auction (E, E, I, I)	wi	0.012 / 0.048	0.025 / 4.172
	cr	0.018 / 0.066	0.036 / 15.106
3-Player Routing (S_H, L, L, U, U)	wi	0.251 / 1.925	5.909 / 110.716
	cr	0.279 / 5.619	1.657 / 7.815
	pr	33.561 / 46.480	291.236 / 3 033.784

Table 2. Selected experiments on counterexample (CE) generation using our CheckMate2.0 approach and the non-compositional CheckMate tool of [Rain et al. 2024]. Full details and experiments are given Appendix E. Runtimes are given in seconds.

Counterexamples and Strategies. Table 2 presents the CheckMate2.0 runtimes to generate counterexamples compared to CheckMate, for selected benchmarks. It reports the execution time required to find one counterexample (for one case split) as well as finding all counterexamples in all cases for violated security properties. The former is useful for quickly identifying scenarios where the property is not met, while the latter proves particularly helpful when revising and refining a protocol. Comprehensive data for all benchmarks can be found in Appendix E.

The use of compositionality in CheckMate2.0 demonstrates notable improvements in execution time, particularly when retrieving all counterexamples. Additionally, the execution times for compositional analysis with and without counterexample extraction are quite similar, indicating that CheckMate2.0 enables counterexample extraction with minimal overhead. The counterexamples to collusion resilience for the Pirate game show this clearly. While CheckMate2.0 requires slightly more time for property analysis compared to CheckMate, we note that the new CheckMate2.0 identifies all counterexamples across all cases in approximately 3 seconds, whereas CheckMate takes almost 80 seconds. Similarly, in the case of the 3-Player Routing game, CheckMate2.0 retrieves all counterexamples for all cases within 291 seconds, while it takes CheckMate over 3,000 seconds (50 minutes).

Similar benefits of CheckMate2.0 can also be observed for strategy extraction, with details in Appendix E.

Sub- and Supertree Reasoning. One of the most significant contributions of the compositional reasoning is that CheckMate2.0 enables analyzing subtrees independently and integrating only their security results in the supertree. This feature of CheckMate2.0 is particularly beneficial in larger models. For instance, the 3-Player Routing and Routing Unlocking benchmarks based on the routing protocol [Poon and Dryja 2016] are generated using a script, as it is not feasible to model protocols of this size manually. Modeling the routing protocol for 3 players results in a game with 21,688 nodes (3-Player Routing), taking 20 MB on disk.

We next detail a more challenging routing example with 4 players, called *4-Player Routing*, which has 144,342,306 nodes. This example exceeds our 200 GB of allocated disk space, and thus could not even be created fully. However, by leveraging compositionality, we intertwine model generation and analysis, making it possible to discard generated subtrees after the results of security analysis have been obtained. Specifically, during the game generation process, each subtree corresponding to a specific phase of the protocol called unlocking phase (a total of 1440 subtrees) is analyzed on the fly, with only the results kept. The final outcome, the *4-Player Routing* game, is a supertree with 396 regular nodes and 1440 nodes representing subtrees, or 1,836 nodes in total. The supertree has a size of about 60 MB and in it all subtrees for the unlocking phase have already been solved. This allows us to directly apply CheckMate2.0 to the supertree. Using CheckMate2.0 compositionally, we conclude that 4-Player Routing is weaker immune, but not weak immune, nor collusion resilient, nor practical.

We note that game modeling is not in the scope of this paper, but interleaving the modeling and the analysis of a game, as described in the paragraph above, makes it feasible to verify even complex real-world models with over 100 million nodes.

8 Related Work and Conclusions

We present the first approach to compositionally analyze the security properties of game-theoretic protocol models. By mapping our work to SMT-based reasoning in combination, we introduce a divide-and-conquer framework to automate compositional reasoning in a sound and complete manner. Our experiments clearly showcase scalability improvements, especially for real-world protocols with millions of nodes/actions.

We believe that our compositional reasoning framework generalizes to other tree search properties that are (in-)equations of utility terms quantified over strategies, histories, and player groups, provided the property can be analyzed player(group)-wise. By memorizing some subtree data, it should be possible to decide such a property given all subtree results. Other common game-theoretic properties, such as Nash Equilibria for example, are in fact covered by our security properties and are thus guaranteed to be of compositional nature.

Our compositional approach is a strong enhancement over the non-compositional setting of [Brugger et al. 2023]. While the definitions of the security properties are the same, their encoding is different: we not only improve practical usage but also provide a sound and complete way to split and combine game-theoretic properties of subgames/supergames. Compared to [Brugger et al. 2023], we minimize the use of SMT solving by applying it only over game leaves.

Compositional game theory, without considering game-theoretic security, is also addressed in [Bolt et al. 2023; Ghani et al. 2018, 2020]. Here, so-called open games are introduced to represent games played relative to a given environment. Open games are, however, restricted to constant numeric utilities and assuming rational behavior of players. Unlike these works, we work with symbolic utilities and capture honest/rational behavior, and thus security, in games.

Divide-and-conquer approaches to parallelize SMT queries have also been studied [Wilson et al. 2023], proposing partitioning strategies to decompose an SMT problem into subproblems. Our approach avoids generating large SMT queries altogether, utilizing instead local tree reasoning.

Related to compositional verification, [Wesley et al. 2021] presents compositional analysis of smart contracts. Instead of verifying a smart contract relative to all users, a few representative users are chosen, thereby avoiding intractability due to state explosion. While game-theoretic security is not addressed in [Wesley et al. 2021], program verification and synthesis are worthy approaches to be further considered in our future work.

Importantly, (automatically) synthesizing game models from the protocol's definition, respectively source code in the case of smart contracts, is a challenge we aim to address in the future. Allowing infinite games and modeling game actions impacted by external factors are other tasks for future work, allowing us to model uncontrollable protocol effects, such as price changes.

9 Data-Availability Statement

The software that implements the techniques described in Section 5 and Section 6 and supports the evaluation results reported in Section 7 is available on Zenodo[6].

Acknowledgments

This research was funded in whole or in part by the ERC Consolidator Grant ARTIST 101002685, the Austrian Science Fund (FWF) SPyCoDe Grant 10.55776/F85, the WWTF Grant ForSmart 10.47379/ICT22007, the TU Wien Doctoral College SecInt, the Amazon Research Award 2023 QuAT, and a Netidee Fellowship 2022.

[6] https://zenodo.org/records/15725152

References

Clark Barrett and Cesare Tinelli. 2018. Satisfiability Modulo Theories. In *Handbook of Model Checking*, Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (Eds.). Springer International Publishing, Cham, 305–343. https://doi.org/10.1007/978-3-319-10575-8_11

Nikolaj Bjørner and Lev Nachmanson. 2024. Arithmetic Solving in Z3. In *Computer Aided Verification: 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24–27, 2024, Proceedings, Part I* (Montreal, QC, Canada). Springer-Verlag, Berlin, Heidelberg, 26–41. https://doi.org/10.1007/978-3-031-65627-9_2

Sam Blackshear, Nikos Gorogiannis, Peter W. O'Hearn, and Ilya Sergey. 2018. RacerD: compositional static race detection. *Proc. ACM Program. Lang.* 2, OOPSLA, Article 144 (Oct. 2018). https://doi.org/10.1145/3276514

Bruno Blanchet. 2014. Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif. In *Foundations of Security Analysis and Design VII: FOSAD 2012/2013 Tutorial Lectures*, Alessandro Aldini, Javier Lopez, and Fabio Martinelli (Eds.). Springer International Publishing, Cham, 54–87. https://doi.org/10.1007/978-3-319-10082-1_3

Joe Bolt, Jules Hedges, and Philipp Zahn. 2023. Bayesian open games. *Compositionality* 5 (Oct. 2023), 9. https://doi.org/10.32408/compositionality-5-9

Lea Salome Brugger, Laura Kovács, Anja Petkovic Komel, Sophie Rain, and Michael Rawson. 2023. CheckMate: Automated Game-Theoretic Security Reasoning. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security* (Copenhagen, Denmark) *(CCS '23)*. Association for Computing Machinery, New York, NY, USA, 1407–1421. https://doi.org/10.1145/3576915.3623183

Vitalik Buterin. 2014. A Next Generation Smart Contract & Decentralized Application Platform. *white paper* 3 (2014). Issue 37. https://ethereum.org/en/whitepaper/

Neil Ghani, Jules Hedges, Viktor Winschel, and Philipp Zahn. 2018. Compositional Game Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science* (Oxford, United Kingdom) *(LICS '18)*. Association for Computing Machinery, New York, NY, USA, 472–481. https://doi.org/10.1145/3209108.3209165

Neil Ghani, Clemens Kupke, Alasdair Lambert, and Fredrik Nordvall Forsberg. 2020. Compositional Game Theory with Mixed Strategies: Probabilistic Open Games Using a Distributive Law. *Electronic Proceedings in Theoretical Computer Science* 323 (Sept. 2020), 95–105. https://doi.org/10.4204/eptcs.323.7

Nadim Kobeissi, Georgio Nicolas, and Mukesh Tiwari. 2020. Verifpal: Cryptographic Protocol Analysis for the Real World. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop* (Virtual Event, USA) *(CCSW'20)*. Association for Computing Machinery, New York, NY, USA, 159. https://doi.org/10.1145/3411495.3421365

Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. 2013. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification*. Springer Berlin Heidelberg, Berlin, Heidelberg, 696–701. https://doi.org/10.1007/978-3-642-39799-8_48

Satoshi Nakamoto. 2009. Bitcoin: A Peer-to-Peer Electronic Cash System. *white paper* (2009). https://bitcoin.org/en/bitcoin-paper

Martin J. Osborne and Ariel Rubinstein. 1994. *A Course in Game Theory*. The MIT Press, Cambridge, USA.

Joseph Poon and Thaddeus Dryja. 2016. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. *white paper* (2016). https://lightning.network/lightning-network-paper.pdf

Sophie Rain, Georgia Avarikioti, Laura Kovács, and Matteo Maffei. 2023. Towards a Game-Theoretic Security Analysis of Off-Chain Protocols. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*. IEEE Computer Society, Los Alamitos, CA, USA, 107–122. https://doi.org/10.1109/CSF57540.2023.00003

Sophie Rain, Lea Salome Brugger, Anja Petković Komel, Laura Kovács, and Michael Rawson. 2024. Scaling CheckMate for Game-Theoretic Security. In *Proceedings of 25th Conference on Logic for Programming, Artificial Intelligence and Reasoning (EPiC Series in Computing, Vol. 100)*, Nikolaj Bjørner, Marijn Heule, and Andrei Voronkov (Eds.). EasyChair, Stockport, UK, 222–231. https://doi.org/10.29007/llnq

Raymond M Smullyan. 1995. *First-Order Logic*. Dover Publications, New York.

Yuepeng Wang, Shuvendu K. Lahiri, Shuo Chen, Rong Pan, Isil Dillig, Cody Born, Immad Naseer, and Kostas Ferles. 2020. Formal Verification of Workflow Policies for Smart Contracts in Azure Blockchain. In *Verified Software. Theories, Tools, and Experiments*, Supratik Chakraborty and Jorge A. Navas (Eds.). Springer International Publishing, Cham, 87–106. https://doi.org/10.1007/978-3-030-41600-3_7

Scott Wesley, Maria Christakis, Jorge A. Navas, Richard Trefler, Valentin Wüstholz, and Arie Gurfinkel. 2021. Compositional Verification of Smart Contracts Through Communication Abstraction. In *Static Analysis: 28th International Symposium, SAS 2021, Chicago, IL, USA, October 17–19, 2021, Proceedings* (Chicago, IL, USA). Springer-Verlag, Berlin, Heidelberg, 429–452. https://doi.org/10.1007/978-3-030-88806-0_21

Amalee Wilson, Andres Nötzli, Andrew Reynolds, Byron Cook, Cesare Tinelli, and Clark W. Barrett. 2023. Partitioning Strategies for Distributed SMT Solving. *2023 Formal Methods in Computer-Aided Design (FMCAD)* (2023), 199–208. https://api.semanticscholar.org/CorpusID:259129858

Paolo Zappalà, Marianna Belotti, Maria Gradinariu Potop-Butucaru, and Stefano Secci. 2020. Game theoretical framework for analyzing Blockchains Robustness. *IACR Cryptol. ePrint Arch.* 2020 (2020). https://api.semanticscholar.org/CorpusID:219616790

A Compositionality Proofs

We restate the theorems for readability.

Theorem 3.11 (Game-Theoretic Security with Total Orders). *For an EFG Γ with honest history h^* and a finite set of initial constraints C, property (1) is equivalent to*

$$\forall(\preceq, T_u)\ \exists\sigma \in \mathcal{S}.\, H(\sigma) = h^* \wedge \quad \forall\vec{x}.\left(\bigwedge_{c\in C\cup\preceq} c[\vec{x}]\right) \rightarrow sp(\sigma)[\vec{x}]\,. \tag{2}$$

Proof. To show implication "$\Leftarrow$", we pick arbitrary values for the variables $\vec{x}$ that satisfy all initial constraints in C. We then consider the total order $\preceq$ on T_u that is consistent with the choice of values for variables $\vec{x}$. By Equation (2), there has to exist a strategy σ yielding the honest history such that for all $\vec{x}$ consistent with $\preceq$ and satisfying the initial constraints C the security property $sp(\sigma)[\vec{x}]$ holds. Since the picked $\vec{x}$ satisfies $\preceq$ and C, the strategy σ works. Hence, 1 is implied.

