INSTITUT FÜR INFORMATIK

# A Dolev-Yao-based Definition of Abuse-free Protocols

Detlef Kähler, Ralf Küsters, Thomas Wilke

Bericht Nr. 0607
Mai 2006


CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

# A Dolev-Yao-based Definition of Abuse-free Protocols

Detlef Kähler, Ralf Küsters, and Thomas Wilke

Christian-Albrechts-Universität zu Kiel
{kaehler,kuesters,wilke}@ti.informatik.uni-kiel.de

**Abstract.** We propose a Dolev-Yao-based definition of abuse freeness for optimistic contract-signing protocols which, unlike other definitions, incorporates a rigorous notion of what it means for an outside party to be convinced by a dishonest party that it has the ability to determine the outcome of the protocol with an honest party, i.e., to determine whether it will obtain a valid contract itself or whether it will prevent the honest party from obtaining a valid contract. Our definition involves a new notion of test (inspired by static equivalence) which the outside party can perform. We show that an optimistic contract-signing protocol proposed by Asokan, Shoup, and Waidner is abusive and that a protocol by Garay, Jakobsson, and MacKenzie is abuse-free according to our definition. Our analysis is based on a synchronous concurrent model in which parties can receive several messages at the same time. This results in new vulnerabilities of the protocols depending on how a trusted third party reacts in case it receives abort and resolve requests at the same time.

## 1 Introduction

Abuse-freeness is a security property introduced in [10] for optimistic contract-signing protocols: An optimistic (two-party) contract-signing protocol is a protocol run by $A$ (Alice), $B$ (Bob), and a trusted third party $T$ (TTP) to exchange signatures on a previously agreed upon contractual text with the additional property that the TTP will only be involved in a run in case of problems. Such a protocol is *not* abuse-free for (honest) Alice if at some point during a protocol run (dishonest) Bob can "convince" an outside party $C$ (Charlie) that he is in an unbalanced state, where, following the terminology of [5], *unbalanced* means that Bob has both (i) a strategy to prevent Alice from getting a valid contract and (ii) a strategy to obtain a valid contract. In other words, Alice can be misused by Bob to get leverage for another contract (with Charlie). Obviously, abuse-free contract-signing protocols are highly desirable.

The main goal of the present work is to present a formal definition of abuse-freeness which is as protocol-independent as possible. The crucial issue with such a formal definition is that it needs to specify what it means for Bob to *convince* Charlie. One of the first proposals for this was presented by Kremer and Raskin [12]. Roughly, their proposal is the following: to convince Charlie a message is presented to Charlie from which he can deduce that "a protocol run has been

started between Alice and Bob". What that means is, however, not specified in a general fashion in [12]. Instead, this is decided on a case by case basis. The objective of this paper is to give a generic definition. The only part which needs to be decided on a case by case basis in our definition is what it means for Alice (or Bob) to have received a valid contract, something which can hardly be described in a generic way, and what the assumptions are that Charlie makes.

Before we explain our approach and the contribution of our work we need to explain the following crucial point: Whether or not Charlie is convinced should be based on evidence provided by Bob. Following [10], we model this evidence as a message that Bob presents to Charlie. (In [10], this is called an off-line attack.) This, however, has an important implication. Since Bob can hold back any message he wants to (he can himself decide which messages he shows to Charlie) and since Charlie is assumed to be an outside party not involved in the protocol, if Bob could convince Charlie to be in some state of the protocol at some point, at any later point he would be able to convince Charlie that he was in the same state, just by providing the same evidence. Therefore, Bob can only convince Charlie that he is or was and still might be in an unbalanced state. We employ this notion of abuse-freeness for our work. Note that this notion is stronger than the one described at the beginning (since Charlie is more easily convinced). Hence, a protocol secure w.r.t. the version of abuse-freeness just explained is also secure w.r.t. that version.

*Contribution of this Work.* We provide a formal definition of the version of abuse-freeness just explained, apply our definition to the optimistic contract-signing protocols by Asokan, Shoup, and Waidner [3] (ASW protocol) and by Garay, Jakobsson, and MacKenzie [10] (GJM protocol), and show, as one would expect, that the ASW protocol is abusive while the GJM protocol is abuse-free according to our definition.

The idea behind our definition of abuse-freeness is that Bob presents a message to Charlie and Charlie performs a certain test on this message. If the message passes the test, then Charlie is convinced that Bob is or was and still might be in an unbalanced state. The test is such that from the point of view of Charlie, Bob can only generate messages passing the test in states where Bob is or was in an unbalanced state and where at least one of these states is in fact unbalanced. To describe the power Bob has, we adopt a Dolev-Yao style approach [9] (see also [2, 1, 8]). Our definition of test is inspired by the notion of static equivalence [2].

We use a synchronous concurrent communication model in which principals and the (Dolev-Yao-style) intruder may send several messages to different parties at the same time. This rather realistic model requires to specify the behavior of protocol participants in case several messages are received at the same time (or within one time slot). This leads to new effects that have not been observed in previous works. In the ASW and GJM protocols, one needs to specify the behavior of the TTP in case an abort and a resolve request are received at the same time (from different parties). The question arises whether the TTP should answer with an abort or a resolve request. We show that if the TTP does the

former, then the ASW and the GJM protocol are unbalanced for the responder, and in the other case, they are unbalanced for the initiator.

*Related Work.* As mentioned above, Kremer et al. [12] analyze the ASM and GJM protocol based on finite-state alternating transition systems, using an automatic analysis tool. They explicitly need to specify the behavior of dishonest principals and which states are the ones that are convincing to Charlie (they use a propositional variable prove2C, which they set manually). This is what our definition makes obsolete.

Chadha et al. [5] introduce a stronger notion than abuse-freeness, namely *balance*: For a protocol to be *unbalanced* one does not require Bob to convince Charlie that he is in an unbalanced state. The fact that an unbalanced state exists is sufficient for a protocol to be unbalanced. Hence, balance is a formally stronger notion than abuse-freeness. Unfortunately, this notion is too strong in some cases. In fact, under certain conditions balance is impossible to achieve [7]. In [6], Chadha et al. also study multi-party contract signing protocols.

Shmatikov and Mitchell [13] employ the finite-state model checker Mur$\phi$ to automatically analyze contract-signing protocols. They, too, approximate the notion of abuse-freeness by a notion similar to balance.

*Structure of the Paper.* The technical part of the report starts with an informal description of the ASW and GJM protocols in Sect. 2. The ASW protocol then serves as a running example for the further definitions. In Sect. 4, we describe our communication and protocol model, with background given in Section 3. In Sect. 5 we present our new definition of abuse-freeness. We then treat the ASW and the GJM protocol in our framework in Sect. 6 and Sect. 7. We conclude in Section 8.

## 2 The ASW and the GJM protocol

In this section, we recall the two contract-signing protocols which we use to explain our framework and which we analyze within that framework: the Asokan-Shoup-Waidner (ASW) protocol from [3] and the Garay-Jakobsson-MacKenzie (GJM) protocol from [10]. Both of them are optimistical contract-signing protocols, which means that a trusted third party gets involved only if a participant deviates from the regular course of actions. This makes these protocols efficient on the one hand, but complicated to analyse on the other hand.

As will be explained below, the ASW protocol, which uses conventional cryptographic means, is not abuse-free [10]. As a remedy, the authors of that paper suggested a new cryptographic primitive, a so-called "private contract signature", and a new protocol, the so-called GJM protocol.

### 2.1 The ASW protocol

The protocol assumes the following scenario: Alice ($A$) and Bob ($B$) want to sign a contract and Tom ($T$) will serve as a trusted third party in the network.

Further, it is agreed upon in the network that the following two types of messages will be recognized as valid contracts between Alice and Bob with contractual text

$$SC = \langle me_1, N_A, me_2, N_B \rangle \ , \qquad RC = \mathsf{sig}_T(\langle me_1, me_2 \rangle) \ ,$$

where $me_1$ and $me_2$ are defined by

$$me_1 = \mathsf{sig}_A(\langle A, B, T, \mathsf{text}, \mathsf{hash}(N_A) \rangle) \ , \quad me_2 = \mathsf{sig}_B(\langle me_1, \mathsf{hash}(N_B) \rangle) \ .$$

Here, as usual, $N_A$ and $N_B$ stand for nonces. In addition to $SC$ and $RC$, the variants of $SC$ and $RC$ which one obtains by exchanging the roles of $A$ and $B$ are regarded as valid contracts. Messages of type $SC$ are called *standard contracts*, while messages of type $RC$ are called *replacement contracts*.

There are three interdependent parts to the protocol: an exchange protocol, an abort protocol, and a resolve protocol. The *exchange protocol* consists of four steps, which, in Alice-Bob notation, are as follows:

$$A \to B\colon me_1 \tag{E1}$$
$$B \to A\colon me_2 \tag{E2}$$
$$A \to B\colon N_A \tag{E3}$$
$$B \to A\colon N_B \tag{E4}$$

In other words, $me_1$ and $me_2$ serve as respective *promises* of Alice and Bob to sign the contract and $N_A$ and $N_B$ serve as *contract authenticators*: after they have been revealed, Alice and Bob can compose the standard contract $SC$.

The *abort protocol* is used by Alice to abort the contract signing process when it does not receive Bob's promise within a reasonable period of time. Alice will obtain an abort receipt or, if the protocol instance has already been resolved (see below), a replacement contract:

$$A \to T\colon ma_1 := \mathsf{sig}_A(\langle \mathsf{aborted}, me_1 \rangle) \tag{A1}$$
$$T \to A\colon \text{if } \mathsf{DB}(me_1) = (\mathsf{resolved}, s), \text{ then}$$
$$s \tag{A2.1}$$
$$\text{else}$$
$$\mathsf{DB}(me_1) := (\mathsf{aborted}, \mathsf{sig}_T(\mathsf{aborted}, ma_1)); \ \mathsf{sig}_T(\langle \mathsf{aborted}, ma_1 \rangle) \tag{A2.2}$$

where $\mathsf{DB}(me_1)$ denotes the entry for the promise $me_1$ in $T$'s database, which is used by $T$ to keep track of aborted and resolved protocol instances. In the notation we adopt the convention that the last expression of a sequence of expressions determines the result of evaluating the sequence.