For implication "$\Rightarrow$", let $\preceq$ be an arbitrary total order over the symbolic terms T_u and $\vec{x}$ values that satisfy $\preceq$ as well as C. Then, from (1) there exists an honest strategy σ such that $sp(\sigma)[\vec{x}]$. Note that whether $sp(\sigma)[\vec{x}]$ holds, depends only on the relation of terms in T_u and not on the actual values of $\vec{x}$. Therefore, $sp(\sigma)[\vec{x}]$ is true for one $\vec{x}$ iff it is true for all $\vec{x}$ consistent with the same term order $\preceq$. Thus, 2 holds. □

Theorem 5.1 (Equivalence of Non-Secure Games). *A game Γ with honest history h^* violates one of the security properties of weak(er) immunity, collusion resilience, or practicality iff there exists a history h along the honest history h^* such that $\Gamma_{|h}$ violates the respective security property.*

Proof. The direction "Γ violates security property $\Rightarrow \exists h \in \mathscr{H}.\, h$ along $h^* \wedge \Gamma_{|h}$ violates security property" is easy to show. Just pick the trivial history $h = \emptyset$, which is always along the honest history.

For the other direction, we consider the security properties individually. We start with *weak immunity* and assume that $\Gamma_{|h}$ is along the honest history h^* and is not weak immune. We fix an arbitrary strategy $\sigma \in \mathcal{S}$ in the entire game Γ that yields the honest history $H(\sigma) = h^*$ and show it is not weak immune. To do so, we consider $\sigma_{|h} \in \mathcal{S}_{|h}$ which still yields the honest history in the subgame: $H_{|h}(\sigma_{|h}) = h^*_{|h}$. Since $\Gamma_{|h}$ is not weak immune, there exists a player $p \in N$ and a strategy $\tau^h \in \mathcal{S}_{|h}$ such that $u_{|h,p}(\tau^h_{N-p}, \sigma_{|h,p}) < 0$. We now construct a strategy $\tau \in \mathcal{S}$ extending τ^h. I.e. $\tau_{|h} = \tau^h$. Everywhere else we set τ to be identical to σ.

Since σ yields the honest history, $H(\sigma) = h^*$, and $\Gamma_{|h}$ lies along the honest history, we get $H(\tau_{N-p}, \sigma_p) = (h, H_{|h}(\tau_{|h,N-p}, \sigma_{|h,p}))$. By construction of τ also $H(\tau_{N-p}, \sigma_p) = (h, H_{|h}(\tau^h_{N-p}, \sigma_{|h,p}))$ holds. Thus, their utilities have to be identical $u_p(\tau_{N-p}, \sigma_p) = u_{|h,p}(\tau^h_{N-p}, \sigma_{|h,p}) < 0$. Hence, σ is not weak immune and as it was chosen arbitrarily, it follows that Γ with honest history h^* is not weak immune. The proof for *weaker immunity* is analog.

The proof of the second direction for *collusion resilience* follows the same idea. We pick an arbitrary $\sigma \in \mathcal{S}$ yielding the honest history h^* and show it is not collusion resilient, assuming that $\Gamma_{|h}$ is along the honest history and is not collusion resilient. Hence there exists a deviating group $S \subset N$ and a strategy $\tau^h \in \mathcal{S}_{|h}$ such that $\sum_{p\in S} u_{|h,p}(\sigma_{|h}) < \sum_{p\in S} u_{|h,p}(\sigma_{|h,N-S}, \tau^h_S)$. We again construct a strategy $\tau \in \mathcal{S}$ extending τ^h: $\tau_{|h} = \tau^h$. Everywhere else τ is identical to σ. Applying the same reasoning as before, we know that $H(\sigma_{N-S}, \tau_S) = (h, H_{|h}(\sigma_{|h,N-S}, \tau^h_S))$ as well

as $H(\sigma) = (h, H(\sigma_{|h}))$, which implies

$$\sum_{p\in S} u_p(\sigma) = \sum_{p\in S} u_{|h,p}(\sigma_{|h}) < \sum_{p\in S} u_{|h,p}(\sigma_{|h,N-S}, \tau_S^h) = \sum_{p\in S} u_p(\sigma_{N-S}, \tau_S) \ . \tag{6}$$

Therefore, σ is not collusion resilient, and since it was chosen arbitrarily, it follows that Γ with honest history h^* is not collusion resilient.

Proving this for *practicality* is straightforward: we again pick an arbitrary honest strategy $\sigma \in \mathscr{H}$ and assume $h \in \mathscr{H}$ is along the honest history and $\Gamma_{|h}$ is not practical. I.e. there exists a history $k \in \mathscr{H}_{|h}$, a player $p \in N$ and a strategy $\tau \in \mathcal{S}_{|(h,k)}$ such that $u_{|(h,k),p}(\sigma_{|(h,k)}) < u_{|(h,k),p}(\tau_p, \sigma_{|(h,k),N-p})$. Using now $\ell := (h, k) \in \mathscr{H}$, the player p and τ, one can see easily that

$$u_{|\ell,p}(\sigma_{|\ell}) < u_{|\ell,p}(\tau_p, \sigma_{|\ell,N-p})$$

and therefore σ is not practical and neither is Γ. □

Theorem 5.6 (Player-Wise Security Properties). *A game Γ satisfies a security property iff it satisfies the respective security property player-wise. That is, the following equivalences hold:*

(1) Γ *weak immune* $\Leftrightarrow \forall p \in N.\ \Gamma$ *weak immune for* p.
(2) Γ *weaker immune* $\Leftrightarrow \forall p \in N.\ \Gamma$ *weaker immune for* p.
(3) Γ *collusion resilient* $\Leftrightarrow \forall S \subset N.\ \Gamma$ *collusion resilient against* S.
(4) Γ *practical* $\Leftrightarrow \Gamma$ *practical for the current player.*

Proof of Theorem 5.6.1-2. We start with equivalence (1) for weak immunity.
Implication "$\Rightarrow$". By Definition 3.1 the game Γ, with honest history h^*, is weak immune if

$$\exists \sigma \in \mathcal{S}.\ H(\sigma) = h^* \ \wedge\ \forall p \in N\ \forall \tau \in \mathcal{S}.\ u_p(\sigma_p, \tau_{N-p}) \geq 0 \ . \tag{wi}$$

Assuming Γ is weak immune, we consider such a strategy and call it σ'. The right-hand side of the equivalence is

$$\forall p \in N\ \exists \sigma \in \mathcal{S}.\ H(\sigma) = h^* \ \wedge\ \forall \tau \in \mathcal{S}.\ u_p(\sigma_p, \tau_{N-p}) \geq 0 \ . \tag{wi$_{\forall p}$}$$

Therefore, we can just pick σ' for all the players and are done.

Implication "$\Leftarrow$". We assume Equation (wi$_{\forall p}$), i.e. Γ is weak immune for all players $p_i \in N$, $i = 1, \ldots, n$, where $n = |N|$. Let their corresponding weak immune for p_i strategies be $\sigma^i \in \mathcal{S}$, for $i = 1, \ldots, n$. Further let $\sigma \in \mathcal{S}$ be a new strategy constructed from the σ^i in the following way $\sigma := (\sigma^1_{p_1}, \ldots, \sigma^n_{p_n})$. That means at a history h where it is player p_i's turn, the choice in σ is the one from σ^i: $\sigma(h) = \sigma^i(h)$. The strategy σ yields the honest history $H(\sigma) = h^*$, since all σ^i yield h^* and combining strategies that extend the same history leads to the new strategy extending the same one.

It remains to show that σ is weak immune, i.e. that this one strategy works for all the players (right conjunct in Equation (wi)). We therefore consider an arbitrary player $p_i \in N$ and an arbitrary strategy $\tau \in \mathcal{S}$. By construction of σ and the fact that σ^i is weak immune for p_i, it follows $u_{p_i}(\sigma_{p_i}, \tau_{N-p_i}) = u_{p_i}(\sigma^i_{p_i}, \tau_{N-p_i}) \geq 0$. This concludes the proof that σ is weak immune, which implies that Γ is weak immune.

The proof of Theorem 5.6.2 is analog. □

Proof of Theorem 5.6.3. *Implication "⇒".* For readability, let us restate the formula for collusion resilience of Γ

$$\exists \sigma \in \mathcal{S}.\ H(\sigma) = h^* \ \wedge\ \forall S \subset N\ \forall \tau \in \mathcal{S}.\ \sum_{p\in S} u_p(\sigma) \geq \sum_{p\in S} u_p(\tau_S, \sigma_{N-S})\ , \tag{cr}$$

as well as for collusion resilience against all subgroups of the game Γ

$$\forall S \subset N\ \exists \sigma \in \mathcal{S}.\ H(\sigma) = h^* \ \wedge\ \forall \tau \in \mathcal{S}.\ \sum_{p\in S} u_p(\sigma) \geq \sum_{p\in S} u_p(\tau_S, \sigma_{N-S}). \tag{cr$_{\forall S}$}$$

The implication then again follows from the definitions *cr* and *cr*$_{\forall S}$. The one strategy σ in *cr* can be used for all $S \subset N$ in *cr*$_{\forall S}$.

Implication "⇐". We prove this direction by contraposition, showing $\neg cr \Rightarrow \neg cr_{\forall S}$. In [Brugger et al. 2023], it is shown that $\neg cr$ is equivalent to the existence of a counterexample (Definition 3.14): There exists a set of deviating players S together with their strategy s_S such that s_S extended by any strategy σ'_{N-S} of players $N-S$, which follows the honest history h^*, yields a terminal history $H(\sigma'_{N-S}, s_S) = t_{\sigma'_{N-S}}$ with

$$\sum_{p\in S} u_p(t_{\sigma'_{N-S}}) > \sum_{p\in S} u_p(h^*) \tag{7}$$

Let a group of deviating players $S \subset N$ and their strategy s_S be a counterexample. To show $\neg cr_{\forall S}$ we fix an arbitrary $\sigma \in \mathcal{S}$ that yields the honest history h^*. For the strategy $\tau = (s_S, \sigma_{N-S})$ we have by Equation (7):

$$\begin{aligned} \sum_{p\in S} u_p(\sigma) &= \sum_{p\in S} u_p(h^*) \\ < \sum_{p\in S} u_p(H(\sigma_{N-S}, s_S)) &= \sum_{p\in S} u_p(\tau_S, \sigma_{N-S}). \end{aligned}$$

Therefore, $\neg cr_{\forall S}$ holds and implication "⇐" is proven.

□

Proof of Theorem 5.6.4. *Implication "⇒".* We again restate the formulae for better readability. Practicality of Γ:

$$\exists \sigma \in \mathcal{S}.\ H(\sigma) = h^* \ \wedge\ \forall h \in \mathscr{H}\ \forall p \in N\ \forall \tau \in \mathcal{S}_{|h}.\ u_{|h,p}(\sigma_{|h}) \geq u_{|h,p}(\sigma_{|h,N-p}, \tau_p) \tag{pr}$$

and practicality for the current player of Γ:

$$\exists \sigma \in \mathcal{S}.\ H(\sigma) = h^* \ \wedge \forall h \in \mathscr{H}\ \forall \tau \in \mathcal{S}_{|h}.\ u_{|h,P(h)}(\sigma_{|h}) \geq u_{|h,P(h)}(\sigma_{|h,N-P(h)}, \tau_{P(h)})\ . \tag{pr$_{P(h)}$}$$

It is now easy to see that *pr* implies *pr*$_{P(h)}$, since for all $h \in \mathscr{H}$, $P(h) \in N$.

Implication "⇐". Assuming Equation (*pr*$_{P(h)}$) holds, we fix σ that satisfies it and show σ also satisfies Equation (*pr*). Let $h \in \mathscr{H}$, $p \in N$ and $\tau \in \mathcal{S}_{|h}$ as in (*pr*) be arbitrary.

If $p = P(h)$, then the inequality $u_{|h,p}(\sigma_{|h}) \geq u_{|h,p}(\sigma_{|h,N-p}, \tau_p)$ is immediately implied by *pr*$_{P(h)}$. If $p \neq P(h)$, we consider the first point in $H_{|h}(\sigma_{|h})$ where it is player p's turn and call the corresponding history $t \in \mathscr{H}_{|h}$. In case no such t exists, then $H_{|h}(\sigma_{|h}) = H_{|h}(\sigma_{|h,N-p}, \tau_p)$, and hence $u_{|h,p}(\sigma_{|h}) = u_{|h,p}(\sigma_{|h,N-p}, \tau_p)$.

If such a t exists, we know from ($pr_{P(h)}$) that $u_{|(h,t),p}(\sigma_{|(h,t)}) \geq u_{|(h,t),p}(\sigma_{|(h,t),N-p}, \tau_{|t,p})$, since $p = P(h,t)$. From the fact that t lies along $H_{|h}(\sigma_{|h})$ follows $u_{|(h,t),p}(\sigma_{|(h,t)}) = u_{|h,p}(\sigma_{|h})$, and as player p has no choices in t, it follows

$$H_{|h}(\sigma_{|h,N-p}, \tau_p) = (t, H_{|(h,t)}(\sigma_{|(h,t),N-p}, \tau_{|t,p}))$$

which leads to identical utilities. Therefore, the inequality in *pr* is also proven for $p \neq P(h)$. This concludes the proof of the theorem, as h, p, and τ were chosen arbitrarily. □

Theorem 5.8 (Compositional Weak Immunity). *Let Γ be an EFG with honest history h^* and $p \in N$ a player. The following hold.*

(1) *A leaf of Γ is weak immune for p iff p's utility is non-negative:*

$$\forall t \in \mathcal{T}.\ wi_p(\Gamma_{|t}) \quad \Leftrightarrow \quad u_p(t) \geq 0\ .$$

(2) *A branch of Γ is weak immune for p, where p is not the current player, iff all children are weak immune for p:*

$$\forall h \in \mathcal{H} \setminus \mathcal{T}.\, p \neq P(h) \Rightarrow \left(wi_p(\Gamma_{|h}) \Leftrightarrow \forall a \in A(h).\, wi_p(\Gamma_{|(h,a)})\right).$$

(3) *A branch of Γ along the honest history h^* is weak immune for the current player p, iff the child following h^* is weak immune for p. Let $a^* \in A(h)$ be the honest choice, i.e. (h, a^*) along h^*, then:*

$$\forall h \in \mathcal{H} \setminus \mathcal{T}.\, p = P(h) \wedge h \text{ along } h^* \quad \Rightarrow \quad \left(wi_p(\Gamma_{|h}) \Leftrightarrow wi_p(\Gamma_{|(h,a^*)})\right).$$

(4) *A branch of Γ off the honest history h^* is weak immune for the current player p, iff there exists a child that is weak immune for p:*

$$\forall h \in \mathcal{H} \setminus \mathcal{T}.\, p = P(h) \wedge h \text{ off } h^* \quad \Rightarrow \quad \left(wi_p(\Gamma_{|h}) \Leftrightarrow \exists a \in A(h).\, wi_p(\Gamma_{|(h,a)})\right).$$

Proof. We start by proving 1). By Definition 3.1, there has to exist a strategy $\sigma \in \mathcal{S}_{|t}$ such that for all $\tau \in \mathcal{S}_{|t}$ $u_{|t,p}(\sigma_p, \tau_{N-p}) \geq 0$. But for a terminal history $t \in \mathcal{T}$, the utility function $u_{|t,p} = u_p(t)$ is just a constant and the only strategy in $\mathcal{S}_{|t}$ is the empty strategy. Thus, equivalence 1) holds by definition.