The resolve protocol can be used by Alice or Bob to resolve the protocol, which either results in a replacement contract or, if the protocol has already

been abortet, in an abort receipt:

$$B \to T\colon \langle me_1, me_2\rangle \tag{R1}$$
$$T \to B\colon \text{if } \mathsf{DB}(me_1) = (\mathsf{aborted}, s), \text{ then}$$
$$s \tag{R2.1}$$
$$\text{else}$$
$$\mathsf{DB}(me_1) := (\mathsf{resolved}, \mathsf{sig}_T(\langle me_1, me_2\rangle)); \mathsf{sig}_T(\langle me_1, me_2\rangle) \tag{R2.2}$$

The same protocol (with roles of $A$ and $B$ exchanged) is also used by Alice when she wants to resolve the protocol after she has already received Bob's promise.

It is assumed that the communication between Alice and the TTP and between Bob and the TTP goes through a channel that is not under the control of the intruder (the dishonest party), i.e., the intruder cannot delay, modify, or insert messages. We refer to such a channel as secure. Whether or not the intruder can read messages sent on this channel does not effect the results shown in this paper.

### 2.2 The GJM protocol

The GJM protocol is similar to the ASW protocol with regard to its general structure: it consists of an exchange subprotocol, an abort subprotocol, and a resolve subprotocol. To guarantee abuse-freeness, however, the protocol uses so-called "private contract signatures" rather than universally verifiable signatures. Specifically, a private contract signature $\mathsf{pcsig}_A(m, B, T)$ by $A$ for $B$ and with trusted third party $T$ is designed in such a way that: (i) it could also have been generated by $B$ (which means it cannot be used by $B$ to convince Charlie that $A$ has been involved), (ii) $A$ can convert it into a universally verifiable signature on $m$ by running $\mathsf{s\text{-}conv}_A(\cdot)$ on it, and (iii) $T$ can do the same by running $\mathsf{t\text{-}conv}_A(\cdot)$ on it. With these new primitives, the protocol can be described.

The *GJM exchange subprotocol* is:

$$A \to B\colon me_1 := \mathsf{pcsig}_A(\mathsf{text}, B, T)$$
$$B \to A\colon me_2 := \mathsf{pcsig}_B(\mathsf{text}, A, T)$$
$$A \to B\colon \mathsf{s\text{-}conv}_A(me_1)$$
$$B \to A\colon \mathsf{s\text{-}conv}_B(me_2)$$

It is similar in spirit to the ASW exchange protocol: Alice and Bob first exchange promises and then authenticators; a standard contract is of the form $\langle \mathsf{s\text{-}conv}_A(me_1), \mathsf{s\text{-}conv}_B(me_2)\rangle$.

The *abort subprotocol* is given by

$$A \to T\colon ma_1 := \mathsf{sig}_A(\langle \mathsf{text}, A, B, \mathsf{aborted}\rangle)$$
$$T \to A\colon \text{if } \mathsf{DB}(\mathsf{text}) = (\mathsf{resolved}, \_, s), \text{ then}$$
$$s$$
$$\text{else}$$
$$\mathsf{DB}(\mathsf{text}) := (\mathsf{aborted}, A, \mathsf{sig}_T(ma_1)); \mathsf{sig}_T(ma_1))$$

The *resolve subprotocol* for $B$ is given by

$$\begin{aligned} B \to T\colon\ & \langle me_1, ms := \mathsf{s\text{-}conv}_B(\mathsf{pcsig}_B(\mathsf{text}, A, T))\rangle \\ T \to B\colon\ & \text{case } \mathsf{DB}(\mathsf{text}) \text{ of} \\ & \quad \mid (\mathsf{aborted}, A, s) \Rightarrow s \\ & \quad \mid (\mathsf{resolved}, A, s) \Rightarrow s \\ & \quad \mid \_ \Rightarrow \mathsf{t\text{-}conv}_T(me_1) \end{aligned}$$

while Alice's resolve protocol is given by

$$\begin{aligned} A \to T\colon\ & \langle me_2, ms := \mathsf{s\text{-}conv}_A(\mathsf{pcsig}_A(\mathsf{text}, B, T))\rangle \\ T \to A\colon\ & \text{case } \mathsf{DB}(\mathsf{text}) \text{ of} \\ & \quad \mid (\mathsf{aborted}, A, s) \Rightarrow s \\ & \quad \mid (\mathsf{resolved}, B, s) \Rightarrow s \\ & \quad \mid \_ \Rightarrow \mathsf{t\text{-}conv}_T(me_2) \end{aligned}$$

From this, it becomes clear that a replacement contract has the same format as the standard with only one difference, either $\mathsf{s\text{-}conv}_A(\cdot)$ or $\mathsf{s\text{-}conv}_B(\cdot)$ is replaced by $\mathsf{t\text{-}conv}_T(\cdot)$.

## 3 Background on Messages, Terms, and Derivation of Messages

Along the lines of [1, 2, 8], we define messages, terms, and the way new messages can be derived from a given set of messages. Properties of cryptographic operators are defined by equational theories.

### 3.1 Messages and Terms

A *signature* $\Sigma$ is a set of *function symbols* where each symbol $f \in \Sigma$ is associated with a non-negative integer $n_f$, the *arity* of $f$. As usual, elements $c \in \Sigma$ with $n_c = 0$ are called *constants*. A signature $\Sigma$ may contain a (countably) infinite number of constants but only a finite number of non-constant function symbols.

*Example 1 (ASW signature).* For the ASW protocol in Section 6, we will use the following signature:

$$\begin{aligned} \Sigma_{ASW} = \{&\mathsf{sig}(\cdot,\cdot), \mathsf{sigcheck}(\cdot,\cdot,\cdot), \mathsf{pk}(\cdot), \mathsf{sk}(\cdot), \langle\cdot,\cdot\rangle, \pi_1(\cdot), \pi_2(\cdot), \mathsf{hash}(\cdot), \\ & A, B, T, \mathsf{text}, \mathsf{ok}, \mathsf{initiator}, \mathsf{responder}, \\ & \mathsf{resolved}, \mathsf{aborted}\} \cup \mathcal{C} \end{aligned}$$

where $\mathcal{C}$ denotes an infinite set of constants (disjoint from the other constants defined in $\Sigma$). The function symbol $\mathsf{sig}(\cdot,\cdot)$ models signatures, $\mathsf{sigcheck}(\cdot,\cdot,\cdot)$ will be used to check the validity of a signature, $\mathsf{pk}(\cdot)$ and $\mathsf{sk}(\cdot)$ model public and

private keys, respectively, $\langle\cdot,\cdot\rangle$ is the pairing symbol, $\pi_1(\cdot)$ and $\pi_2(\cdot)$ yield the first and second component of a pair, respectively, and $\mathsf{hash}(\cdot)$ models the hash. The remaining symbols are constants.

Let $\mathcal{V}$ denote a (countably infinite) set of variables. The set $\mathcal{T}(\Sigma,\mathcal{V})$ of $\Sigma$-*terms* (or simply *terms*) over $\mathcal{V}$ is inductively defined as follows:

1. $\mathcal{V}\subseteq\mathcal{T}(\Sigma,\mathcal{V})$,
2. $f(t_1,\ldots,t_{n_f})\in\mathcal{T}(\Sigma,\mathcal{V})$ for each $f\in\Sigma$ and every choice $t_1,\ldots,t_{n_f}\in\mathcal{T}(\Sigma,\mathcal{V})$.

The set of variables occurring in $t$ is denoted by $\mathcal{V}(t)$.

A term without variables is called *ground*. The set of ground terms is denoted by $\mathcal{T}(\Sigma)$.

The *size* $|t|$ of a term $t\in\mathcal{T}(\Sigma,\mathcal{V})$ is defined inductively as follows: $|c|=1$ and $|x|=1$ for all constansts $c$ and variables $x$, and $|f(t_1,\ldots,t_n)|=1+|t_1|+\cdots+|t_n|$.

For a term $t$ of the form $f(t_1,\ldots,t_n)$, we refer to $f$ as the *root symbol of* $t$ and denote it by $\mathsf{root}(t)$.

The *set of positions* $\mathrm{Pos}(t)$ of a term $t$ is inductively defined as follows: $\mathrm{Pos}(c)=\mathrm{Pos}(x)=\{\varepsilon\}$ for every constant $c\in\Sigma$ and variable $x\in\mathcal{V}$ and $\mathrm{Pos}(f(t_1,\ldots,t_n))=\{\varepsilon\}\cup\bigcup_{i=1}^{n} i\cdot\mathrm{Pos}(t_i)$ where '$\cdot$' denotes concatenation. For example, $\mathrm{Pos}(\langle\mathsf{text},\mathsf{sig}(\mathsf{sk}(A),\mathsf{text})\rangle)=\{\varepsilon,1,2,21,211,22\}$.

For $p\in\mathrm{Pos}(t)$, we denote by $t|_p$ the subterm of $t$ rooted at position $p$. That is, $t|_\varepsilon=t$ and $f(t_1,\ldots,t_n)|_{i\cdot p}=t_i|_p$. For example, $\langle\mathsf{text},\mathsf{sig}(\mathsf{sk}(A),\mathsf{text})\rangle|_{211}=A$.

We denote by $\mathrm{Sub}(t)=\{t|_p\mid p\in\mathrm{Pos}(t)\}$ the set of subterms of $t$. Analogously, $\mathrm{Sub}(T)=\bigcup_{t\in T}\mathrm{Sub}(t)$ for a set $T$ of terms.

A *(ground) substitution* assigns (ground) terms to variables. The *domain* of a substitution is denoted by $\mathsf{dom}(\sigma)$ and is defined by $\mathsf{dom}(\sigma)=\{x\in\mathcal{V}\mid\sigma(x)\neq x\}$. Substitutions are required to have finite domains.

Given two substitutions $\sigma$ and $\tau$ with disjoint domains, their union $\sigma\cup\tau$ is defined in the obvious way. Given a term $t$, the term $t\sigma$ is obtained from $t$ by simultaneously substituting each variable $x$ occurring in $t$ by $\sigma(x)$. For substitutions $\sigma$ and $\tau$ we define the substitution $\sigma\tau$ by $\sigma\tau(x)=\tau(\sigma(x))$ for every $x$.

For terms $t,t_1,\ldots,t_n$, we denote by $t[t_1,\ldots,t_n/x_1,\ldots,x_n]$ the term $t\sigma$ with $\sigma=\{x_1\mapsto t_1,\ldots,x_n\mapsto t_n\}$.

### 3.2 Equational Theories

A pair $(s,t)\in\mathcal{T}(\Sigma,\mathcal{V})\times\mathcal{T}(\Sigma,\mathcal{V})$ is called a $\Sigma$-*identity* (or simply *identity*). Identities will be written as $l=r$. We call $l$ the *left-hand side* (lhs) and $r$ the *right-hand side* (rhs) of the identity $l=r$.

We call a set of $\Sigma$-identities an *equational theory over* $\Sigma$ (or simply an *equational theory*).

*Example 2 (ASW equational theory).* For the ASW protocol in Section 6, we will use the equational theory $\mathcal{H}_{ASW}$, consisting of the following three identities:

$$\mathsf{sigcheck}(x, \mathsf{sig}(\mathsf{sk}(y), x), \mathsf{pk}(y)) = \mathsf{ok} \ , \tag{1}$$

$$\pi_1(\langle x, y\rangle) = x \ , \tag{2}$$

$$\pi_2(\langle x, y\rangle) = y \ . \tag{3}$$

An equational theory $\mathcal{H}$ (over $\Sigma$) induces a *reduction relation* $\to_{\mathcal{H}} \subseteq \mathcal{T}(\Sigma, \mathcal{V}) \times \mathcal{T}(\Sigma, \mathcal{V})$ as follows: $s \to_{\mathcal{H}} t$ if there exist $(l, r) \in \mathcal{H}$, $p \in \mathrm{Pos}(s)$, and a substitution $\sigma$ such that $s|_p = \sigma(l)$ and $t = s[\sigma(r)]_p$ where $s[t]_p$ denotes the term constructed from $s$ by replacing the term at position $p$ in $s$ by term $t$.

We denote by $\overset{*}{\to}_{\mathcal{H}}$ the reflexive transitiv closure of $\to_{\mathcal{H}}$, and by $\overset{*}{\leftrightarrow}_{\mathcal{H}}$ the reflexive, symmetric, and transitive closure of $\to_{\mathcal{H}}$. Instead of $t \overset{*}{\leftrightarrow}_{\mathcal{H}} s$ we write $t \equiv_{\mathcal{H}} s$ (or simply $t \equiv s$) and say that $t$ and $s$ are *equivalent* (w.r.t. $\mathcal{H}$).

Following common terminology [4], $\to_{\mathcal{H}}$ is call *confluent* if for every $t, t_1, t_1$ with $t \overset{*}{\to}_{\mathcal{H}} t_1$ and $t \overset{*}{\to}_{\mathcal{H}} t_2$ there exists $t'$ such that $t_1 \overset{*}{\to}_{\mathcal{H}} t'$ and $t_2 \overset{*}{\to}_{\mathcal{H}} t'$. Moreover, $\to_{\mathcal{H}}$ is call *terminating* if there do not exist $t_0, t_1, \ldots$ such that $t_0 \overset{*}{\to}_{\mathcal{H}} t_1 \overset{*}{\to}_{\mathcal{H}} \cdots$. The reduction relation $\to_{\mathcal{H}}$ is called *convergent* if it is both confluent and terminating.

A term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ is called *reduced* or *irreducible* (w.r.t. $\mathcal{H}$) if $t \overset{*}{\to}_{\mathcal{H}} t'$ implies that $t = t'$ for all $t' \in \mathcal{T}(\Sigma, \mathcal{V})$. We also say that $t$ is *in normal form*.

We call $t'$ a *normal form of* $t$ (w.r.t. $\mathcal{H}$) if $t \overset{*}{\leftrightarrow}_{\mathcal{H}} t'$ and $t'$ is in normal form. If $t'$ is uniquely determined, we denote $t'$ by $t{\downarrow}$ and call it *the normal form of* $t$. For $\mathcal{K} \subseteq \mathcal{T}(\Sigma, \mathcal{V})$, we set $\mathcal{K}{\downarrow} = \{m{\downarrow} \mid m \in \mathcal{K}\}$ if all terms in $\mathcal{K}$ have uniquely determined normal forms. For the equational theories we consider in this paper, normal forms will always be uniquely determined.

The following is well-known (see, e.g., [4]):

**Lemma 1.** *If $\to_{\mathcal{H}}$ is convergent, then:*
1. *Every $t$ has a normal form w.r.t. $\to_{\mathcal{H}}$.*
2. *$t \equiv_{\mathcal{H}} s$ iff $t{\downarrow} = s{\downarrow}$ for every $t$ and $s$.*

It is easy to see that $\to_{\mathcal{H}}$ with $\mathcal{H}$ defined as in Example 2 is convergent. Hence, every term has a unique normal form w.r.t. $\mathcal{H}$. For example, with $m = \langle \mathsf{text}, \mathsf{sig}(\mathsf{sk}(A), \mathsf{text})\rangle$, we have that $\mathsf{sigcheck}(\pi_1(m), \pi_2(m), \mathsf{pk}(A)){\downarrow} = \mathsf{ok}$.

For an equational theorie $\mathcal{H}$ such that $\to_{\mathcal{H}}$ is convergent we call elements from $\mathcal{T}(\Sigma){\downarrow}$ *messages* and denote the set of messages by $\mathcal{M}$.

### 3.3 Deriving Messages

New messages can be derived from a given set of messages by iteratively applying cryptographic operations to the messages in the given set (and then normalizing the resulting messages). Here, operations are modeled as terms and applying an operation means substituting the variables.

A set $\mathcal{S}$ is called a set of *operations* if $\mathcal{S} \subseteq \mathcal{T}(\Sigma, \mathcal{V})$.

*Example 3 (ASW operations).* For the ASW protocol, we need operations that model pairing, projections, using a public key, checking a signature, signing and hashing:

$$\mathcal{S}_{ASW} = \{\langle x_1, x_2\rangle, \pi_1(x_1), \pi_2(x_1), \mathsf{sigcheck}(x_1, x_2, x_3), \mathsf{sig}(x_2, x_1), \mathsf{hash}(x_1)\}$$

Given a set $\mathcal{S}$ of operations, we first define $\mathcal{S}$-terms, which model that operations from $\mathcal{S}$ can be applied iteratively. Let $\mathcal{V}'$ be a set of variables disjoint from $\Sigma \cup \mathcal{V}$. The *set of $\mathcal{S}$-terms* (w.r.t. $\mathcal{V}'$) is the smallest subset of $\mathcal{T}(\Sigma, \mathcal{V}')$ satisfying the following three conditions:

- each $x \in \mathcal{V}'$ is an $\mathcal{S}$-term,
- each $t \in S \cap \mathcal{T}(\Sigma)$ is an $\mathcal{S}$-term, and
- for each $t \in S$ with $\mathcal{V}(t) = \{x_1, \ldots, x_n\}$ and for each choice $t_1, \ldots, t_n$ of $\mathcal{S}$-terms, the term $t[t_1/x_1, \ldots, t_n/x_n]$ is an $\mathcal{S}$-term.

Now, assume that we are given a set $\mathcal{K}$ of ground terms. If $t$ is an $\mathcal{S}$-term and $\sigma$ is a substitution of the variables in $t$ by elements of $\mathcal{K}$, we call $t\sigma$ a *$\mathcal{K}$-instance* of $t$.

Next, let $\mathcal{H}$ be some fixed equational theory for $\Sigma$. We define

$$d_{\mathcal{S}}(\mathcal{K}) = \{m{\downarrow} \mid m \text{ is an } (\mathcal{S} \cup \mathcal{K})\text{-term without variables}\}$$

to be the set of messages (in normal form) that can be derived from $\mathcal{K}$ using $\mathcal{S}$. We fix one variable $x' \in \mathcal{V}'$. An $\mathcal{S}$-term $N$ is called *simple* if $\mathcal{V}(N) \subseteq \{x'\}$, that is, if a variable occurs in $t$, then it must be $x'$.

Note that

$$d_{\mathcal{S}}(\mathcal{K}) = d_{\mathcal{S} \cup \mathcal{K}}(\emptyset). \tag{4}$$

The reason why we separate $\mathcal{S}$ and $\mathcal{K}$ is that $\mathcal{S}$ is typically fixed and $\mathcal{K}$ frequently changes. (An agent applies a fixed set of operations, defined by $\mathcal{S}$, to the messages in its current knowledge $\mathcal{K}$.)

As an example, consider the signature $\Sigma$ and the equational theory $\mathcal{H}$ as defined in Example 1 and 2, and let $\mathcal{K} = \{\langle \mathsf{text}, \mathsf{sig}(\mathsf{sk}(A), \mathsf{text})\rangle\}$ and $S = \{\mathsf{sig}(y, x), \pi_1(x), \pi_2(y), \mathsf{sk}(B)\}$. Then, $\mathsf{sig}(\mathsf{sk}(B), \mathsf{text}) \in d_S(\mathcal{K})$ since with the $\mathcal{S}$-term $s = \mathsf{sig}(z_2, \pi_1(z_1))$ and the substitution $\sigma = \{z_1 \mapsto \langle \mathsf{text}, \mathsf{sig}(\mathsf{sk}(A), \mathsf{text})\rangle, z_2 \mapsto \mathsf{sk}(B)\}$ we have that $(s\sigma){\downarrow} = \mathsf{sig}(\mathsf{sk}(B), \mathsf{text})$.