For equivalence 2), let $h \in \mathcal{H} \setminus \mathcal{T}$ be a non-terminal history such that p is not the current player $p \neq P(h)$. We prove implication "⇒" first: Assuming $\Gamma_{|h}$ is weak immune for p, there exists a strategy $\sigma \in \mathcal{S}_h$ such that – if h is along h^*, it yields the honest history and – for all $\tau \in \mathcal{S}_{|h}$ the player p's utility $u_{|h,p}(\sigma_p, \tau_{N-p}) \geq 0$. Let now $a \in A(h)$ be an arbitrary choice after history h. We consider $\sigma_{|(a)} \in \mathcal{S}_{|(h,a)}$ – if (h,a) along h^*, then $H_{|(h,a)}(\sigma_{|(a)}) = h^*_{|(h,a)}$. Let further $\tau^a \in \mathcal{S}_{|(h,a)}$ be arbitrary. For $\tau' \in \mathcal{S}_{|h}$, such that $\tau'_{|(a)} = \tau^a$ and $\tau'(h) = a$ we know by assumption that $u_{|h,p}(\sigma_p, \tau'_{N-p}) \geq 0$. Since $p \neq P(h)$ and $\tau'(h) = a$, it follows $H_{|h}(\sigma_p, \tau'_{N-p}) = (a, H_{|(h,a)}(\sigma_{|(a),p}, \tau^a_{N-p}))$. Therefore, $u_{|(h,a),p}(\sigma_{|(a),p}, \tau^a_{N-p}) = u_{|h,p}(\sigma_p, \tau'_{N-p}) \geq 0$. As τ' was chosen arbitrarily, $\sigma_{|(a)}$ was constructed based on a, and a was also chosen arbitrarily, it follows the right-hand side of the equivalence: $\forall a \in A(h).\, wi_p(\Gamma_{|(h,a)})$.

To show implication "⇐", we assume $\forall a \in A(h).\, wi_p(\Gamma_{|(h,a)})$ and we construct a $\sigma' \in \mathcal{S}_{|h}$ weak immune for p. For all $a \in A(h)$ let $\sigma'_{|(a)} := \sigma^a$, where $\sigma^a \in \mathcal{S}_{|(h,a)}$ weak immune for p – and if (h,a) along h^*, then σ^a also yields the honest history. Such strategies have to exist according to our assumptions. If h is along h^*, then define $\sigma'(h) := a^*$, where a^* is the next choice in h^* after h, which makes $H_{|h}(\sigma') = h^*_h$. Otherwise, let it be arbitrary. For an arbitrary $\tau \in \mathcal{S}_{|h}$, call $\tau(h) = a'$. Since $p \neq P(h)$, follows $H_{|h}(\sigma'_p, \tau_{N-p}) = (a', H_{|(h,a')}(\sigma^a_p, \tau_{|(h,a),N-p}))$. Hence, as σ^a was weak immune for p, we get $u_{|h,p}(\sigma'_p, \tau_{N-p}) = u_{|(h,a'),p}(\sigma^a_p, \tau_{|(h,a),N-p}) \geq 0$. Strategy τ was chosen arbitrarily; therefore, $\Gamma_{|h}$ is weak immune for player p.

We proceed by showing equivalence 3). Let $h \in \mathcal{H} \setminus \mathcal{T}$ be such that $p = P(h)$ and h along h^*. We further fix the honest choice after h to be $a^* \in A(h)$, i.e. (h, a^*) is along h^*. Let an honest (i.e.

h^*-yielding) strategy $\sigma' \in \mathcal{S}_{|h}$ and an honest strategy $\sigma^{a^*} \in \mathcal{S}_{|(h,a^*)}$ be connected in the following way: $\sigma'_{|(a^*)} = \sigma^{a^*}$. Then, one of them is weak immune for p iff the other is. Assume σ' is weak immune for p. Pick an arbitrary $\tau^{a^*} \in \mathcal{S}_{|(h,a^*)}$, any extension of τ^{a^*} to a $\tau \in \mathcal{S}_{|h}$ has the property $\tau^{a^*}_{N-p} = \tau_{N-p}$, since $p = P(h)$. As σ' is weak immune for p, we know $u_{|h,p}(\sigma'_p, \tau_{N-p}) \geq 0$. Further, by definition of σ^{a^*}, follows $H_{|h}(\sigma'_p, \tau_{N-p}) = (a^*, H_{|(h,a^*)}(\sigma^{a^*}_p, \tau^{a^*}_{N-p}))$ and thus $u_{|(h,a^*),p}(\sigma^{a^*}_p, \tau^{a^*}_{N-p}) \geq 0$. Strategy $\tau^{a^*} \in \mathcal{S}_{|(h,a^*)}$ was arbitrary, so σ^{a^*} is weak immune for p. The other direction works the same way.

For equivalence 4), let $h \in \mathscr{H} \setminus \mathcal{T}$ be such that $p = P(h)$ and h not along h^*. For implication "$\Rightarrow$", we assume $\sigma \in \mathcal{S}$ is weak immune for p in $\Gamma_{|h}$. Let $a := \sigma(h) \in A(h)$ and $\sigma^a := \sigma_{|(a)} \in \mathcal{S}_{|(h,a)}$. We now fix an arbitrary $\tau^a \in \mathcal{S}_{|(h,a)}$, let $\tau \in \mathcal{S}_{|h}$ be any extension of τ^a, i.e. $\tau_{|(a)} = \tau^a$. Then, since $p = P(h)$, we know $H_{|h}(\sigma_p, \tau_{N-p}) = (a, H_{|(h,a)}(\sigma^a_p, \tau^a_{N-p}))$, which further implies $u_{|(h,a),p}(\sigma^a_p, \tau^a_{N-p}) = u_{|h,p}(\sigma_p, \tau_{N-p}) \geq 0$. As τ^a was chosen arbitrarily and as we are not along h^*, that means σ^a is weak immune for p in $\Gamma_{(h,a)}$, which proves the implication. For the other direction "$\Leftarrow$", the same reasoning applies, when we assume $\sigma^a \in \mathcal{S}_{|(h,a)}$ is weak immune for p in $\Gamma_{(h,a)}$ and construct $\sigma \in \mathcal{S}$ such that $\sigma(h) = a$, $\sigma_{|(a)} = \sigma^a$ and the rest arbitrary. This concludes the last implication and, therefore, the proof of the theorem. □

Theorem 5.10 (Compositional Collusion Resilience). *Let Γ be an EFG with honest history h^* and honest utility $u^* = u(h^*)$. The following equivalences hold.*

(1) *A leaf of Γ is collusion resilient against $S \subset N$ iff the honest joint utility of the deviating players $p \in S$ is greater than or equal to their joint utility at that leaf:*

$$\forall t \in \mathcal{T}\,.\, cr_S(\Gamma_{|t}) \quad \Leftrightarrow \quad \sum_{p\in S} u^*_p \geq \sum_{p\in S} u_p(t)\ .$$

(2) *A branch of Γ, where the current player is in the deviating group $S \subset N$, is collusion resilient against S iff all children are collusion resilient against S:*

$$\forall h \in \mathscr{H} \setminus \mathcal{T}\,.\, P(h) \in S \quad \Rightarrow \quad \big(cr_S(\Gamma_{|h}) \ \Leftrightarrow\ \forall a \in A(h).\ cr_S(\Gamma_{|(h,a)})\big)\ .$$

(3) *A branch of Γ along the honest history h^*, where the current player is not in the deviating group $S \subset N$, is collusion resilient against S iff the child following h^* is collusion resilient against S. Let $a^* \in A(h)$ be the honest action, i.e. (h, a^*) along h^*, then:*

$$\forall h \in \mathscr{H} \setminus \mathcal{T}\,.\, P(h) \notin S \wedge h \text{ along } h^* \quad \Rightarrow \quad \big(cr_S(\Gamma_{|h}) \ \Leftrightarrow\ cr_S(\Gamma_{|(h,a^*)})\big)\ .$$

(4) *A branch of Γ off the honest history h^*, where the current player is not in the deviating group $S \subset N$, is collusion resilient against S iff there exists a child that is collusion resilient against S:*

$$\forall h \in \mathscr{H} \setminus \mathcal{T}\,.\, P(h) \notin S \wedge h \text{ off } h^* \quad \Rightarrow \quad \big(cr_S(\Gamma_{|h}) \ \Leftrightarrow\ \exists a \in A(h).\ cr_S(\Gamma_{|(h,a)})\big)\ .$$

Proof. We show equivalence 1) first. By Definition 5.4 there has to exist a $\sigma \in \mathcal{S}_{|t}$ (yielding h^* if applicable) such that for all $\tau \in \mathcal{S}_{|t}$ the inequality $\sum_{p\in S} u^*_p \geq \sum_{p\in S} u_{|t,p}(\sigma_{N-S}, \tau_S)$ holds. Since t is a terminal history, the utility function $u_{|t,p} = u_p(t)$ is constant and the only existing strategy is the empty strategy. Hence, the collusion resilience against S of $\Gamma_{|t}$ is equivalent to $\sum_{p\in S} u^*_p \geq \sum_{p\in S} u_p(t)$.

To show claim 2), we assume $h \in \mathscr{H} \setminus \mathcal{T}$ is a non-terminal history at which a player from the deviating group S has a turn $P(h) \in S$. We start with implication "$\Rightarrow$", assuming $\sigma \in \mathcal{S}$ is a collusion resilient strategy against S in $\Gamma_{|h}$. We pick an arbitrary action $a \in A(h)$ and define strategy $\sigma^a := \sigma_{|(a)} \in \mathcal{S}_{|(h,a)}$. If (h, a) is along the honest history h^*, then so is h. Thus the collusion resilient against S strategy σ yields the honest history $H_{|h}(\sigma) = h^*_{|h}$ which implies $H_{|(h,a)}(\sigma^a) = h^*_{|(h,a)}$. Let $\tau^a \in \mathcal{S}_{|(h,a)}$ be arbitrary. We extend it to $\tau \in \mathcal{S}_{|h}$ by defining $\tau(h) = a$, $\tau_{|(a)} = \tau^a$ and letting

the rest be arbitrary. With this construction and the fact that $P(h) \in S$ we get $H_{|h}(\sigma_{N-S}, \tau_S) = (a, H_{|(h,a)}(\sigma^a_{N-S}, \tau^a_S))$. Therefore, also $\sum_{p\in S} u^*_p \geq \sum_{p\in S} u_{|h,p}(\sigma_{N-S}, \tau_S) = \sum_{p\in S} u_{|(h,a),p}(\sigma^a_{N-S}, \tau^a_S)$, by the collusion resilience against S of σ. As τ^a was chosen arbitrarily, and σ^a was constructed for an arbitrary a the implication was proven.

For the other direction of 2) "$\Leftarrow$", we assume for all $a \in A(h)$ that $\sigma^a \in \mathcal{S}_{|(h,a)}$ is collusion resilient against S. Let $\sigma \in \mathcal{S}_{|h}$ be such that for all $a \in A(h)$ $\sigma_{|(a)} := \sigma^a$, and if h is along h^*, we additionally require $\sigma(h) = a^*$, where a^* is the honest choice after h. In this case follows $H_{|h}(\sigma) = h^*_{|h}$, since $H_{|(h,a^*)}(\sigma^{a^*}) = h^*_{|(h,a^*)}$ by assumption. We now pick an arbitrary strategy $\tau \in \mathcal{S}_{|h}$ and consider $a' := \tau(h) \in A(h)$. As $P(h) \in S$, for $\tau^{a'} := \tau_{|a'}$, it holds $H_{|h}(\sigma_{N-S}, \tau_S) = (a', H_{|(h,a')}(\sigma^{a'}_{N-S}, \tau^{a'}_S))$. Thus, also their utilities are identical which implies by the assumption that $\sigma^{a'}$ is collusion resilient against S, that $\sum_{p\in S} u^*_p \geq \sum_{p\in S} u_{|(h,a),p}(\sigma^a_{N-S}, \tau^a_S) = \sum_{p\in S} u_{|h,p}(\sigma_{N-S}, \tau_S)$. Hence, σ – and therefore $\Gamma_{|h}$ – is collusion resilient against S.

To show equivalence 3), we assume $h \in \mathscr{H} \setminus \mathscr{T}$ is a non-terminal history along the honest history h^* at which a not-deviating player has a turn $P(h) \notin S$. We again start by proving implication "$\Rightarrow$" and assume strategy $\sigma \in \mathcal{S}_{|h}$ yields the honest history and is collusion resilient against S in $\Gamma_{|h}$. We define $\sigma^{a^*} := \sigma_{|(a^*)} \in \mathcal{S}_{|(h,a^*)}$, which by construction also yields the honest history. For an arbitrary $\tau^{a^*} \in \mathcal{S}_{|(h,a^*)}$, let $\tau \in \mathcal{S}_{|h}$ be an extension, i.e. $\tau_{|(a^*)} := \tau^{a^*}$, rest arbitrary. Then, due to $P(h) \notin S$ and $\sigma(h) = a^*$, follows $H_{|h}(\sigma_{N-S}, \tau_S) = (a^*, H_{|(h,a^*)}(\sigma^{a^*}_{N-S}, \tau^{a^*}_S))$. As before, this implies that the collusion resilience against S of σ^* and, therefore, the collusion resilience against S of $\Gamma_{|(h,a^*)}$. For the other direction "$\Leftarrow$", we assume $\sigma^{a^*} \in \mathcal{S}_{|(h,a^*)}$ is honest and collusion resilient against S. We construct $\sigma \in \mathcal{S}_{|h}$ such that $\sigma_{|(a^*)} := \sigma^{a^*}$, $\sigma(h) = a^*$ and the rest arbitrary, with similar reasoning as before we conclude that σ yields the honest history and is collusion resilient against S.

For the last equivalence 4), let $h \in \mathscr{H} \setminus \mathscr{T}$ be a non-terminal history not along the honest history h^* at which a not-deviating player has a turn $P(h) \notin S$. The first implication "$\Rightarrow$" can be shown by assuming $\sigma \in \mathcal{S}$ is collusion resilient against S and by choosing $\sigma^a := \sigma_{|(a)} \in \mathcal{S}_{|(h,a)}$, where $a := \sigma(h)$. Fixing now an arbitrary $\tau^a \in \mathcal{S}_{|(h,a)}$, and an extension $\tau \in \mathcal{S}$ of it, since $\sigma(h) = a$ and $P(h) \notin S$, follows $H_{|h}(\sigma_{N-S}, \tau_S) = (a, H_{|(h,a)}(\sigma^a_{N-S}, \tau^a_S))$. As shown before this implies σ^a is collusion resilient against S in $\Gamma_{|(h,a)}$. The other implication "$\Leftarrow$" is similar to prove. Assume $a \in A(h)$ is such that $\sigma^a \in \mathcal{S}_{|(h,a)}$ is collusion resilient against S in $\Gamma_{|(h,a)}$. We construct a strategy $\sigma \in \mathcal{S}_{|h}$, by setting $\sigma(h) = a$ and $\sigma_{|(a)} := \sigma^a$, the rest can be arbitrary. Let now be $\tau \in \mathcal{S}_{|h}$ arbitrary. With $\tau^a := \tau_{|(a)} \in \mathcal{S}_{|(h,a)}$, we arrive at $H_{|h}(\sigma_{N-S}, \tau_S) = (a, H_{|(h,a)}(\sigma^a_{N-S}, \tau^a_S))$. This again implies that σ is collusion resilient against S in $\Gamma_{|h}$, which concludes the proof of equivalence 4) and hence the theorem. □

We need a lemma to show Theorem 5.12.

Lemma A.6. *A strategy $\sigma \in \mathcal{S}_{|h}$ is practical in subtree $\Gamma_{|h}$ iff strategy $\sigma_{|g} \in \mathcal{S}_{|(h,g)}$ is practical in subtree $\Gamma_{|(h,g)}$, for all $g \in \mathscr{H}_{|h}$.*

Proof. By Definition 3.6 and Definition 3.9, a strategy $\sigma \in \mathcal{S}_{|h}$ is practical if for all $g' \in \mathscr{H}_{|h}$ and all $\tau \in \mathcal{S}_{|(h,g')}$ the utility inequality holds. Whereas, $\sigma_{|g} \in \mathcal{S}_{|(h,g)}$ is practical for all $g \in \mathscr{H}_{|h}$, if for all $k \in \mathscr{H}_{|(h,g)}$ and for all $\tau \in \mathcal{S}_{|(h,g,k)}$ the utility inequality holds. Hence, by substituting $g' = (g, k)$ in the above-sketched formulae, it is easy to see that they are equivalent. □

Theorem 5.12 (Compositional Practicality). *Let Γ be an EFG with honest history h^* and $\mathbb{U}(h)$ be the set of practical utilities of subtree $\Gamma_{|h}$. Let u^* be the honest utility $u^* = u(h^*)$. Then the following identities and equivalences hold.*

(1) *In a leaf of* Γ *the only practical utility is that of the leaf.*

$$\forall t \in \mathscr{T}\,.\,\mathbb{U}(t) = \{u(t)\}\ .$$

(2) *The honest utility* u^* *is practical in a branch of* Γ *along* h^* *iff it is practical in the child following* h^* *and if for every other child at least one practical utility is not greater than* u^* *for the current player. Let* $a^* \in A(h)$ *be the honest action after* h*, then:*

$$\forall h \in \mathscr{H} \setminus \mathscr{T}\,.\,h \text{ along } h^* \quad \Rightarrow \quad \big(pr(\Gamma_{|h}) \quad \Leftrightarrow \\ pr(\Gamma_{|(h,a^*)}) \ \wedge\ \forall a \in A(h) \setminus \{a^*\}\ \exists u \in \mathbb{U}((h,a))\,.\,u^*_{P(h)} \geq u_{P(h)}\big)\ .$$

(3) *A utility is practical in a branch of* Γ *off the honest history* h^* *iff it is practical in a child and if, for every other child, at least one practical utility is not greater for the current player.*

$$\forall h \in \mathscr{H} \setminus \mathscr{T}\,.\,h \text{ off } h^* \quad \Rightarrow \quad \big(\forall t \in \mathscr{T}_{|h}.\,u(t) \in \mathbb{U}(h) \quad \Leftrightarrow \\ \exists a \in A(h).\,u(t) \in \mathbb{U}((h,a)) \wedge \\ \forall a' \in A(h) \setminus \{a\}\ \exists u' \in \mathbb{U}((h,a'))\,.\,u_{P(h)}(t) \geq u'_{P(h)}\big)\ .$$

Proof. We show claim 1) first. By Definitions 3.9 and 5.5, we know that the utility in the leaf after terminal history t is practical in subtree $\Gamma_{|h}$, if there is $\sigma \in \mathcal{S}_{|h}$, extending history t and $\forall g \in \mathscr{H}_{|h}\,\forall \tau \in \mathcal{S}_{|(h,g)}$

$$u_{|(h,g),P(h,g)}(\sigma_{|g}) \geq u_{|(h,g),P(h,g)}(\sigma_{|g,N-P(h,g)}, \tau_{P(h,g)})\ .$$

In our case, we have $h = t$; hence the only strategy $\sigma \in \mathcal{S}_{|t}$ is the empty one. Further, the only history $g \in \mathscr{H}_{|t}$ is the empty history (hence $(h,g) = t$), and thus also the only strategy $\tau \in \mathcal{S}_{|t}$ is the empty strategy. This leaves us with $u(t)$ being practical iff $u_{|t,P(t)}(\sigma) \geq u_{|t,P(t)}(\tau)$, which is equivalent to $u_{P(t)}(t) = u_{P(t)}(t)$. Therefore, $u(t) \in \mathbb{U}(t)$, and since there are no other utilities is $\Gamma_{|t}$, follows $\mathbb{U}(t) = \{u(t)\}$.

To show equivalence 2), we fix a history $h \in \mathscr{H} \setminus \mathscr{T}$, which is along the honest history, and let a^* be the honest action after h. For implication "$\Rightarrow$" we assume strategy $\sigma \in \mathcal{S}_{|h}$ is practical in $\Gamma_{|h}$ and yields the honest history. Theorem 5.1 implies that also $\Gamma_{|(h,a^*)}$ is practical. To show the other conjunct of the right-hand side, let $a \in A(h) \setminus \{a^*\}$ be arbitrary. We apply Lemma A.6, to get that $\sigma_{|(a)}$ is practical in $\Gamma_{|(h,a)}$. Hence, $u := u_{|(h,a)}(\sigma_{|(a)}) \in \mathbb{U}(h,a)$. Since σ is practical in $\Gamma_{|h}$, for $g = \emptyset \in \mathscr{H}_{|h}$ and $\tau \in \mathcal{S}_{|h}$ such that $\tau(h) = a$ and $\tau_{|(a)} = \sigma_{|(a)}$ follows

$$u^* = u_{|h,P(h)}(\sigma) \geq u_{|h,P(h)}(\tau_{P(h)}, \sigma_{N-P(h)}) = u_{|(h,a)}(\sigma_{|(a)})\ .$$

The last utility is exactly the considered $u \in \mathbb{U}(h,a)$, which concludes the proof of the implication.

To show the other direction "$\Leftarrow$", we assume the right-hand side holds. I.e. there exists an honest strategy $\sigma^{a^*} \in \mathcal{S}_{|(h,a^*)}$ that is practical in $\Gamma_{|(h,a^*)}$ as well as strategies $\sigma^a \in \mathcal{S}_{|(h,a)}$ for $a \neq a^*$, that are practical in $\Gamma_{|(h,a)}$ and whose utilities $u^a \in \mathbb{U}(h,a)$ satisfy $u^*_{P(h)} \geq u^a_{P(h)}$. We now construct a strategy $\sigma \in \mathcal{S}_{|h}$ in the following way: Let $\sigma(h) = a^*$ and for all $a \in A(h)$ (including a^*) let $\sigma_{|(a)} = \sigma^a$. First, we note that σ yields the honest history by construction. Second, we prove it is practical in $\Gamma_{|h}$ by picking an arbitrary history $g \in \mathscr{H}_{|h}$ and an arbitrary strategy $\tau \in \mathcal{S}_{|(h,g)}$ and showing $u_{|(h,g),P(h,g)}(\sigma_{|g}) \geq u_{|(h,g),P(h,g)}(\tau_{P(h,g)}, \sigma_{|g,N-P(h,g)})$:

Case 1: $g = \emptyset$. Then $u_{|(h),P(h)}(\sigma) = u^*_{P(h)}$, as σ yields the honest history and h along h^*. Also, $u_{|(h),P(h)}(\tau_{P(h)}, \sigma_{N-P(h)}) = u_{|(h,a),P(h)}(\tau_{|(a),P(h)}, \sigma_{|(a),N-P(h)})$, where $a = \tau(h)$. Using Theorem 5.6 and the fact that $\sigma_{|(a)}$ is practical in $\Gamma_{|(h,a)}$, follows $u_{|(h,a),P(h)}(\tau_{|(a),P(h)}, \sigma_{|(a),N-P(h)}) \leq u_{|(h,a),P(h)}(\sigma_{|(a)}) = u^a_{P(h)}$. Applying our assumption $u^*_{P(h)} \geq u^a_{P(h)}$, the required inequality holds.
Case 2: $g = (a, g')$, where $a \in A(h)$, $g' \in \mathscr{H}_{|(h,a)}$. Then, $u_{|(h,g),P(h,g)}(\sigma_{|g}) = u_{|(h,g),P(h,g)}(\sigma^a_{|g'})$ which

is – due to the practicality of σ^a – greater than or equal to

$$\begin{aligned} & u_{|(h,g),P(h,g)}(\tau_{P(h,g)}, \sigma^a_{|g',N-P(h,g)}) \\ &= u_{|(h,a,g'),P(h,a,g')}(\tau_{P(h,a,g')}, \sigma^a_{|g',N-P(h,a,g')}) \\ &= u_{|(h,g),P(h,g)}(\tau_{P(h,g)}, \sigma_{|g,N-P(h,g)}). \end{aligned}$$

Hence, the second direction of the second claim is proven.

For equivalence 3), let $h \in \mathscr{H} \setminus \mathscr{T}$ be off the honest history h^* and $t \in \mathscr{T}_{|h}$. We prove implication "$\Rightarrow$" first and assume $u_{|h}(t) \in \mathbb{U}(h)$ is a practical utility in $\Gamma_{|h}$. Thus, there exists a strategy $\sigma \in \mathcal{S}_{|h}$ such that $H(\sigma) = t$ and σ is practical. History t is terminal, while h is not. Therefore, t is not empty. Hence, we can split it into $t := (a^t, t')$, where a^t is the first action along t. Applying Lemma A.6, we know that also $\sigma_{(a^t)} \in \mathcal{S}_{|(h,a^t)}$ is practical. Since $H(\sigma) = t$, follows $\sigma(t') = a^t$, and thus $u(t) = u_{|(h,a^t)}(\sigma_{|(a^t)}) \in \mathbb{U}(h, a^t)$ which proves the first conjunct of the implication. For the second, pick an arbitrary $a \in A(h) \setminus \{a^t\}$ and consider the strategy $\sigma \in \mathcal{S}_{|h}$ from before. Using Lemma A.6 again, it follows that also $\sigma_{|(a)} \in \mathcal{S}_{|(h,a)}$ is practical and by definition of $\mathbb{U}$ we know $u_{|(h,a)}(\sigma_{|(a)}) \in \mathbb{U}(h, a)$. Towards using the practicality of σ in $\Gamma_{|h}$, let $\tau \in \mathcal{S}_{|h}$ be such that $\tau(h) = a$ and $\tau_{|(a)} = \sigma_{|(a)}$. Finally, we conclude $u_{|h,P(h)}(t) = u_{|h,P(h)}(\sigma) \geq u_{|h,P(h)}(\tau_{P(h)}, \sigma_{N-P(h)}) = u_{|(h,a),P(h)}(\sigma_{|(a)})$. Hence also, the second conjunct holds, and the implication is proven.

The other direction, "$\Leftarrow$" of equivalence 3), can be shown similarly to "$\Leftarrow$" of equivalence 2). We assume $u_{|h}(t) \in \mathbb{U}(h, a')$ and for all other actions $a \in A(h) \setminus \{a'\}$ exists a practical utility $u^a \in \mathbb{U}(h, a)$ such that $u_{|h,P(h)}(t) \geq u^a_{P(h)}$. Then, we construct a strategy $\sigma \in \mathcal{S}_{|h}$ in the following way. Let $\sigma(h) = a'$. For all $a \in A(h) \setminus \{a'\}$, let $\sigma_{|(a)} = \sigma^a$, where σ^a is a practical strategy in $\Gamma_{|(h,a)}$ with practical utility $u^a = u_{|(h,a)}(\sigma^a) \in \mathbb{U}(h, a)$. For a', let $\sigma_{(a')}$ be the practical strategy in $\Gamma_{|(h,a')}$ that yields utility $u_{|h}(t)$. It follows that strategy σ also yields utility $u_{|h}(t)$. With this construction, we can proceed as in equivalence 2), direction "$\Leftarrow$", to prove the practicality of σ which implies the practicality of $u_{|h}(t)$ in $\Gamma_{|h}$ to conclude the proof of the theorem. □

B Algorithms

Function `ComputeWERI`. The function for weaker immunity is identical to `ComputeWI` in Algorithm 2, except that `GetUtility` returns only the real part of the requested utility.