## 4 The Concurrent Protocol and Intruder Model

In this section, we introduce our protocol and intruder model. As mentioned in the introduction, unlike most other (Dolev-Yao based) models, our model captures real concurrent computation in the sense that at the same time many agents (adversary and principals) can be active and many messages can be sent to several agents. More precisely: every honest instance of a protocol is modeled by a seperate automaton with input and output ports; the intruder together with the dishonest instances is an automaton by itself; every output port is connected to at most one input port (of another instance); communication proceeds in rounds. In every round, every automaton does the following:

1. It reads the input on all its input ports.
2. Depending on the input read and its current state, it writes output on one or more of the output ports and updates its current state.
   Output written on a port connected to an input port will be read in the next round by the agent connected to this input port.

Our definition proceeds in two steps. We first define a general notion of concurrent system and then go over to cryptographic systems, which, in our context, will be called Dolev-Yao systems.

*A note on notation.* For sets $A$ and $B$ we denote by $A^B$ the set of all mappings from $B$ to $A$. Given $C \subseteq B$ and $f \in A^B$, we denote by $f|_C$ the restriction of $f$ to $C$.

### 4.1 Concurrent Systems

A concurrent system in our framework is made up of several components, which are automata provided with input and output ports for inter-component communication. Each such port can either carry a message from a given set $\mathcal{M}$ of messages or the special symbol '$\circ$' (no message). We use $\mathcal{M}_\circ$ to denote $\mathcal{M} \cup \{\circ\}$.

We note that a port will always only carry at most one message. This is w.l.o.g. since in every round a component can read all messages on its input port, and hence, after one round, messages on input ports can be removed. Also, a component only needs to write one message on the output port since this message can be read by only one component, and hence, instead of writing several messages on one output port in one round, the sending automaton can simply concatenate all messages it wishes to send on the port. In other words, it can simply send one long message.

We start by defining components of concurrent systems.

**Definition 1 (component of concurrent system).** *A* component *of a concurrent system over a set $\mathcal{M}$ of messages is a tuple*

$$\mathcal{A} = (S, \boldsymbol{In}, \boldsymbol{Out}, I, \Delta) \tag{5}$$

*where*

- *$S$ is a (possibly infinite) set of* states*,*
- ***In*** *is the set of* input ports*,*
- ***Out*** *is the set of* output ports *disjoint with* ***In****,*
- *$I \subseteq S$ is the set of* initial states*, and*
- *$\Delta \subseteq \mathcal{M}_\circ^{\boldsymbol{In}} \times S \times S \times \mathcal{M}_\circ^{\boldsymbol{Out}}$ is the* transition relation *such that for every $(m, s) \in \mathcal{M}_\circ^{\boldsymbol{In}} \times S$ there exist $s'$ and $m'$ with $(m, s, s', m') \in \Delta$.*

A transition $(m, s, s', m')$ is meant to model that if $\mathcal{A}$ is in state $s$ and reads the messages $m$ on its input ports, then it writes $m'$ on its output ports and goes into state $s'$. In other words, a transition describes a possible I/O step of a component.

Note that by definition a component is required to be complete, i.e., it can take a transition for every input it gets and every state it is in. This is a convenient requirement, which comes at no loss of generality, because every non-complete component could be made complete by adding a dummy state and appropriate transitions.

Several components can be composed to a concurrent system:

**Definition 2 (concurrent system).** *A* concurrent system *over a set $\mathcal{M}$ of messages is a finite family $\{\mathcal{A}_i\}_{i\in P}$ of components over $\mathcal{M}$ of the form $(S_i, \boldsymbol{In}_i, \boldsymbol{Out}_i, I_i, \Delta_i)$ such that for every $i$,*

- *$\boldsymbol{In}_i \cap \boldsymbol{In}_j = \emptyset$ for $i \neq j$, and*
- *$\boldsymbol{Out}_i \cap \boldsymbol{Out}_j = \emptyset$ for $i \neq j$.*

Hence, we require that different components have disjoint sets of input and output ports, respectively. It follows that every port $p$ belonging to one set $\mathbf{In}_i$ of input ports can belong to at most one set $\mathbf{Out}_j$ of output ports. This will allow component $\mathcal{A}_j$ to send messages to component $\mathcal{A}_i$ over the port $p$.

Given a concurrent system $\mathcal{G} = \{\mathcal{A}_i\}_{i\in P}$ as above, its set of input and output ports is determined by

$$\mathbf{In} = \bigcup_{i\in P} \mathbf{In}_i \ , \qquad\qquad \mathbf{Out} = \bigcup_{i\in P} \mathbf{Out}_i \ . \tag{6}$$

We set $\mathbf{P} = \mathbf{In} \cup \mathbf{Out}$. The state set and the initial state set of $\mathcal{G}$ are defined by

$$S = \prod_{i\in P} S_i \ , \tag{7}$$

$$I = \{s \in S \mid s(i) \in I_i \text{ for } i \in P\} \ . \tag{8}$$

A *concurrent transition* is a tuple of the form $(m, s, s', m')$ satisfying $(m|_{\mathbf{In}_i}, s(i), s'(i), m'|_{\mathbf{Out}_i}) \in \Delta_i$ for every $i \in P$. A *global state* of $\mathcal{G}$ is a pair $(m, s)$ with $m \in \mathcal{M}_\circ^{\mathbf{P}}$ and $s \in S$.

An *infinite $(m, s)$-computation* of $\mathcal{G}$ is an infinite sequence $\rho = m_0 s_0 m_1 s_1 \ldots$ of global states such that $(m_0, s_0) = (m, s)$ and for every $i$,

- $(m_i, s_i, s_{i+1}, m_{i+1})$ is a concurrent transition and
- $m_i(p) = \circ$ for every $p \in \mathbf{P} \setminus \mathbf{Out}$.

*Finite $(m, s)$-computations* are defined in the same way.

An infinite sequence $\rho = m_0 s_0 \ldots$ is called a *computation* if it is an $(m, s)$-computation for some $(m, s)$.

If $m$ is such that $m(p) = \circ$ for every $p \in \mathbf{P}$ and if $s \in I$, we call an infinite $(m, s)$-computation a *run* of $\mathcal{G}$, i.e., a run is a computation where the ports initially do not carry a message and the components are in one of their initial states. A finite prefix of a run is called a *run segment*.

A global state $(m, s)$ is called *reachable* if there is a run segment $\rho = m_0 s_0 m_1 s_1 \ldots m_{k-1} s_{k-1}$ such that $(m_{k-1}, s_{k-1}) = (m, s)$. Let $(m, s)$ and $(m', s')$ be global states. We call $(m', s')$ a *descendant of* $(m, s)$ if there is an $(m, s)$-computation $\rho = m_0 s_0 m_1 s_1 \ldots$ such that $(m', s') = (m_i, s_i)$ for some $i \geq 0$, in particular, $(m, s)$ is a descendent of $(m, s)$.

### 4.2 Dolev-Yao Systems

In our framework, a cryptographic system is a concurrent system where the components play certain roles and are connected in a certain fashion. In every such system, there is one special component which models the Dolev-Yao intruder and subsumes all executions of protocols by (completely) dishonest principals, and there are other components modeling executions of protocols by honest (or semi-honest) principals. More precisely, every such component models one instance of a protocol run by one honest principal (*honest instance*).

The components are connected by ports as follows: Every component modeling an honest instance has network and secure channel ports to all other instances (some of these ports might not be used by that instance). The network ports model network communication. Since, as usual, the Dolev-Yao intruder is assumed to have full control over the network, he is connected to the network ports of the honest instances. Conversely, honest instances are directly connected via secure channel ports. In particular, messages sent over these ports are not redirected through the adversary. However, if an honest instance is connect via secure channel ports with a dishonest principal, then this means that the Dolev-Yao intruder connects to these ports as dishonest principals are simulated by the Dolev-Yao intruder. Instead of connecting two honest principals directly through a secure channel, one could plug between two honest principals a secure channel component for more flexible scheduling. However, for simplicity and since this does not change our results (if secure channel components between honest principals are not controlled by the adversary), we choose direct secure channel links.

To define cryptographic systems formally, let $ALL$ be a set of principals, partitioned in a set $HON$ of *honest principals* and a set $DIS = ALL \setminus HON$ of *dishonest principals*. These sets determine the set $\mathbf{P}$ of the following ports where $\pi, \pi' \in ALL$ with $\{\pi, \pi'\} \cap HON \neq \emptyset$:

- $\mathrm{netin}^{\pi}_{\pi'}$, used by instance $\pi$ to write a message into the network which is intended for $\pi'$,
- $\mathrm{netout}^{\pi}_{\pi'}$, used by $\pi'$ to read messages from the network presumably from $\pi$,
- $\mathrm{sec}^{\pi}_{\pi'}$, used by $\pi$ as an output port to send a message over the secure channel to $\pi'$ by $\pi'$ as an input port to receive a message from $\pi$ over the secure channel.

As explained above, $\mathrm{netin}^{\pi}_{\pi'}$ will be an input port for the Dolev-Yao intruder, $\mathrm{netout}^{\pi'}_{\pi}$ will be an output port of the Dolev-Yao intruder, and $\mathrm{sec}^{\pi}_{\pi'}$ directly connects the instances $\pi$ and $\pi'$, in particular, it is not directed through the Dolev-Yao intruder.

Note that in the above definition we require $\{\pi, \pi'\} \cap HON \neq \emptyset$ since dishonest instances are simulated by the Dolev-Yao intruder, and hence, no ports are needed between dishonest instances.

We now define the intruder component, modeling the Dolev-Yao intruder. The definition of this component depends on the following parameters: the signature $\Sigma$, the set of operations $\mathcal{S} \subseteq \mathcal{T}(\Sigma, \mathcal{V})$ the intruder is able to perform, the semantics of the operations given by an equational theory $\mathcal{H}$, the finite set

$\mathcal{K} \subseteq \mathcal{M} = \mathcal{T}(\Sigma)$ (the intial intruder knowledge), and the set $P$ partitioned into honest and dishonest principals.