Function `ComputeCR`. The function in Algorithm 4 is similar to `ComputeWI` in Algorithm 2. It differs in the considered utility comparison $ut_1[\vec{x}] \geq ut_2[\vec{x}]$. While ut_1 was player pg's utility and $ut_2 = 0$, for collusion resilience, ut_1 is the sum of the honest utilities of the deviating players pg and ut_2 is the sum of the utilities of the deviating players $p \in$ pg in the current leaf. Hence pg is a group of players rather than a single player. Other than that, in the branch-case (lines 10–32) the roles of whether the current player `CurrentPlayer`(Γ) is in pg or not, are reversed, as it is in Theorem 5.10.

Function `ComputePR`. The function `ComputePR` to compute the practicality of game tree Γ for honest history h^*, and in the case specified in set S operates a little differently than the ones for the other properties. As described in Theorem 5.12, we have to keep track of all practical utilities of subtrees, in order to decide the practicality of h^*.

Hence, for a leaf trivially the only practical utility is its utility (lines 1–3). For branches, we first check if any results of the subtrees yields an empty list of practical utilities in lines 4–9. If yes, then either we need a case split (if split $\neq$ null) or the honest history was not practical in a subtree which implies that it is not practical in the entire tree (Theorem 5.1), otherwise. In any case returning the respective `ComputePR` result does precisely that.

If none of the subtrees led to an empty set of practical utilities, we distinguish between whether the current subtree Γ is along the honest history h^* (`AlongHonest`(Γ, h^*)). If so, in lines 10–19 it is analyzed whether the honest utility u^* dominates at least one practical utility for each sibling,

Algorithm 3: Relay Function `ComputeSP`.

input : input instance Π, honest history h^*, security property $sp \in \{wi, weri, cr, pr\}$, set S containing initial constraints and currently analyzed case, player group pg.
output : (result, split), where result states whether Π satisfies sp for pg, given S, and split a crucial utility comparison we cannot decide.

```
1  if sp = wi then
2  |  return ComputeWI(Γ, h*, S, pg)
3  else if sp = weri then
4  |  return ComputeWERI(Γ, h*, S, pg)
5  else if sp = cr then
6  |  return ComputeCR(Γ, h*, S, pg)
7  else
8  |  (result, split) ← ComputePR(Γ, h*, S)
9  |  if result = ∅ then
10 |  |  return (false, split)
11 |  else
12 |  |  return (true, split)
13 |  end
14 end
```

thereby proceeding as in Theorem 5.12. In case we found a dominated utility per sibling, the honest utility is returned (line 18), otherwise the empty list of utilities, together with a crucial case split (if it exists) is returned (line 15).

The function `ExistsDominated`, as defined in Algorithm 6, computes whether one of the utilities in $\mathbb{U}$ has to be less than or equal to u for the current player p for all values of $\vec{x}$ that satisfy the preconditions in S. If the relation of two terms cannot be decided, we store it in split.

For the case where Γ is not along h^* (lines 20–35), the list of all practical utilities has to be computed. Similarly, as before, a utility that was practical in a subtree is practical now if it dominates at least one practical utility of each sibling. To decide domination, again function `ExistsDominated` is employed. If a further case distinction is needed, the empty list together with that split is returned (line 26); otherwise, the list of practical utilities together with the null-split (line 35).

C Soundness and Completeness Proofs

To prove Theorem 6.4, we need some preliminary results first. We start with a result about the practicality of subtrees which are off the honest history.

Lemma C.1 (Subtrees off Honest History are Practical). *Every subtree $\Gamma_{|h}$ of a game Γ that is off the honest history is* practical.

Proof. We fix an arbitrary subtree $\Gamma_{|h}$ not along the honest history for which we prove practicality. We construct a strategy $\sigma \in \mathcal{S}_{|h}$ in the following inductive way bottom-up: We consider histories k in $\Gamma_{|h}$, such that for all possible actions $a \in A_{|h}(k)$ after k the strategy σ has been defined already for the subtree after (k, a). Initially, this is only the case for histories k, where all actions $a \in A_{|h}(k)$ lead to leaves.

At each such history k we compare the utilities for the current player $P(k)$ after the possible choices $a \in A_{|h}(k)$ according to σ: $u_{|(h,k,a),P(k)}(\sigma_{|(k,a)})$. We define strategy σ to pick an action

Algorithm 4: Function `ComputeCR` for Collusion Resilience.

input : a game tree Γ, an honest history h^*, the set S containing the initial constraints and the current case, and the player group pg.

output: (result, split), where result is true iff Γ is collusion resilient against pg, given S, and split a crucial utility comparison we cannot decide.

```
if isLeaf(Γ) then
    if Check(S, GetUtility(h*, pg) < GetUtility(Γ, pg)) = unsat then
        return (true, null)
    end
    if Check(S, GetUtility(h*, pg) ≥ GetUtility(Γ, pg)) = unsat then
        return (false, null)
    end
    return (false, GetUtility(h*, pg) ≥ GetUtility(Γ, pg)))
end
if CurrentPlayer(Γ) ∈ pg then
    for a ∈ Actions(Γ) do
        (result, split) ← ComputeCR(Γ_|(a), h*, S, pg)
        if result = false then
            return (result, split)
        end
    end
    return (true, null)
end
if AlongHonest(Γ, h*) then
    a* ← HonestAction(Γ, h*)
    return ComputeCR(Γ_|(a*), h*, S, pg)
end
newsplit ← null
for a ∈ Actions(Γ) do
    (result, split) ← ComputeCR(Γ_|(a), h*, S, pg)
    if result = true then
        return (true, null)
    else if split ≠ null then
        newsplit ← split
    end
end
return (false, newsplit)
```

$a' = \sigma(k) \in A_{|h}(k)$ which maximizes $P(k)$'s utility. If this is done iteratively, one eventually reaches the root of $\Gamma_{|h}$, at which point the strategy $\sigma \in \mathcal{S}_{|h}$ is fully defined.

We now show σ is practical for the fixed case split. According to Definition 3.6, we pick an arbitrary history $g \in \mathcal{H}_{|h}$, a player $p \in N$ and a strategy $\tau \in \mathcal{S}_{|(h,g)}$ in the subgame after g, and

Algorithm 5: Function `ComputePR` for Practicality.

input : a game tree Γ, honest history h^*, set S containing the initial constraints and the current case.

output: $(\mathbb{U}_\Gamma, \text{split})$, where $\mathbb{U}_\Gamma$ are all practical (and honest if along h^*) utilities in Γ, and split a crucial utility comparison that cannot be decided.

```
1  if isLeaf(Γ) then
2  |  return ([GetUtility(Γ)], null)
3  end
4  subtrees ← [ComputePR(Γ_|(a), h*, S) for a ∈ Actions(Γ)]
5  p ← CurrentPlayer(Γ)
6  U_Γ ← ∅
7  if AnyUtilityEmpty(subtrees) then
8  |  return subtrees.ReturnEmpty()
9  end
10 if AlongHonest(Γ, h*) then
11 |  ([u*], split*) ← GetHonestResult(subtrees)
12 |  for (U^a, _) ∈ subtrees \ {([u*], split*)} do
13 |  |  (result, split) ← ExistsDominated(U^a, u*, S, p)
14 |  |  if ¬result then
15 |  |  |  return (∅, split)
16 |  |  end
17 |  end
18 |  return ([u*], null)
19 end
20 for (U^a, split^a) ∈ subtrees do
21 |  for u ∈ U^a do
22 |  |  toAdd ← true
23 |  |  for (U^s, _) ∈ subtrees \ (U^a, split^a) do
24 |  |  |  (result, split) ← ExistsDominated(U^s, u, S, p)
25 |  |  |  if ¬result ∧ split ≠ null then
26 |  |  |  |  return (∅, split)
27 |  |  |  else if ¬result ∧ split = null then
28 |  |  |  |  toAdd ← false
29 |  |  |  end
30 |  |  end
31 |  |  if toAdd then
32 |  |  |  U_Γ.Add(u)
33 |  end
34 end
35 return (U_Γ, null)
```

Algorithm 6: Function `ExistsDominated`.

input : a list of utility tuples $\mathbb{U}$, a single utility tuple u, set S containing the initial constraints and the current case, a player p.

output: (result, split), where result is `true` if there exists a utility $u' \in \mathbb{U}$ such that $u_p \geq u'_p$ for all values that satisfy S; `false` otherwise; and split a crucial utility comparison that cannot be decided.

```
existsDominated ← false
split ← null
for u' ∈ 𝕌 do
    if Check(S, u[p] < u'[p]) = unsat then
        existsDominated ← true
    else if Check(S, u[p] ≥ u'[p]) = sat then
        split ← u[p] ≥ u'[p]
    end
end
return (existsDominated, split)
```

prove:

$$u_{|(h,g),p}(\sigma_{|g}) \geq u_{|(h,g),p}(\tau_p, \sigma_{|g,N-p}) \ . \tag{8}$$

If τ_p and $\sigma_{|g,p}$ are identical, the inequality holds trivially, hence we assume them to be distinct. The strategy $\sigma_{|g}$ generates a history which we call $h_\sigma = H(\sigma_{|g})$ and $H(\tau_p, \sigma_{|g,N-p}) = h_\tau$. Note that the strategies differ only in choices player p made. We will now use induction on the deviation points ℓ_n of σ along h_τ to show Equation (8). Our induction hypothesis is

$$u_{|(h,g,\ell_n),p}(\sigma_{|(g,\ell_n)}) \geq u_{|(h,g,\ell_n),p}(\tau_{|\ell_n,p}, \sigma_{|(g,\ell_n),N-p}) \ . \tag{9}$$

For the base case, we consider the last point along the generated history h_τ, where the choice taken in strategy $(\tau_p, \sigma_{|g,N-p})$ differs from the one taken in $\sigma_{|g}$. We call the history leading to this point $\ell_1 \in \mathscr{H}_{|(h,g)}$. The player at ℓ_1 has to be p. Revisiting now the construction of σ, at ℓ_1 we chose $\sigma_{|g}(\ell_1) =: a_1$ to maximize the utility of $P(\ell_1) = p$. Therefore, we know for $c_1 := \tau(\ell_1)$

$$\begin{aligned} u_{|(h,g,\ell_1,a_1),p}(\sigma_{|(g,\ell_1,a_1)}) &\geq u_{|(h,g,\ell_1,c_1),p}(\sigma_{|(g,\ell_1,c_1)}) \\ &= u_{|(h,g,\ell_1,c_1),p}(\tau_{|(\ell_1,c_1),p}, \sigma_{|(g,\ell_1,c_1),N-p}). \end{aligned}$$

The equality holds, because σ and (τ_p, σ_{N-p}) are identical on $\Gamma_{|(h,g,\ell_1,c_1)}$. By definition of a_1 and c_1 we also know

$$u_{|(h,g,\ell_1),p}(\sigma_{|(g,\ell_1)}) \geq u_{|(h,g,\ell_1),p}(\tau_{|\ell_1,p}, \sigma_{|(g,\ell_1),N-p}) \ , \tag{10}$$

which concludes the base case.

For the inductive case, we assume the induction hypothesis Equation (9), for the previous deviation point ℓ_{n-1} and consider deviation point ℓ_n along h_τ. We define $c_n := \tau(\ell_n)$ an $a_n := \sigma_{|g}(\ell_n)$. By the definition of σ, we have

$$u_{|(h,g,\ell_n,a_n),p}(\sigma_{|(g,\ell_n,a_n)}) \geq u_{|(h,g,\ell_n,c_n),p}(\sigma_{|(g,\ell_n,c_n)}) \ . \tag{11}$$

We further know that

$$u_{|(h,g,\ell_n,c_n),p}(\sigma_{|(g,\ell_n,c_n)}) = u_{|(h,g,\ell_{n-1}),p}(\sigma_{|(g,\ell_{n-1})}) \ . \tag{12}$$

This is the case, since by definition of ℓ_n and ℓ_{n-1} as subsequent deviation points, the history of $\sigma_{|g}$ between (ℓ_n, c_n) and ℓ_{n-1} is identical to the one of $(\tau_p, \sigma)_{|g,N-p}$. Combining now Equation (11), Equation (12) and the inductive hypothesis we derive

$$u_{|(h,g,\ell_n,a_n),p}(\sigma_{|(g,\ell_n,a_n)}) \geq u_{|(h,g,\ell_{n-1}),p}(\tau_{|l_{n-1},p}, \sigma_{|(g,\ell_{n-1}),N-p}) \ . \tag{13}$$

By the definition of a_n (for the left-hand side of the equation) and the fact that ℓ_n and ℓ_{n-1} are along h_τ (for the right-hand side), we conclude that Equation (9) holds also for ℓ_n.

To finalize the proof, we consider the first point ℓ_M along h_τ, where σ_g and $(\tau_p, \sigma_{|g,N-p})$ differ. This is exactly the splitting point of h_σ and h_τ, thus

$$u_{|(h,g),p}(\sigma_{|g}) = u_{|(h,g,\ell_M),p}(\sigma_{|(g,\ell_M)}) \ , \text{and} \tag{14}$$

$$u_{|(h,g),p}(\tau_p, \sigma_{|g,N-p}) = u_{|(h,g,\ell_M),p}(\tau_{|l_M,p}, \sigma_{|(g,\ell_M),N-p}) \ . \tag{15}$$

From this, together with Equation (9), Equation (8) follows, which shows that σ is indeed practical. Hence, we proved that any subtree not along the honest history is practical. □

Next, two lemmas about the `ComputePR` function are stated.

Lemma C.2. *Let Γ be a subtree of a game Γ', h^* an honest history of the Γ', S be a set of initial constraints C and case* `case`*, and* $\texttt{ComputePR}(\Gamma, h^*, \mathsf{S}) = (\mathbb{U}_\Gamma, \mathsf{split})$*. If a utility u is an element of $\mathbb{U}_\Gamma$, then for all values $\vec{x}$ satisfying $C \cup \texttt{case}\ u[\vec{x}]$ is practical in Γ, and – if Γ is along h^* – $u = u(h^*)$.*

Proof. We prove this claim using structural induction. For the base case, we assume Γ is a leaf. Then according to Algorithm 5, lines 1–3, the utility of the leaf will be returned. It is by definition also the only practical utility of the leaf (for all values of $\vec{x}$). If the leaf is along the honest history, it is further the honest utility, hence the base case is shown.

Let us now assume Γ is a branch and that we have shown the property for all subtrees of Γ. We first consider the case where Γ is along h^*. Then, by induction hypothesis and assuming a^* is the honest action, if $\mathbb{U}_{\Gamma_{|(a^*)}}$ contains an element u_{a^*} it has to be the honest utility $u_{a^*} = u(h^*)$ and it has to be practical for all values of $\vec{x}$ that satisfy $C \cup \texttt{case}$. Hence, $u^* = u(h^*)$ in line 11. Following lines 12–17 of the algorithm and Algorithm 6, $u(h^*)$ is returned iff for all siblings Γ_a of $\Gamma_{(a^*)}$ exists a (by induction hypothesis practical) utility u_a such that the (by construction satisfiable) constraints in S together with the constraint $u_p(h^*) < u_{a,p}$ is unsatisfiable, where p is the current player at Γ. This equivalent to all $\vec{x}$ that satisfy $C \cup \texttt{case}$ also satisfy $u_p(h^*)[\vec{x}] \geq u_{a,p}[\vec{x}]$. Applying now Theorem 5.12, it follows that $u(h^*)[x]$ is practical in Γ for all $\vec{x}$ that satisfy $C \cup \texttt{case}$ in this exact case. Hence, if $u \in \mathbb{U}_\Gamma$, then $u = u(h^*)$ and it is practical for all $\vec{x}$ that satisfy $C \cup \texttt{case}$.

Secondly, assume Γ is not along the honest history. Again, by induction hypothesis all utilities occurring in line 4 subtrees, are practical in their subgames for all $\vec{x}$ satisfying $C \cup \texttt{case}$. A utility $u_a \in \mathbb{U}_{\Gamma_{|(a)}}$ is now added to the returned set $\mathbb{U}_\Gamma$, exactly if for all siblings $\Gamma_{|b}$ there exists a (by induction hypothesis practical in $\Gamma_{|b}$ for the $\vec{x}$ satisfying the constraints) utility u_b such that $C \cup \texttt{case} \cup u_{a,p} < u_{b,p}$ is unsat. Which is again according to Theorem 5.12 equivalent to u_a being practical in Γ for all $\vec{x}$ satisfying $C \cup \texttt{case}$. This concludes the induction step and hence the proof of the lemma. □

Lemma C.3. *Let Γ be a subtree of a game Γ', h^* an honest history of Γ', S a set of initial constraints C and a case* `case`*, and* $\texttt{ComputePR}(\Gamma, h^*, \mathsf{S}) = (\mathbb{U}_\Gamma, \mathsf{null})$*. If a utility u from Γ is not an element of $\mathbb{U}_\Gamma$, then for all values $\vec{x}$ satisfying $C \cup \texttt{case}$ the utility $u[\vec{x}]$ is not practical in Γ, or – if Γ is along h^* – $u \neq u(h^*)$.*

Proof. We prove the lemma by structural induction. For the base case we assume Γ is a leaf. If a utility is not in $\mathbb{U}_\Gamma$, that means that it is not the leaf utility which is equivalent to not being practical in Γ.

For the inductive case, we assume Γ is a branch. First, we further assume that Γ is along h^*. By induction hypothesis, Lemma C.2 and by Algorithm 5, the only candidate for being in $\mathbb{U}_\Gamma$ is $u(h^*)$, if it was in $\mathbb{U}_{\Gamma_{|(a^*)}}$. All others are by the algorithm not in $\mathbb{U}_{\Gamma_{a^*}}$ and by induction hypothesis therefore either not $u(h^*)$ or are $u(h^*)$ but not practical in $\Gamma_{|(a^*)}$. In any case, all of those are also in Γ not $u(h^*)$ or are $u(h^*)$ but not practical in Γ, according to Theorem 5.12. Therefore, assume $u(h^*) \in \mathbb{U}_{\Gamma_{|(a^*)}}$, but is not in $\mathbb{U}_{|\Gamma}$. Following the algorithm, this implies that there is a child $\Gamma_{|(a)}$, for which all practical utilities $u \in \mathbb{U}_{\Gamma_{|(a)}}$ (by induction hypothesis and Lemma C.2) the constraints in S imply $u_p(h^*) < u_p$, where p is the current player. This has to be the case because otherwise either the returned split would not have been null or $u(h^*) \in \mathbb{U}_\Gamma$. It, however, implies that $u(h^*)$ is not practical for all $\vec{x}$ satisfying the constraints in S.

Finally, we assume Γ is not along the honest history. We know that only utilities that occurred in a $\mathbb{U}_{\Gamma_{|(a)}}$ are candidates to be in $\mathbb{U}_\Gamma$, the others not. The others are, according to the induction hypothesis, not practical in $\mathbb{U}_{\Gamma_{|(a)}}$ for all $\vec{x}$ satisfying the constraints in S. Therefore they are also not practical in $\mathbb{U}_\Gamma$ for all $\vec{x}$ satisfying the constraints in S. The ones that are in a $\mathbb{U}_{\Gamma_{|(a)}}$ are practical for all such $\vec{x}$ by Lemma C.2. Hence, if a utility $u \in \mathbb{U}_{\Gamma_{|(a)}}$ but not in $\mathbb{U}_\Gamma$ according to the algorithm, there has to exist a sibling $\Gamma_{|(a')}$ such that for all their practical utilities $u_{a'}$, the constraints of S together with $u_p \geq u_{a',p}$ are unsatisfiable. Otherwise, either the split would not be null, or u would have been added to $\mathbb{U}_\Gamma$. By Theorem 5.12, u is not practical in Γ for all $\vec{x}$ that satisfy the constraints in S. □

The next lemma reasons about the negative output of `ComputeSP`.

Lemma C.4. *Given an input instance* Π*, an honest history* h^**, a security property* sp *and* S *the set of initial constraints* C *and a case* `case`*, if there exists a player group* pg *such that*

$$\texttt{ComputeSP}(\Pi, h^*, \mathsf{S}, sp, \mathsf{pg}) = (\texttt{false}, \mathsf{null}) \ ,$$

then

$$\forall \vec{x}.\ \forall c \in C \cup \texttt{case}.\ c[\vec{x}] \longrightarrow \neg sp(\Gamma, h^*)[\vec{x}] \ .$$

Proof. We prove the lemma per security property algorithm. First, we consider security properties weak and weaker immunity: There exists a player group such that `ComputeSP` returns $(\texttt{false}, \mathsf{null})$, iff `ComputeWI` returns $(\texttt{false}, \mathsf{null})$ for pg. We proceed by structural induction. For the base case, we assume Γ is a leaf. Following Algorithm 2, one can see in lines 1–9 that $(\texttt{false}, \mathsf{null})$ is returned if $C \cup \texttt{case} \cup u_{\mathsf{pg}} < 0$ is satisfiable but $C \cup \texttt{case} \cup u_{\mathsf{pg}} \geq 0$ is unsatisfiable, where u is the utility of the leaf respectively its real part for weaker immunity. This implies that for all $\vec{x}$ that satisfy $C \cup \texttt{case}$ the inequality $u_{\mathsf{pg}} < 0$ holds. Therefore, $\neg sp(\Gamma, h^*)$ for all such $\vec{x}$.

For the induction step we assume Γ is a branch and by induction hypothesis all subtrees of Γ satisfy the property. For the first case we assume pg is not the current player in Γ. If $(\texttt{false}, \mathsf{null})$ is returned from `ComputeWI`, there had to be a child $\Gamma_{|(a)}$ of Γ such that `ComputeWI` returned `false` in line 14. Applying the induction hypothesis this implies that for all $\vec{x}$ satisfying the constraints in S $\Gamma_{|(a)}$ is not weak(er) immune for pg. Let us now fix such an $\vec{x}$ arbitrarily. Following Theorem 5.8, we know that for values $\vec{x}$ the game Γ is not weak(er) immune for pg. Since $\vec{x}$ was chosen arbitrarily, Γ is not weak(er) immune for pg for all $\vec{x}$ that satisfy the constraints in S.

Secondly, assume pg is the current player and Γ is along h^* and `ComputeWI` returned $(\texttt{false}, \mathsf{null})$. Following Algorithm 2 lines 19–21, this can only happen, if $(\texttt{false}, \mathsf{null})$ was returned in line 21. By induction hypothesis this implies that for all $\vec{x}$ satisfying the constraints in S the subgame $\Gamma_{|(a^*)}$ is not weak(er) immune for pg. By Theorem 5.8, it follows that also Γ is not weak(er) immune for pg, for all $\vec{x}$ satisfying the constraints in S.

Lastly, we assume pg is the current player and Γ is not along h^* and `ComputeWI` returned $(\texttt{false}, \mathsf{null})$. According to the algorithm, that implies that all children had returned $(\texttt{false}, \mathsf{null})$ in line 25. By induction hypothesis, this means that all children are not weak(er) immune for pg for all $\vec{x}$ satisfying the constraints in S. Applying Theorem 5.8, it follows that also Γ is not weak(er) immune for pg for all $\vec{x}$ satisfying $C \cup \texttt{case}$.

The property for collusion resilience holds due to the same reasoning as for weak(er) immunity. For practicality, note that $\texttt{ComputeSP}(\Pi, h^*, \mathsf{S}, pr, \mathsf{pg}) = (\texttt{false}, \mathsf{null})$ iff $\texttt{ComputePR}(\Gamma, h^*, \mathsf{S})$ returns $(\emptyset, \mathsf{null})$.

According to Lemma C.3 $\texttt{ComputePR}(\Gamma, h^*, \mathsf{S}) = (\emptyset, \mathsf{null})$ implies that first Γ has to be along h^* (as otherwise no utility would be practical in Γ for an arbitrary $\vec{x}$ satisfying the constraints in S, which was proven impossible in Lemma C.1) and second that the honest history is not practical in Γ for all $\vec{x}$ that satisfy the constraints in S. This concludes the proof for practicality and, thus, the lemma. □

The last lemma we need to prove the theorem gives insight into the positive output of the `ComputeSP` function:

Lemma C.5. *Given an input instance* Π*, an honest history* h^**, a security property* sp *and a set* S *of initial constraints* C *and a case* case*, if for all player groups* pg

$$\texttt{ComputeSP}(\Pi, h^*, \mathsf{S}, sp, \mathsf{pg}) = (\texttt{true}, \mathsf{split}) \ ,$$

independent of what split *is, then*

$$\forall \vec{x}.\ \forall c \in C \cup \texttt{case}.\ c[\vec{x}] \rightarrow sp(\Gamma, h^*)[\vec{x}] \ .$$

Proof. Let us fix a player group pg and assume sp is not pr. We again prove the lemma by structural induction on the game tree Γ. For the base case we assume Γ is a leaf and the return value is $(\texttt{true}, \mathsf{split})$. According to Algorithms 2 and 4, this can only happen if $C \cup \texttt{case}$ together with the negated property inequality of the leaf utility u ($u_{\mathsf{pg}} < 0$, respectively $u_{\mathsf{pg}}(h^*) < u_{\mathsf{pg}}$) is unsat. This implies according to Theorems 5.8 and 5.10, that for all $\vec{x}$ satisfying $C \cup \texttt{case}$ the game Γ does not satisfy the security property for/against pg. The induction step is analog to the one of Lemma C.4.

For $sp = pr$, we employ Lemma C.2 to conclude that Γ with honest history h^* is practical for all $\vec{x}$ that satisfy the constraints in S.

We showed that for all $\vec{x}$ that satisfy the constraints in S, Γ with honest history h^* satisfies sp for/against pg. Since pg was chosen arbitrarily, it holds for all player groups. We can further separate the for-all quantification into both sides of the implication. As the player group quantification and the $\vec{x}$ quantification are independent, their order can be switched. By Theorem 5.6, it finally follows that for all $\vec{x}$ that satisfy the constraints in S, the $sp(\Gamma, h^*)$ holds. □

Theorem 6.4 (Correctness of Algorithm 1). *The compositional approach to compute the game-theoretic security of an input instance* Π *for honest history* h^* *described in Algorithm 1 is sound and complete. That is,* $\texttt{SatisfiesProperty}(\Pi, h^*, sp, \emptyset) = \texttt{true}$ *iff* Π *with honest history* h^* *satisfies the property* sp*. Otherwise, it returns* `false`.

Proof. We prove direction "$\Leftarrow$" by contraposition: we assume $\texttt{SatisfiesProperty}(\Pi, h^*, sp, \emptyset)$ returns `false` and show that Π with honest history h^* does not satisfy the security property sp. From our assumption and Algorithm 1, we know that for the return value to be `false`, there has to exist a set of utility term comparisons case such that

$$\texttt{SatisfiesProperty}(\Pi, h^*, sp, \texttt{case}) = \texttt{false} \ .$$

This implies that there has to exist a player group pg such that function $\texttt{ComputeSP}(\Pi, h^*, \text{S}, sp, \text{pg})$ returns $(\texttt{false}, \text{null})$, where S contains the constraints from C, (defined in Π) and case.