For ease in description, $\Sigma$, $\mathcal{S}$, and $\mathcal{H}$ will henceforth be understood to be present without being mentioned explicitly. Also, when we say that *ALL* is a *partitioned set of principals* we assume that a set *HON* has been fixed and that $DIS = ALL \setminus HON$.

**Definition 3 (intruder component).** *Let $\mathcal{K} \subseteq \mathcal{M}$ be a set of messages and ALL a partitoned set of principals. The* intruder component $\mathcal{A}_{\mathcal{I}}$ *for $\mathcal{K}$ and ALL is the tuple*

$$\mathcal{A}_{\mathcal{I}} = (S_{\mathcal{I}}, \boldsymbol{In}_{\mathcal{I}}, \boldsymbol{Out}_{\mathcal{I}}, \{\mathcal{K}\}, \Delta_{\mathcal{I}}) \tag{9}$$

*where the sets of the input and output ports are defined by*

$$\boldsymbol{In}_{\mathcal{I}} = \{netin^{\pi}_{\pi'} \mid \pi \in HON, \pi' \in ALL\} \cup \{sec^{\pi}_{\pi'} \mid \pi \in HON, \pi' \in DIS\} \tag{10}$$

$$\boldsymbol{Out}_{\mathcal{I}} = \{netout^{\pi}_{\pi'} \mid \pi' \in HON, \pi \in ALL\} \cup \{sec^{\pi}_{\pi'} \mid \pi \in DIS, \pi' \in HON\} \tag{11}$$

*and*

- *$S_{\mathcal{I}}$ is the set of all finite subsets of $\mathcal{M}$, and*
- *$\Delta_{\mathcal{I}}$ is defined by $(m, s, s', m') \in \Delta_{\mathcal{I}}$ iff*
  - *$s' = s \cup \{m(p) \mid m(p) \neq \circ, p \in \boldsymbol{In}_{\mathcal{I}}\}$ and*
  - *$m'(p) \in d_{\mathcal{S}}(s')$ for all $p \in \boldsymbol{Out}_{\mathcal{I}}$ with $m'(p) \neq \circ$.*

Given a partitioned set *ALL* of principals, an *honest instance component* for $\pi \in HON$ is a component with ports specified by

$$\mathbf{In} = \{\mathrm{netout}^{\pi'}_{\pi} \mid \pi' \in ALL\} \cup \{\mathrm{sec}^{\pi'}_{\pi} \mid \pi' \in ALL\} \ , \tag{12}$$

$$\mathbf{Out} = \{\mathrm{netin}^{\pi}_{\pi'} \mid \pi' \in ALL\} \cup \{\mathrm{sec}^{\pi}_{\pi'} \mid \pi' \in ALL\} \ . \tag{13}$$

Now we can define our notion of cryptographic system, also called a Dolev-Yao system:

**Definition 4 (Dolev-Yao system).** *Let $\mathcal{K} \subseteq \mathcal{M}$, ALL a partitioned set of principals with HON, and $\{\mathcal{A}_{\pi}\}_{\pi \in HON}$ a family of honest instance components.*

*The* Dolev-Yao system *determined by this is the concurrent system*

$$\{\mathcal{A}_{\pi}\}_{\pi \in HON \cup \{\mathcal{I}\}} \tag{14}$$

*where $\mathcal{A}_{\mathcal{I}}$ is the intruder component for $\mathcal{K}$ and ALL. We refer to such a Dolev-Yao system by $DY[\{\mathcal{A}_{\pi}\}_{\pi \in HON}, HON, DIS, \mathcal{K}]$; the set $\mathcal{S}$ of operations the intruder may use to derive new messages is understood from the context.*

We define $\mathcal{K}(m, s) = s(\mathcal{I}) \cup \{m(p) \mid m(p) \neq \circ, p \in \mathbf{In}_{\mathcal{I}}\}$ to be the knowledge of the intruder at state $(m, s)$. We say that the intruder can *deduce message $m'$ at state $(m, s)$* if $m' \in d_{\mathcal{S}}(\mathcal{K}(m, s))$.

### 4.3 Modeling the ASW Protocol

It is almost straightforward to turn the informal description of the ASW protocol into our framework. We demonstrate this by modeling an instance of Alice and describe how we model the TTP for the ASW protocol in detail.

To describe instance automata of honest participants we use two possible ways: pseudocode and transition table.

Our pseudocode is a list of items of the following form

```
<condition>
  <action-list>
```

The condition `<condition>` is a condition about the internal state of a participant together with conditions about messages pending on input ports for this participant. For example in

$\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$ and
$(\pi_1(\pi_1(\mathrm{netout}_A^B)) \neq me_1$ or
$\mathsf{sigcheck}(\pi_1(\mathrm{netout}_A^B), \pi_2(\mathrm{netout}_A^B), \mathsf{pk}(B)) \neq \mathsf{ok})$:

- $\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$
- $\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$
  $m_2 = \langle \mathsf{aborted}, me_1 \rangle$
  $ma_1 = \langle m_2, \mathsf{sig}(\mathsf{sk}(A), m_2) \rangle$
  $\mathrm{sec}_T^A = ma_1$

the condition checks if the local state of participant $A$ is $(i, \mathsf{pos\text{-}sent})$ (indicating that Alice has sent her promise of signature to Bob) and the message pending on channel $\mathrm{netout}_A^B$ is not $me_1$, i.e., is not of the form of a promise of signature from Bob. According to the protocol Alice has two possibilities in this situation i) waiting for the promise of signature from Bob to come and ii) sending an abort request to the TTP. Waiting for Bob's promise of signature is modelled by staying in state $(i, \mathsf{pos\text{-}sent})$ and sending an abort request to the TTP is divided in simpler steps, i.e.,

- $\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$ : change local state to $(i, \mathsf{abort\text{-}req\text{-}sent})$
  $m_2 = \langle \mathsf{aborted}, me_1 \rangle$ : construct abort request $ma_1$
  $ma_1 = \langle m_2, \mathsf{sig}(\mathsf{sk}(A), m_2) \rangle$
  $\mathrm{sec}_T^A = ma_1$ : send $ma_1$ over secure channel $\mathrm{sec}_T^A$

The semantics of a list

```
<condition-1>
  <action-list-1>
<condition-2>
  <action-list-2>
...
<condition-n>
  <action-list-n>
```

is that the list is gone through in a top down manner and for the first condition `<condition-i>` that is satisfied one action of the action list `<action-list-i>` is chosen nondeterministically to be performed. If no condition is satisfied, the state of the automaton does not change. It is straightforward to construct an instance automata from such a pseudocode description.

A simple and compact way to describe the behaviour of the TTP is by giving a kind of transition table, see Figure 2. According to the protocol the TTP maintains a database for the protocol instances for which she were invoked by one of the principals, i.e., one of the participants sent a request to her. The key for the database is a tuple the form $\langle \mathsf{text}, \mathsf{pk}(X), \mathsf{pk}(Y)\rangle$ where this tuple stands for the protocol instance where $X$ is the initiator, $Y$ is the responder, and text is the contractual text of the contract in consideration. For an instance of the protocol in which the TTP is involved she stores two things i) a status information for this instance and ii) a message that is needed for responding to later requests. The status information is needed to distinguish between the cases that the TTP has resolved a contract signing procedure by issuing a contract or has aborted a contract signing procedure.

The table is partitioned into two parts. The left part (first three columns) describes the conditions that must be true for the TTP to perform the actions described by the right part of the table.

If the TTP has not yet received a request for instance $\langle \mathsf{text}, \mathsf{pk}(X), \mathsf{pk}(Y)\rangle$, then the database has no entry for this instance, i.e., there is no status information for this instance in the database (indicated by $\perp$).

The columns 'from $A$' and 'from $B$' specify the kind of request the TTP is expecting from $A$ or $B$. Now we explain the abbreviations used in the table in more detail.

The first two letters of these abbreviations indicate which participant is in the initiator role and which participant is in the responder role. An abbreviation that starts with AB stands for an instance where A is in the initiator and B is in the responder role. The abbreviations in detail are:

| | |
|---|---|
| ABabort | abort request message sent by A |
| ABabort-ack | abort acknowledge message sent by the TTP |
| ABresA | resolve request message sent by A |
| ABresB | resolve request message sent by B |
| ABrepl-contract | replacement contract signed by the TTP |

Now, the first row of the table is read as: If on channel $\mathsf{sec}_T^A$ message ABabort is pending, on channel $\mathsf{sec}_T^B$ no request for the corresponding instance is received, and the status for this instance is $\perp$, then the new status for this instance is aborted, message ABabort-ack is sent to $A$ on channel $\mathsf{sec}_A^T$ and the abort acknowledge message ABabort-ack is stored for this instance. In the table we see (fourth row) that in this specification of the TTP resolve requests take priority over abort request when both types of request are pending. The value that is stored in the database is used to respond to later requests. For example, in the 8th row, describing the situation that the status for instance $\langle \mathsf{text}, \mathsf{pk}(A), \mathsf{pk}(B)\rangle$ is aborted and the TTP receives a resolve request from $B$. In this situation the

stored value in the database for this instance is ABabort-ack and this stored value is sent to $B$ as a response to his resolve request.

The TTP must handle requests for various instances. These instances are described by the same table with the names of the participants modified accordingly. In case the TTP receives requests at the same time for different instances, she simply looks up the appropriate action in the corresponding tables.

## 5 Balanced and Abuse-free Protocols

In this section, we present our formal definition of abuse-freeness, based on the notion of balance, which, in turn, is based on the notion of strategy.

To pave the way for our formal definitions, we start with informal descriptions of balance and abuse-freeness and some comments.

**Balance** (informal definition): A contract signing protocol is unbalanced for participant $A$ if at some point during the execution of the protocol (another) participant $B$ has both (i) a strategy to prevent $A$ from getting a valid contract and (ii) a strategy to obtain a valid contract.