We can now apply Lemma C.4 to conclude that for all values of $\vec{x}$ that satisfy the (satisfiable) set of constraints in S, the game Γ (of Π) with honest history h^* violates security property sp for player group pg. The constraint set case can be extended to a total order $\preceq$ on the utility terms T_u. Thus, by the fact the all $\vec{x}$ that satisfy $C \cup \preceq$ also satisfy $C \cup \texttt{case}$, it follows that for all $\vec{x}$ that satisfy $C \cup \preceq$ security property sp does not hold for player group pg. Using Theorem 3.11, this means Equation (1) does not hold. That means exactly $\neg sp(\Pi, h^*)$. Hence, the entire input instance Π violates security property sp in general.

For the other direction "$\Rightarrow$", we first assume that the function $\texttt{SatisfiesProperty}(\Pi, h^*, sp, \emptyset)$ returns true. Following then the Algorithm 1, this implies that all final cases returned true. Further, all considered cases are pair-wise disjoint, and their disjunction is a tautology. Considering one such case case, it has to be the case that for all player groups pg

$$\texttt{ComputeSP}(\Pi, h^*, \text{S}, sp, \text{pg}) = (\texttt{true}, \text{split}) \ ,$$

where the value of split is irrelevant and S contains the constraints of C and case. According to Lemma C.5, this implies that for all $\vec{x}$ that satisfy C and case that security property sp holds for player group pg, game Γ and honest history h^*. As before, this implies that for all total orders $\preceq$ extending case, the security property holds for pg. Further, the disjunction of all total orders that extend case is equivalent to case itself. Since this result holds for all the considered cases and those cases span the considered universe, it follows that for all total orderes $\preceq$ holds that all $\vec{x}$ that satisfy $C \cup \preceq$ satisfy the security property for all player groups. According to Theorem 3.11, follows Equation (1) holds. Hence, the input instance Π with honest history h^* satisfies the security property.

Finally, we have to show that SatisfiesProperty always returns either true or false. This is equivalent to proving termination. The Compute<SP> functions terminate since they walk through the finite game tree once, and all SMT queries are decidable, as unquantified non-linear real arithmetic is decidable. Further, the function SatisfiesProperty splits only on utility comparisons of players/groups of players, which are also finitely many. Hence the algorithm terminates. □

D Extracting Strategies and Finding Counterexamples

Theorem 6.5 (Weak(er) Immune Strategies). *For a weak(er) immune game Γ, with honest history h^* and total order $\preceq$, strategy σ is honest and weak(er) immune for all $\vec{x}$ satisfying $\preceq$, where*

$$\sigma := (\sigma^{p_1}, \ldots, \sigma^{p_{|N|}}) \ ,$$

and $\sigma^{p_i} \in \mathcal{S}_{p_i}$ is a strategy for player p_i. Strategy σ^{p_i} picks the honest choice along the honest history, whereas at other nodes, where it is p_i's turn, it picks an arbitrary action a that yields a weak(er) immune for p_i subtree after action a.

Proof. Consider $\sigma^{p_i} \in \mathcal{S}_{p_i}$ as in the theorem. This strategy is weak(er) immune for p_i. To show this, consider an arbitrary joint strategy τ. Then $u_{p_i}(\tau_{N-p_i}, \sigma^{p_i}) \geq 0$ (respectively its real part), since along the generated history of $(\tau_{N-p_i}, \sigma^{p_i})$ either it is not player p_i's turn, in which case according to Theorem 5.8, all choices have to be weak(er) immune for p_i, or it is player p_i's turn, in which case we chose a weak(er) immune for p_i action for σ^{p_i}. Eventually, we thus have to reach a weak(er) immune for p_i leaf. A leaf is weak(er) immune for p_i, iff its (real part of the) utility for p_i is non-negative for all $\vec{x}$ satisfying $\preceq$.

Applying now the proof of Theorem 5.6 for weak(er) immunity, it follows that strategy σ is weak(er) immune for all $\vec{x}$ satisfying $\preceq$. □

For collusion resilience it is also possible to compute an honest collusion resilient strategy compositionally. However, proving that the constructed strategy is collusion resilient requires more involved reasoning.

Theorem D.2 (Collusion Resilient Strategies). *For a collusion resilient game Γ, with honest history h^* and total order $\preceq$, strategy σ is honest and collusion resilient for all $\vec{x}$ satisfying $\preceq$, where we proceed top-down and breath-first, to pick the following action for strategy σ at history $h \in \mathscr{H} \setminus \mathscr{T}$:*

(1) *if h is along h^*, pick the honest action a^*: $\sigma(h) = a^*$;*
(2) *otherwise, maintain the set of players $S \subseteq N$ that had to deviate from σ to reach h. Then pick an arbitrary action a, for which the subtree $\Gamma_{|(h,a)}$ is collusion resilient against all supersets of S other than N.*

Note that in case (2) of Theorem D.2, there always exists an action a, such that the subtree $\Gamma_{|(h,a)}$ is collusion resilient against all supersets of S other than N. Theorem D.2 is also constructive, yielding an algoritmic way to compute a collusion resilient strategy. We can proceed in the same way as for weak immunity in Theorem 6.5: during analysis for each player group pg and each branch, we store whether the branch is collusion resilient against pg. If the game turns out to be collusion resilient, we can collect the choices for σ according to the theorem.

Proof. First, we have to prove that such a strategy always exists, provided that h^* is collusion resilient. We fix an arbitrary total order $\preceq$ and consider only values for $\vec{x}$ that satisfy $\preceq$. Towards a contradiction, we assume we cannot pick an action according to the theorem at history h. We further assume h is a shortest history with that property. As we can always pick the honest choice along h^*, h has to be off the honest history and for each choice $a \in A(h)$ there has to exist a superset of S (other than N) against which $\Gamma_{|(h,a)}$ is not collusion resilient.

Consider the set S of players who had to deviate from the partially defined strategy σ to reach h. Pick now the last time in h where a player $p \notin S$ has a turn, and call the respective history t. If only players of S ever have turns along h, we define $t := \emptyset$. In any case $\Gamma_{|t}$ is collusion resilient against all supersets of S (other than N) by construction (since we could pick an action according to the theorem to reach t; and if $t = \emptyset$ because Γ is collusion resilient).

From the proof of Theorem 5.6.3 for collusion resilience follows that there exists a strategy σ^S that is collusion resilient against all supersets S' of S, $S' \neq N$.

Consider action $a \in A(h)$, with $\sigma^S(h) = a$. Such an a has to exist since $\Gamma_{|t}$ is a supertree of $\Gamma_{|h}$. By assumption, $\Gamma_{|(h,a)}$ is not collusion resilient against at least one S'. Fix such an S'. Therefore, there exists a strategy $\tau^a \in \mathcal{S}_{|(h,a)}$ such that

$$\sum_{p \in S'} u_p^* < \sum_{p \in S'} u_{|(h,a),p}(\sigma^S_{|(t',a),N-S'}, \tau^a_{S'}) \,, \tag{16}$$

where u^* is the honest utility $u(h^*)$ and t' is the suffix of h after t: $(t, t') = h$.

Towards a contradiction, we construct $\tau \in \mathcal{S}_{|t}$ as follows. Let $\tau_{|(t',a)} = \tau^a$, and let τ yield (t', a). The rest can be picked arbitrarily. It can be observed that

$$u_{|(h,a)}(\sigma^S_{|(t',a),N-S'}, \tau^a_{S'}) = u_{|t}(\sigma^S_{N-S'}, \tau_{S'}) \,, \tag{17}$$

as only players $p \in S'$ have turns in t', $\sigma^S(h) = a$, $\tau(h) = a$ and $h = (t, t')$. But σ^S is collusion resilient against S' in Γ_t, hence

$$\sum_{p \in S'} u_{|t,p}(\sigma^S_{N-S'}, \tau_{S'}) \leq \sum_{p \in S'} u_p^* \,. \tag{18}$$

Finally, Equations (16) to (18) yield a contradiction.

What remains to be shown is that the constructed strategy σ is collusion resilient and honest in Γ. It yields the honest history h^* by construction. We again prove the collusion resilience by contradiction and assume σ is not cr (but the tree Γ still is collusion resilient, for another strategy). Then, there has to exist a set of players $S \subset N$ and a strategy $\tau \in \mathcal{S}$ such that

$$\sum_{p \in S} u_p(\sigma) = \sum_{p \in S} u_p^* < \sum_{p \in S} u_p(\sigma_{N-S}, \tau_S) \ . \tag{19}$$

Let h be the prefix of history $H(\sigma_{N-S}, \tau_S)$ at which for the last time in $H(\sigma_{N-S}, \tau_S)$ an honest player $p = P(h) \notin S$ has a turn. Note that such an h has to exist and has to be off the honest history h^*. If this were not the case, τ would cause every honest strategy to not be collusion resilient as the right side of inequality in (19) would not depend on σ_{N-S} at all, which would imply that h^* is not collusion resilient.

By the construction of σ, we know that $a = \sigma(h)$ leads to a subtree $\Gamma_{|(h,a)}$ that is collusion resilient against the set of deviating players S^d (deviating from σ to h) and all supersets (other than N). Further, $S \supseteq S^d$ is such a superset, since more players could deviate from σ in other parts of the tree. Hence, there exists a strategy $\sigma^S \in \mathcal{S}_{|(h,a)}$ that is collusion resilient against S. Therefore, also for the considered strategy τ

$$\sum_{p \in S} u_p^* \geq \sum_{p \in S} u_{|(h,a),p}(\sigma^S_{N-S}, \tau_{|(h,a),S}) \ . \tag{20}$$

Note that after h no more honest players have a turn in $H(\sigma_{N-S}, \tau_S)$, and $(\sigma_{N-S}, \tau_S)(h) = \sigma(h) = a$, because $P(h) \in N - S$. Therefore, $H(\sigma_{N-S}, \tau_S) = (h, a, H(\tau_{|(h,a)}))$. This implies that also no honest player has a turn in $H(\sigma^S_{N-S}, \tau_{|(h,a),S})$, as $H(\sigma^S_{N-S}, \tau_{|(h,a),S}) = H(\tau_{|(h,a)})$. Since the histories align, the strategies also have to yield the same utilities $u(\sigma_{N-S}, \tau_S) = u_{|(h,a)}(\sigma^S_{N-S}, \tau_{|(h,a),S})$. This contradicts Equations (19) and (20) and shows that σ is an honest and collusion resilient strategy. □

Theorem D.3 (Practical Strategies). *Let Γ be a game tree, $\preceq$ a total order, $h \in \mathscr{H}$ a history. Further, let u be a utility practical (under $\preceq$) in $\Gamma_{|h}$. The following strategy $\sigma^u \in \mathcal{S}_{|h}$ yields utility u and is practical (under $\preceq$).*

1. *For action $a \in A(h)$, where u is practical in $\Gamma_{|(h,a)}$ (under $\preceq$) we set $\sigma^u(h) = a$ and $\sigma^u_{|(a)} = \sigma^{u,a}$, where $\sigma^{u,a} \in \mathcal{S}_{|(h,a)}$ is a practical strategy in $\Gamma_{|(h,a)}$ that yields utility u.*
2. *For all other $a' \in A(h) \setminus \{a\}$ we define $\sigma^u_{|(a')} = \sigma^{u',a'}$, where u' is a utility practical in $\Gamma_{|(h,a')}$, $\sigma^u_{|(a')}$ a practical strategy yielding u' and for all $\vec{x}$ satisfying $\preceq$: $u_{P(h)}[\vec{x}] \geq u'_{P(h)}[\vec{x}]$.*

Similarly to our previous results, Theorem D.3 provides an algorithmic solution to extract a practical strategy for the honest history. For a game tree Γ with practical honest history h^* and total order $\preceq$, an honest and practical strategy can be computed as follows: bottom-up, for each subtree and each practical utility, a corresponding practical strategy is stored. When proceeding one level up, we do as stated in Theorem D.3. Along the honest history, the only stored practical strategy is the one for the honest utility $u^* = u(h^*)$ as, ultimately, this is the only one required at the root.

Proof. According to Theorem 5.12, u can only be practical in $\Gamma_{|h}$, if it was practical in at least one child $\Gamma_{|(h,a)}$. For every other child a' there had to exist a utility u' practical in $\Gamma_{|(h,a')}$ such that $u_{P(h)} \geq u'_{P(h)}$. Hence, strategy σ^u can always be constructed. It further yields utility u (by construction) and is practical since by construction and the proof of Lemma C.1 no player can at any point of the game deviate profitably. □

Pseudo-Algorithm for Counterexamples to Collusion Resilience. When analyzing the collusion resilience against a group of players S, whenever it is the turn of one of the players in S and there exists an action leading to a not collusion resilient against S subtree (line 14 with split = null in Algorithm 4), we store the action, the current history and player group S.

After the analysis terminated and the result was not collusion resilient, we generate a counterexample of the collusion resilience against player group S by walking through the tree again: Starting from the root, we proceed as follows, assuming the current history is h.

- If $P(h) \notin S$ and h is along the honest history, we follow the honest action to the honest subtree. This is sufficient since an honest player $P(h)$ follows the honest history.
- If $P(h) \notin S$ but h is not along the honest history, all choices had to lead to not collusion resilient against S subtrees for the current tree to be not collusion resilient against S. We, therefore, have to follow all choices to compute a counterexample.
- Otherwise, if $P(h) \in S$, we check our stored data for a choice a that is not collusion resilient against S. By construction and Theorem 5.10, it has to exist. We add it to our partial strategy s_S, i.e. $s_S(h) = a$. Then, we continue at history (h, a).
- At a leaf nothing has to be considered. A not collusion resilient against S leaf has a joint utility for S greater than their joint honest utility.

With the same arguments as for weak(er) immunity, we conclude that the generated partial strategy s_S together with player group S is a counterexample to the collusion resilience of game Γ with the considered honest history.