As explained in the introduction, abuse-freeness will be formulated based on off-line attacks. The informal definition is the following:

**Abuse-freeness** (informal definition): A contract signing protocol is not abuse-free for a participant $A$ if at some point during the execution of the protocol another participant $B$ can convince an outside party $C$ that he is or was in a state unbalanced for $A$ and that he could (still) be in such a state.

The following subsections formalize this in detail.

### 5.1 Strategies

Strategies in the context of abuse-freeness need to be defined with respect to partial information, since participant $B$ will not necessarily know the global state of the entire protocol at any point of protocol execution. We formalize this using the notion of view functions.

**Definition 5 (view function).** *Let $\mathcal{G}$ be a concurrent system with set of ports $\boldsymbol{P}$ and state set $S$. A function with domain $(\mathcal{M}_\circ^{\boldsymbol{P}} \times S)^+$ is called a* view function *for $\mathcal{G}$.*

*When* view *is a view function and $\rho$ a run segment, we say that* view$(\rho)$ *is the* view of $\rho$ w. r. t. view.

Strategies in our context are—as mentioned above—based on partial information. In addition, they are carried out by a coalition of components of a given concurrent system.

Assume $\{\mathcal{A}_i\}_{i\in P}$ is a concurrent system as above. Any subset of $P$ is called a *coalition*. Given a coalition $J$, we write $\mathbf{Out}_J$ for $\bigcup_{j\in J} \mathbf{Out}_j$.

**Definition 6 (strategy).** *Let $\mathcal{G} = \{\mathcal{A}_i\}_{i\in P}$ be a concurrent system. Assume $J \subseteq P$ is a coalition of $\mathcal{G}$ and* view*: $(\mathcal{M}_\circ^P \times S)^+ \to W$ is a view function for $\mathcal{G}$.*

*A function*

$$\sigma\colon W \to \mathcal{M}_\circ^{\boldsymbol{Out}_J} \times \prod_{i\in J} S_i \tag{15}$$

*is a* view-strategy for $J$ *if it satisfies the following condition for every non-empty run segment $\rho = m_0 s_0 \dots m_l s_l$ of $\mathcal{G}$: if $\sigma(\mathsf{view}(\rho)) = (m, s)$, then for every $j \in J$*

$$(m_l|_{\boldsymbol{In}_j}, s_l(j), s(j), m|_{\boldsymbol{Out}_j}) \in \Delta_j \ , \tag{16}$$

*Let $(m, s) \in \mathcal{M}_\circ^P \times S_\mathcal{G}$ be a global state of $\mathcal{G}$. The* outcome *of $\mathcal{G}$ w. r. t. $\sigma$ is the set denoted $\mathsf{out}((m, s), \sigma)$ and defined to consist of all infinite $(m, s)$-computations $m_0 s_0 m_1 s_1 \dots$ satisfying*

$$\sigma(\mathsf{view}(m_0 s_0 \dots m_l s_l)) = (m_{l+1}|_{\boldsymbol{Out}_J}, s_{l+1}|_J) \tag{17}$$

*for every $l \geq 0$.*

### 5.2 Balanced Protocols

In our formal definition of balance, participant $B$ is modelled by a coalition and the two goals $B$ wants to achieve (e.g., prevent $A$ from getting a valid contract or obtain a valid contract) is defined as a path property, that is, by two sets of runs.

We start with a definition of path properties, adjusted to our context.

**Definition 7 (property).** *Let $\mathcal{G}$ be a concurrent system with set of ports $\boldsymbol{P}$ and state set $S$. A set $\varphi \subseteq (\mathcal{M}_\circ^P \times S)^\omega$ is called a* $\mathcal{G}$-property.

A situation in which we are interested in balance is always determined by a tuple $(I, \mathsf{view}, \varphi_1, \varphi_2)$ consisting of a coaltion $I$, a view function view and path properties $\varphi_1$ and $\varphi_2$. That is why we refer to such a tuple as a *balance specifier*, usually denoted by $\beta$.

**Definition 8 (balance).** *Let $\mathcal{G}$ be a concurrent system with index set $P$, $(m, s)$ a reachable state of $\mathcal{G}$, and $\beta = (I, \mathsf{view}, \varphi_1, \varphi_2)$ a balance specifier.*

*The state $(m, s)$ is* $\beta$-unbalanced *if there are view-strategies $\sigma_1$ and $\sigma_2$ for $I$ such that:*

1. *$\rho \in \varphi_1$ for every $\rho \in \mathsf{out}((m, s), \sigma_1)$, and*
2. *$\rho \in \varphi_2$ for every $\rho \in \mathsf{out}((m, s), \sigma_2)$.*

*The system $\mathcal{G}$ is* $\beta$-unbalanced *if there is a reachable state $(m, s)$ of $\mathcal{G}$ that is $\beta$-unbalanced.*

### 5.3 Tests

In our formalization of abuse-freeness, we use a specific but natural notion of tests that $C$ can make use of to verify that $B$ is in fact in the position he claims to be in. This notion of test is defined first, before we discuss abuse-freeness.

With regard to the expressiveness of tests, we note the following. Suppose a contract signing protocol is abuse-free with respect to a weak notion of test. Then it is also abuse-free with respect to a stronger notion of test, because $C$ has at least as much power as before to verify the evidence presented to him by $B$. On the contrary, suppose a protocol is not abuse-free with respect to a strong notion of test. Then it is neither abuse-free for a weaker notion of test. This means it is important to have different notions of test with various degrees of expressiveness.

**Definition 9 (atomic test).** *Let $\mathcal{X} \subseteq \mathcal{T}(\Sigma)\downarrow$. A pair $(M, M')$ of simple $(\mathcal{S} \cup \mathcal{X})$-terms is called an* atomic $\mathcal{X}$-test *or simply test if $\mathcal{X}$ is clear from the context. A message $m \in \mathcal{M}$* passes the test $(M, M')$*, denoted $m \models_{\mathcal{H}} (M, M')$, if $M[m/x] \equiv_{\mathcal{H}} M'[m/x]$. The message $m$* fails the test $(M, M')$ *if $m$ does not pass it.*

An ordinary test is a boolean combination of atomic tests:

**Definition 10 (boolean test).** *A* boolean test *is a boolean combination of atomic tests. A message $m$* passes *a boolean test $\theta$, denoted $m \models_{\mathcal{H}} \theta$, if $\theta$ evaluates to true when each atomic test is replaced by true if $m$ passes this test and by false otherwise.*

The strongest notion of test is the following.

**Definition 11 ($\omega$-test).** *An* $\omega$-test *is a boolean combination with possibly infinite disjunctions and conjunctions. A message $m$ passes an $\omega$-test $\theta$, denoted $m \models_{\mathcal{H}} \theta$, if $\theta$ evaluates to true when each atomic test is replaced by true if $m$ passes this test and by false otherwise.*

In the following, when we speak of a test, we mean a test of any of the three types. For a given test $\theta$, a state $(m, s)$ of a Dolev-Yao system $\mathcal{G}$ is called *$\theta$-possible* if there exists a message $m' \in d_{\mathcal{S}}(\mathcal{K}(m, s))$ such that $m' \models \theta$.

Clearly, $\omega$-tests are strictly stronger than boolean test, which, in turn, are strictly stronger than atomic tests. Consider, for instance, the case where $\Sigma = \{c, f, g\}$ with constant symbol $c$ and unary function symbols $f$ and $g$, $\mathcal{S} = \{c, f(x), g(x)\}$ and $\mathcal{H} = \emptyset$. Then it is easy to see that there is an $\omega$-test which defines exactly the set of messages of the form $f(f(f(\ldots f(c)\ldots)))$, which cannot be achieved by boolean tests. Similarly, the set $\{c, f(c)\}$ can be defined by a boolean test, but not by an atomic one.

### 5.4 Abuse-free protocols

As already explained earlier, when a protocol is considered abuse-free, then this means that from Charlie's point of view Bob has no way of convincing him that he is in an unbalanced state. That is, the property of being abuse-free is relative to the view that Charlie has of the protocol. Technically, such a view is determined by a Dolev-Yao system and a balance specifier. This motivates the following definition. A pair $(\mathcal{G}^e, \beta^e)$ consisting of a Dolev-Yao system $\mathcal{G}^e$ and a balance specifier $\beta^e$ is called an *external view (with respect to abuse-freeness)*.

The next definition puts everything together. A protocol is not abuse-free if there is a test which indicates unbalanced states as explained before:

**Definition 12 (abuse-freeness).** *Let $\mathcal{X} \subseteq \mathcal{M}$. An external view $(\mathcal{G}^e, \beta^e)$ is* $\mathcal{X}$-abusive *if there exists an $\mathcal{X}$-test $\theta$ such that the following two conditions are satisfied:*

1. *There exists a $\theta$-possible and $\beta$-unbalanced state in $\mathcal{G}^e$.*
2. *Each $\theta$-possible state $(m, s)$ of $\mathcal{G}^e$ is a descendant of a $\theta$-possible and $\beta$-unbalanced state in $\mathcal{G}^e$.*

*Such a test is called* $(\mathcal{G}^e, \beta^e)$-convincing. *The external view $(\mathcal{G}^e, \beta^e)$ is called* $\mathcal{X}$-abuse-free *if $\mathcal{G}^e$ it is not $\mathcal{X}$-abusive.*

## 6 ASW Protocol Analyzed

In this section, we analyse the ASW contract signing protocol in our concurrent setting. We show that it is unbalanced (in contrast to what is known from settings where concurrency is modeled by interleaving and the honest party is not optimistic) and not abuse-free. Moreover, we demonstrate that the security properties of the ASW protocol are sensitive to how the TTP handles messages that are received at the same time, a phenomenon that has not been observed previously. But note that independently of how the TTP is modeled no version turns out to be secure (abuse-free).

For our formal analysis of the ASW protocol, we will use the model developed in Section 4, see Examples 1, 2, and 3. That is, we assume $\Sigma = \Sigma_{ASW}$, $\mathcal{S} = \mathcal{S}_{ASW}$, and $\mathcal{H} = \mathcal{H}_{ASW}$.