Pseudo-Algorithm for Counterexamples to Practicality. When analyzing the tree according to Algorithm 5, we can only return "false" (i.e. an empty list), together with case split null along the honest history. This is the case exactly when, for at least one sibling, all practical utilities are strictly better for the current player than the honest one (line 15). Before returning, we store the action a leading to said sibling together with the set of its practical utilities $\mathbb{U}(h, a)$. For convenience, we also provide the histories $t \in \mathscr{H}_{|(h,a)}$ to those utilities $u(t) = u \in \mathbb{U}(h, a)$. Using Theorem 5.12, we thus computed a counterexample to practicality.

Remark. It is also possible to compute *all* counterexamples to a security property. This can be done by simply storing *all* actions that lead to not weak(er) immune, respectively collusion resilient subtrees in the pseudo-algorithms, for weak(er) immunity and collusion resilience. For practicality, it requires storing all siblings along the entire honest history, whose practical utilities lead to a better-than-honest utility.

E Benchmarks

The experimental results of all benchmarks are reported in Table 3. Further, the exhaustive data for counterexample generation is presented in Table 4. Note that in the last line of Table 4, where the runtime for counterexample generation of the Unlocking Routing benchmark is listed, we report *error*. This means that when running CheckMate on that instance we encountered an exception thrown from CheckMate's Z3 backend.

As briefly mentioned in Section 7, we also compared the two approaches in terms of efficiency for strategy extraction and report the results in Table 5. The findings closely mirror those observed for counterexamples. Firstly, strategy extraction in the compositional approach outperforms the previous method across nearly all benchmarks. Secondly, the compositional approach incurs almost no additional overhead for strategy extraction, maintaining its overall runtime efficiency.

One benchmark that stands out is Tic Tac Toe, where the additional overhead for strategy extraction is clearly noticeable for collusion resilience and practicality. However, strategy extraction

	Game	Nodes	Players	Security property	Secure yes/no	Time	Nodes evaluated CheckMate2.0/CheckMate	Nodes evaluated (reps)	Calls
small toy examples	$\text{Splits}_{\text{wi}}$	5	2	wi	y	0.010 / 0.018	5	18 / 10	10 / 3
	(q)			weri	y	0.010 / 0.018	5	18 / 10	10 / 3
				cr	y	0.010 / 0.015	4 / 5	6 / 10	3 / 1
				pr	y	0.011 / 0.017	5	25 / 5	19 / 3
	$\text{Splits}_{\text{cr}}$	5	2	wi	y	0.011 / 0.019	4 / 5	6 / 10	3 / 1
	(n)			weri	y	0.011 / 0.019	4 / 5	6 / 10	3 / 1
				cr	y	0.011 / 0.018	5	16 / 10	10 / 3
				pr	y	0.011 / 0.018	5	15 / 5	20 / 3
	Market Entry	5	2	wi	n	0.011 / 0.014	5	8 / 10	5 / 1
	(e, i)			weri	n	0.010 / 0.014	5	8 / 10	5 / 1
				cr	y	0.010 / 0.014	5	8 / 10	4 / 1
				pr	y	0.010 / 0.015	5	5	2 / 1
	G	5	2	wi	n	0.010 / 0.012	5 / 5	28 / 10	18 / 4
	(r_A, l_B)			weri	n	0.009 / 0.012	5 / 5	28 / 10	18 / 4
				cr	n	0.008 / 0.010	2 / 5	2 / 10	2 / 1
				pr	n	0.009 / 0.010	5 / 5	9 / 5	5 / 1
	Simplified Closing	8	2	wi	y	0.009 / 0.012	8	10 / 16	8 / 1
	(H)			weri	y	0.009 / 0.011	8	10 / 16	8 / 1
				cr	y	0.008 / 0.011	7 / 8	9 / 16	6 / 1
				pr	n	0.009 / 0.012	8	8	8 / 1
	(C_h, S)			wi	n	0.008 / 0.012	3 / 8	3 / 16	2 / 1
				weri	n	0.008 / 0.011	3 / 8	3 / 16	2 / 1
				cr	y	0.008 / 0.012	8	11 / 16	7 / 1
				pr	y	0.009 / 0.013	8	8	6 / 1
medium-sized games	Simplified Routing	17	5	wi	n	0.008 / 0.012	7 / 17	7 / 85	2 / 1
	$(S_H, L, L, L, L, U, U, U, U)$			weri	y	0.009 / 0.011	17	77 / 85	28 / 1
				cr	n	0.010 / 0.017	16 / 17	105 / 510	24 / 1
				pr	y	0.009 / 0.012	17	17	8 / 1
	Centipede	19	3	wi	n	0.044 / 0.051	19	602 / 57	345 / 18
	$(C, C, C, C, C, C, C, C, C)$			weri	n	0.033 / 0.052	19	602 / 57	345 / 18
				cr	n	0.044 / 0.038	19	534 / 114	305 / 9
				pr	n	0.011 / 0.028	19	103 / 19	39 / 7
	EBOS	31	4	wi	n	0.009 / 0.013	28 / 31	38 / 124	21 / 1
	$(Mine, Mine, Mine, Mine)$			weri	n	0.008 / 0.013	28 / 31	38 / 124	21 / 1
				cr	n	0.039 / 0.021	31	476 / 434	304 / 4
				pr	n	0.019 / 0.024	31	167 / 31	184 / 5
	Pirate	79	4	wi	n	0.010 / 0.015	10 / 79	10 / 316	5 / 1
	(y, n, n, n, y, y)			weri	n	0.009 / 0.016	10 / 79	10 / 316	5 / 1
				cr	n	0.041 / 0.029	79	622 / 1106	368 / 4
				pr	n	0.036 / 0.049	79	482 / 79	554 / 8
	Auction	92	4	wi	n	0.012 / 0.033	16 / 92	16 / 368	9 / 1
	(E, E, I, I)			weri	y	0.016 / 0.027	90 / 92	229 / 368	162 / 1
				cr	n	0.018 / 0.030	66 / 92	128 / 1,288	103 / 1
				pr	y	0.021 / 0.145	92	92	188 / 1
real-world models	Closing	221	2	wi	y	0.011 / 0.024	20 / 221	22 / 442	16 / 1
	(H)			weri	y	0.010 / 0.021	20 / 221	22 / 442	16 / 1
				cr	y	0.012 / 0.023	44 / 221	46 / 442	36 / 1
				pr	n	0.097 / 0.346	221	568 / 221	1454 / 1
	(C_h, S)			wi	y	0.011 / 0.024	33 / 221	36 / 442	25 / 1
				weri	y	0.011 / 0.020	33 / 221	36 / 442	25 / 1
				cr	y	0.013 / 0.023	60 / 221	63 / 442	48 / 1
				pr	y	2.144 / 0.345	221	14353 / 221	38220 / 1
	3-Player Routing	21,688	3	wi	n	0.248 / 0.984	16 / 21,688	16 / 65,064	9 / 1
	(S_H, L, L, U, U)			weri	y	0.514 / 1.008	7,084 / 21,688	7,570 / 65,064	5,441 / 1
				cr	n	0.272 / 1.886	430 / 21,688	474 / 130,128	299 / 1
				pr	n	33.162 / 34.717	21,688	416,156 / 21,688	569,418 / 13
	Unlocking Routing	36,113	5	wi	n	0.621 / 2.121	1,184 / 36,113	1,184 / 180,565	714 / 1
	(U, U, U, U)			weri	y	1.525 / 1.625	32,429 / 36,113	55,090 / 180,565	27,897 / 1
				cr	n	0.584 / 15.247	319 / 36,113	373 / 1,083,390	60 / 1
				pr	y	2.848 / 4.382	36,113 / 36,113	36,113 / 36,113	46,636 / 1
	Tic Tac Toe Concise	58,748	2	wi	y	0.557 / 6.372	1,345 / 58,748	1,355 / 117,496	698 / 1
	$(CM, LU, RU, LD, LM$			weri	y	0.541 / 6.373	1,345 / 58,748	1,355 / 117,496	698 / 1
	$RM, CU, CD, RD)$			cr	y	0.543 / 7.352	1,345 / 58,748	1,355 / 117,496	698 / 1
				pr	y	3.937 / 227.807	58,748 / 58,748	58,748 / 58,748	57,250 / 1
	Tic Tac Toe	549,946	2	wi	y	5.276 / 255.368	18,026 / 549,946	18,036 / 1,099,892	10,694 / 1
	$(CM, RU, LU, RD, RM,$			weri	y	5.256 / 255.600	18,026 / 549,946	18,036 / 1,099,892	10,694 / 1
	$LM, CU, CD, LD)$			cr	y	5.302 / 286.574	18,026 / 549,946	18,036 / 1,099,892	10,694 / 1
				pr	y	36.530 / TO	549,946 / TO	549,946 / TO	527,198 / TO

Table 3. Full experimental results of game-theoretic security, using the compositional CheckMate2.0 approach and the non-compositional CheckMate setting of [Rain et al. 2024]. Runtimes are given in seconds, with a timeout (TO) after 8 hours. For each game, columns 2–3 list the size (tree nodes and game players) of the game from column 1. Column 4 shows the game-theoretic security property we analyzed and (dis)proved, as indicated in column 5. Columns 6–9 presents the results of CheckMate2.0 compared to CheckMate, using the slash / sign.

Game	Property	Time (one CE) CheckMate2.0/CheckMate	Time (all CE) CheckMate2.0/CheckMate
Market Entry	wi	0.010 / 0.016	0.010 / 0.023
(e, i)	weri	0.010 / 0.017	0.009 / 0.019
G	wi	0.010 / 0.013	0.010 / 0.020
(r_A, l_B)	weri	0.010 / 0.014	0.009 / 0.020
	cr	0.008 / 0.011	0.008 / 0.013
	pr	0.009 / 0.016	0.009 / 0.028
Simplified Closing (H)	pr	0.009 / 0.019	0.008 / 0.019
(C_h, S)	wi	0.009 / 0.014	0.008 / 0.016
	weri	0.009 / 0.014	0.008 / 0.014
Simplified Routing	wi	0.009 / 0.014	0.010 / 0.033
$(S_H, L, L, L, L, U, U, U, U)$	cr	0.010 / 0.023	0.016 / 0.096
Centipede	wi	0.046 / 0.049	0.080 / 0.495
$(C, C, C, C, C, C, C, C, C)$	weri	0.034 / 0.050	0.061 / 0.495
	cr	0.045 / 0.047	0.078 / 0.538
	pr	0.012 / 0.062	0.022 / 0.400
EBOS	wi	0.010 / 0.015	0.011 / 0.058
$(Mine, Mine, Mine, Mine)$	weri	0.010 / 0.015	0.010 / 0.057
	cr	0.040 / 0.028	0.057 / 10.760
	pr	0.020 / 0.032	0.021 / 0.032
Pirate	wi	0.010 / 0.020	0.157 / 9.465
(y, n, n, n, y, y)	weri	0.009 / 0.020	0.157 / 9.495
	cr	0.041 / 0.039	3.232 / 79.839
	pr	0.037 / 0.064	7.414 / 35.227
Auction	wi	0.012 / 0.048	0.025 / 4.172
(E, E, I, I)	cr	0.018 / 0.066	0.036 / 15.106
Closing (H)	pr	0.096 / 0.650	2.204 / 8.846
3-Player Routing	wi	0.251 / 1.925	5.909 / 110.716
(S_H, L, L, U, U)	cr	0.279 / 5.619	1.657 / 7.815
	pr	33.561 / 46.480	291.236 / 3 033.784
Unlocking Routing	wi	0.602 / 5.219	2.090 / 1 988.997
(U, U, U, U)	cr	0.564 / 116.906	3.562 / error

Table 4. Full experiments on counterexample (CE) generation using our CheckMate2.0 approach and the non-compositional CheckMate tool of [Rain et al. 2024]. Runtimes are given in seconds; *error* means we encountered an exception thrown from CheckMate's Z3 backend.

for these properties is still achievable within reasonable time, namely 18 seconds for collusion resilience and 276 seconds for practicality. This represents a significant improvement over the non-compositional approach, which takes 347 seconds for collusion resilience and fails to terminate within the 8-hour time limit for practicality.

Game	Property	Time CheckMate2.0/CheckMate
$\text{Splits}_{\text{wi}}$	wi	0.010 / 0.019
(q)	weri	0.010 / 0.018
	cr	0.010 / 0.015
	pr	0.011 / 0.017
$\text{Splits}_{\text{cr}}$	wi	0.011 / 0.018
(n)	weri	0.011 / 0.018
	cr	0.011 / 0.019
	pr	0.011 / 0.018
Market Entry	cr	0.010 / 0.014
(e, i)	pr	0.010 / 0.014
Simplified Closing	wi	0.009 / 0.012
(H)	weri	0.009 / 0.011
	cr	0.009 / 0.012
(C_h, S)	cr	0.010 / 0.012
	pr	0.010 / 0.014
Simplified Routing	weri	0.010 / 0.011
$(S_H, L, L, L, L, U, U, U, U)$	pr	0.010 / 0.012
Auction	weri	0.017 / 0.028
(E, E, I, I)	pr	0.022 / 0.153
Closing	wi	0.011 / 0.025
(H)	weri	0.011 / 0.022
	cr	0.023 / 0.025
(C_h, S)	wi	0.012 / 0.025
	weri	0.011 / 0.021
	cr	0.024 / 0.025
	pr	2.185 / 0.0364
3-Player Routing	weri	0.539 / 1.163
(S_H, L, L, U, U)		
Unlocking Routing	weri	2.194 / 4.233
(U, U, U, U)	pr	4.241 / 5.718
Tic Tac Toe Concise	wi	0.556 / 7.003
$(CM, LU, RU,$	weri	0.561 / 7.780
$LD, LM, RM,$	cr	1.883 / 8.894
$CU, CD, RD)$	pr	8.644 / 219.689
Tic Tac Toe	wi	5.509 / 276.333
$(CM, RU, LU,$	weri	5.507 / 306.763
$RD, RM, LM,$	cr	18.608 / 347.093
$CU, CD, LD)$	pr	276.719 / TO

Table 5. Full experiments on strategy extraction using our CheckMate2.0 approach and the non-compositional CheckMate tool of [Rain et al. 2024]. Runtimes are given in seconds, with a timeout (TO) after 8 hours.