### 6.1 ASW Protocol is not Balanced

First, recall that the ASW protocol is balanced when an interleaving model is adopted and the honest party is not optimistic (can be shown, for instance, like the results shown in [5] for the GJM protocol). Basically, the reason is that Alice can contact the TTP to abort the contract signing if Bob does not respond to the first message of Alice in the exchange protocol, and because of the interleaving assumption, Bob cannot make sure a resolve request from his part is handled by the TTP before Alice's abort request is.

By contrast, if we consider a concurrent setting and make the assumptions that (1) the message Bob (the intruder) sends arrive at least as fast as those of

Alice, and (2) the TTP handles a resolve request first when an abort request is pending at the same time, then we can argue (informally) that the protocol is unbalanced: Bob has (i) a strategy to prevent Alice from getting a valid contract, namely by simply doing nothing, and (ii) a strategy to resolve the contract signing after Alice has sent the first message of the exchange protocol, namely by sending a resolve request to the TTP. Even if Alice sends an abort request to the TTP at the same time, because of assumption (1) her request cannot reach the TTP before Bob's resolve request and with assumption (2) we know that Bobs resolve request takes priority over Alice's abort request if the two request arrive at the same time.

We analyse the situation in which (i) there is one protocol instance for each Alice (the initiator), Bob (the responder), and the TTP, (ii) Alice and the TTP are specified by the pseudo code given in Figures 1 and 2, respectively, and (iii) Bob is considered to be dishonest and modeled by the intruder. That is, we let $ALL = \{A, B, T\}$, $HON = \{A, T\}$, $\mathcal{K} = \{A, B, T, \mathsf{sk}(B), \mathsf{text}, \mathsf{initiator}, \mathsf{responder}, \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{pk}(T), \mathsf{ok}\}$, and $\mathcal{G}_{ASW} = \{\mathcal{A}_i\}_{i \in P}$ with $P = \{\mathcal{I}, A, T\}$ be the Dolev-Yao system induced.

Next, let $I = \{\mathcal{I}\}$. To formally define balance, we use tests to check whether a message has the form of a valid contract for Bob in the role of the responder and of a valid contract for Bob in the role of the initiator:

$$\theta^{\mathsf{contract}} = \theta^{\mathsf{contract}}_{\mathsf{initiator}} \vee \theta^{\mathsf{contract}}_{\mathsf{responder}} \ . \tag{18}$$

We will define $\theta^{\mathsf{contract}}_{\mathsf{initiator}}$ in more detail; the other test is analogous. The test $\theta^{\mathsf{contract}}_{\mathsf{initiator}}$ has to test if a message has the form of a standard contract or the form of a replacement contract. Thus, $\theta^{\mathsf{contract}}_{\mathsf{initiator}}$ is of the form

$$\theta^{\mathsf{contract}}_{\mathsf{initiator}} = \theta^{\mathsf{contract}}_{i,\mathsf{standard}} \vee \theta^{\mathsf{contract}}_{i,\mathsf{replacement}} \ . \tag{19}$$

The test $\theta^{\mathsf{contract}}_{i,\mathsf{standard}}$ is defined by

$$\begin{aligned}\theta^{\mathsf{contract}}_{i,\mathsf{standard}} =& (\mathsf{sigcheck}(\langle \mathsf{pk}(B), \mathsf{pk}(A), \mathsf{pk}(T), \mathsf{text}, \mathsf{hash}(\pi_2(x))\rangle, \pi_1(x), \mathsf{pk}(B)), \mathsf{ok}) \ \wedge \\ & (\mathsf{sigcheck}(\langle \pi_1(x), \mathsf{hash}(\pi_{222}(x))\rangle, \pi_{221(x)}, \mathsf{pk}(A)), \mathsf{ok}) \ .\end{aligned}$$

The other parts of the test $\theta^{\mathsf{contract}}$ can be defined in a similar way. Using these tests, we can define in a straightforward manner the path property $\varphi_{\mathcal{I}}$ to describe that the intruder does get a valid contract.

$$\varphi_{\mathcal{I}} = \{m_0 s_0 \cdots \in (\mathcal{M}^{\mathbf{P}}_{\circ} \times S)^{\omega} \mid \exists i \exists m (m \in d_S(\mathcal{K}(m_i, s_i)) \wedge m \models \theta^{\mathsf{contract}})\} \ .$$

The path property $\bar{\varphi}_A$ describes that Alice does not get a valid contract and is defined by

$$\bar{\varphi}_A = \{m_0 s_0 \cdots \in (\mathcal{M}^{\mathbf{P}}_{\circ} \times S)^{\omega} \mid \forall i (\mathsf{state}_A(s_i) \notin \{(i, \mathsf{contract}), (i, \mathsf{resolved})\})\}$$

where $\mathsf{state}_A(s_i)$ denotes the value of the variable $\mathsf{state}_A$ in state $s_i$ (see Figure 1).

We now define the balance specifier $\beta_{ASW} = (\{\mathcal{I}\}, \mathsf{view}_{\mathcal{I}}, \bar{\varphi}_A, \varphi_{\mathcal{I}})$ that describe being unbalanced for Alice.

We also assume that Bob's view of the system is limited to his own history, that is, we use the view function $\mathsf{view}_{\mathcal{I}}$ defined by $\mathsf{view}_{\mathcal{I}}(m_0 s_0 \dots m_l s_l) = m_0|_{\mathbf{In}_{\mathcal{I}} \cup \mathbf{Out}_{\mathcal{I}}} s_0(\mathcal{I}) \dots m_l|_{\mathbf{In}_{\mathcal{I}} \cup \mathbf{Out}_{\mathcal{I}}} s_l(\mathcal{I})$.

The following proposition states that the state of $\mathcal{G}_{ASW}$ after Alice sent her first message of the exchange protocol is $\beta_{ASW}$-unbalanced. The proof of this proposition corresponds exactly to the informal argumentation above that describes why the ASW protocol is unbalanced for Alice.

**Lemma 2.** *Let $m_0 s_0 \dots m_k s_k$ be a run segment of $\mathcal{G}_{ASW}$ and assume, for all $i < k$, that $m_i(netin_B^A) = \circ$ as well as $m_k(netin_B^A) \neq \circ$. Then $(m_k, s_k)$ is $\beta_{ASW}$-unbalanced.*

The above condition is satisfied in a run if Alice just sent her first message. Obviously, there are runs in which such a state exist. Thus, there is a reachable state of $\mathcal{G}_{ASW}$ that is $\beta_{ASW}$-unbalanced which shows the following theorem

**Theorem 1 (ASW is initiator unbalanced).** *The Dolev-Yao system $\mathcal{G}_{ASW}$ is $\beta_{ASW}$-unbalanced.*

Note that the initial state of $\mathcal{G}_{ASW}$ is not $\beta_{ASW}$-unbalanced because the intruder cannot force Alice to participate in the protocol.

Our proofs crucially relies on the fact that the TTP serves a resolve request before an abort request when both are pending at the same time. However, we can also show that if the TTP reverses its resolution strategy with respect to competing requests, that is, if she first serves an abort request and then a resolve request, then the protocol is unbalanced for the responder: Once the (honest) responder sent his first message, the (dishonest) initiator can send an abort to the TTP. Even if the responder sends a resolve right after he sent his first message, the two messages (abort from the initiator and resolve from the responder) would reach the TTP at the same time, and hence, the TTP would abort. Conversely, the initiator, once he got the first message from the responder, has obviously a strategy to obtain a valid contract.

More precisely, for a corresponding Dolev-Yao system $\mathcal{G}'_{ASW}$ and appropriate properties $\varphi'_{\mathcal{I}}$ and $\bar{\varphi}_B$, we have the following theorem where $\beta'_{ASW}$ denotes the corresponding balance specifier:

**Theorem 2 (ASW is responder unbalanced).** *The Dolev-Yao system $\mathcal{G}'_{ASW}$ is $\beta'_{ASW}$-unbalanced.*

Even if the TTP resolved competing requests randomly, the protocol would be unbalanced to a serious extent, because Alice and Bob would have strategies that would be successful with high probability (1/2).

### 6.2 ASW Protocol is not Abuse-free

For abuse-freeness, we imagine that Charlie assumes that there is only one instance of the ASW protocol running, but that he does not know whether Alice is

the initiator or responder, which is a realistic assumption. Formally, we replace $\mathcal{A}_A$ by a variant of it, denoted $\mathcal{A}_{A'}$, which in the beginning decides whether it wants to play the role of the initiator or the responder and then sends a corresponding message to Bob, see pseudocode description in Figures 3 and 4. We set $\mathcal{G}^e_{ASW} = \mathrm{DY}[\{\mathcal{A}_i\}_{i\in\{A',T\}}, \{A', T\}, \{B\}, \mathcal{K}]$ with $\mathcal{K}$ as above, $\beta^e_{ASW} = (\{\mathcal{I}^e\}, \mathsf{view}_{\mathcal{I}^e}, \bar{\varphi}_A, \varphi_{\mathcal{I}^e})$, and $\mathcal{X} = \{A, B, C, T, \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{pk}(T), \mathsf{pk}(C), \mathsf{sk}(C), \mathsf{text}, \mathsf{ok}\}$. Here, $\mathcal{I}^e$ denotes the intruder of $\mathcal{G}^e_{ASW}$. We also consider the situation where $T$ is replaced by $T'$ which gives priority to abort over resolve. The corresponding external view is denoted by $(\mathcal{G}'^e_{ASW}, \beta'^e_{ASW})$. We prove:

**Theorem 3 (ASW not abuse-free).** *The external views $(\mathcal{G}^e_{ASW}, \beta^e_{ASW})$ and $(\mathcal{G}'^e_{ASW}, \beta'^e_{ASW})$ are $\mathcal{X}$-abusive.*

In order to prove this theorem we need the following lemma; the proof of this lemma is straightforward.

**Lemma 3.** *Let $\mathcal{K} \subseteq \mathcal{T}(\Sigma)\downarrow$ be a set of ground terms such that $\mathsf{sk}(A) \notin d_{\mathcal{S}_{ASW}}(\mathcal{K} \cup \mathcal{C})$. Then for all $m \in d_{\mathcal{S}_{ASW}}(\mathcal{K} \cup \mathcal{C})$ where $m\downarrow = \mathsf{sig}(\mathsf{sk}(A), m')$ we have that $m\downarrow \in \mathit{Sub}(\mathcal{K}\downarrow)$.*

Now we can give the proof of Theorem 3.

**Proof:** We first show that the external view $(\mathcal{G}^e_{ASW}, \beta^e_{ASW})$ are $\mathcal{X}$-abusive. According to Definition 12 we have to show that there is a $(\mathcal{G}^e_{ASW}, \beta^e_{ASW})$-convincing $\mathcal{X}$-test $\theta$ and a reachable state $(m, s)$ of $\mathcal{G}^e_{ASW}$ such that the intruder can deduce a message $m'$ in $(m, s)$ that passes $\theta$.

Define $\theta$ by $\theta = \theta_1 \wedge \theta_2$, where

$$\begin{aligned}\theta_1 = {} & (\pi_1(\pi_1(x)), \mathsf{pk}(A)) \wedge (\pi_1(\pi_2(\pi_1(x))), \mathsf{pk}(B)) \wedge \\ & (\pi_1(\pi_2(\pi_2(\pi_1(x)))), \mathsf{pk}(T)) \wedge (\pi_1(\pi_2(\pi_2(\pi_2(\pi_1(x))))), \mathsf{text})\end{aligned}$$

and $\theta_2 = (\mathsf{sigcheck}(\pi_1(x), \pi_2(x), \mathsf{pk}(A)), \mathsf{ok})$. By $\theta_1$ Charlie is testing if Alice is in the initiator role and by $\theta_2$ he is testing if Alice has already sent the first message of the exchange protocol.

We first have to show that $\theta$ is a $(\mathcal{G}^e_{ASW}, \beta^e_{ASW})$-convincing $\mathcal{X}$-test. It is easy to see that a message $m' \in \mathcal{T}(\Sigma)\downarrow$ that passes $\theta$ is of the form $m' = \langle m'', \mathsf{sig}(\mathsf{sk}(A), m'')\rangle$, where $m''$ has the form $m'' = \langle \mathsf{pk}(A), \langle \mathsf{pk}(B), \langle \mathsf{pk}(T), \langle \mathsf{text}, m'''\rangle\rangle\rangle\rangle$.

Since $m'$ contains a signature from Alice and $\mathcal{K}_{\mathcal{I}}\downarrow$ does not contain $\mathsf{sk}(A)$ as a subterm we can conclude using Lemma 3 that Alice must have sent her first message of the exchange protocol.

Since $m'$ passes $\theta_2$ we know that Alice must be in the initiator role. So using Proposition 2 we know that a $\theta$-possible state $(m, s)$ of $\mathcal{G}^e_{ASW}$ is $\beta^e_{ASW}$-unbalanced or a descendant of an $\beta^e_{ASW}$-unbalanced state of $\mathcal{G}^e_{ASW}$.

Now we have to show that there is a reachable state $(m, s)$ in $\mathcal{G}^e_{ASW}$ such that the intruder can deduce a message in $(m, s)$ that passes $\theta$. The first message Alice sends in the exchange protocol to the intruder is of the form $m' = \langle m'', \mathsf{sig}(\mathsf{sk}(A), m'')$, where $m''$ has the form $m'' = \langle \mathsf{pk}(A), \langle \mathsf{pk}(B), \langle \mathsf{pk}(T), \langle \mathsf{text},$

$m'''\rangle\rangle\rangle\rangle$. Since $m'$ obviously passes $\theta$ the intruder is able to deduce a message that passes $\theta$ after Alice has sent her first message to the intruder.

So $(\mathcal{G}^e_{ASW}, \beta^e_{ASW})$ is $\mathcal{X}$-abusive.

Note that if Charlie would have performed $\theta_2$ on the message shown by the intruder only, he would not be convinced that Alice is in the initiator role, so $\theta_2$ is not a $(\mathcal{G}^e_{ASW}, \beta^e_{ASW})$-convincing $\mathcal{X}$-test.

The proof of the fact that the external view $(\mathcal{G}'^e_{ASW}, \beta'^e_{ASW})$ is $\mathcal{X}$-abusive is similar to the proof that $(\mathcal{G}'^e_{ASW}, \beta'^e_{ASW})$ is $\mathcal{X}$-abusive. The main difference is that we have to provide a test that checks whether a message has the form of the first message sent by the responder, i.e., his promise of signature instead of the test given above. □

## 7 GJM Protocol Analyzed

In this section we will analyse the GJM contract signing protocol. We will show that this protocol run in a concurrent setting is unbalanced but abuse-free.

The signature that we consider is

$$\begin{aligned}\Sigma_{GJM} = \{&\mathsf{sig}(\cdot,\cdot,\cdot), \mathsf{sigcheck}(\cdot,\cdot,\cdot), \mathsf{pk}(\cdot), \mathsf{sk}(\cdot), \langle\cdot,\cdot\rangle, \pi_1(\cdot), \pi_2(\cdot),\\ &\mathsf{fake}(\cdot,\cdot,\cdot,\cdot,\cdot), \mathsf{pcs}(\cdot,\cdot,\cdot,\cdot,\cdot), \mathsf{pcsver}(\cdot,\cdot,\cdot,\cdot,\cdot),\\ &\mathsf{sconvert}(\cdot,\cdot,\cdot), \mathsf{tpconvert}(\cdot,\cdot,\cdot), \mathsf{sver}(\cdot,\cdot,\cdot,\cdot), \mathsf{tpver}(\cdot,\cdot,\cdot,\cdot),\\ &A, B, T, \mathsf{text}, \mathsf{ok}, \mathsf{pcsok}, \mathsf{sok}, \mathsf{tpok}, \mathsf{initiator}, \mathsf{responder}, \mathsf{aborted}\} \cup \mathcal{C} \cup \mathcal{R}\end{aligned}$$

where $\mathcal{R}$ is a set of constants that represent the random coins used by different participants in their computations. The set $\mathcal{R}$ is the disjoint union of $\mathcal{R}_A$, $\mathcal{R}_{TTP}$, $\mathcal{R}_B$, and $\mathcal{R}_{\mathcal{I}}$ where $\mathcal{R}_A$ repreent the randomn coins used by Alice, $\mathcal{R}_B$ are those for Bob, $\mathcal{R}_{TTP}$ the ones for the TTP, and $\mathcal{R}_{\mathcal{I}}$ are the randomn coins for the intruder. Let $\mathcal{K}_{\mathcal{I}} = \mathcal{K} \cup \mathcal{C} \cup \mathcal{R}_{\mathcal{I}}$.

A term of the form $\mathsf{pcs}(u, \mathsf{sk}(x), w, \mathsf{pk}(y), \mathsf{pk}(z))$ stands for a PCS computed by $x$ (with $\mathsf{sk}(x)$) involving the text $w$, the party $y$, and the TTP $z$ while $u$ models the random coins used to compute the PCS. Everbody can verify the PCS with the public keys involved (identity (23)), but cannot determine whether the PCS was computed by $x$ or $y$ (identity (24)): instead of $x$ computing the "real" PCS, $y$ could have computed a "fake" PCS which would also pass the verification with $\mathsf{pcsver}$. Using $\mathsf{sconvert}$ and $\mathsf{tpconvert}$, see (25) and (26), a "real" PCS can be converted by $x$ and the TTP $z$, respectively, into a universally verifiable signature (verifiable by everone who possess $\mathsf{pk}(x)$ and $\mathsf{pk}(z)$). We model the third party-accountable version here: One can tell whether $x$ or the TTP converted the PCS. (We can easily also model the TTP-invisible version. However, it does not effect our results.) The equational theory $\mathcal{H}_{GJM}$ that models the above semantics along

with some obvious rules for projection and signature verification is given below.

$$\pi_1(\langle x, y\rangle) = x, \quad (20)$$
$$\pi_2(\langle x, y\rangle) = y, \quad (21)$$
$$\mathsf{sigcheck}(x, \mathsf{sig}(u, \mathsf{sk}(y), x), \mathsf{pk}(y)) = \mathsf{ok}, \quad (22)$$
$$\mathsf{pcsver}(w, \mathsf{pk}(x), \mathsf{pk}(y), \mathsf{pk}(z), \mathsf{pcs}(u, \mathsf{sk}(x), w, \mathsf{pk}(y), \mathsf{pk}(z))) = \mathsf{pcsok}, \quad (23)$$
$$\mathsf{pcsver}(w, \mathsf{pk}(x), \mathsf{pk}(y), \mathsf{pk}(z), \mathsf{fake}(u, \mathsf{sk}(y), w, \mathsf{pk}(x), \mathsf{pk}(z))) = \mathsf{pcsok}, \quad (24)$$
$$\mathsf{sver}(w, \mathsf{pk}(x), \mathsf{pk}(z), \mathsf{sconvert}(u, \mathsf{sk}(x), \mathsf{pcs}(v, \mathsf{sk}(x), w, \mathsf{pk}(y), \mathsf{pk}(z)))) = \mathsf{sok}, \quad (25)$$
$$\mathsf{tpver}(w, \mathsf{pk}(x), \mathsf{pk}(z), \mathsf{tpconvert}(u, \mathsf{sk}(z), \mathsf{pcs}(v, \mathsf{sk}(x), w, \mathsf{pk}(y), \mathsf{pk}(z)))) = \mathsf{tpok}. \quad (26)$$

The set $\mathcal{S}_{GJM}$ of operations available is defined by

$$\begin{aligned}\mathcal{S}_{GJM} = \{&\langle x_1, x_2\rangle, \pi_1(x_1), \pi_2(x_1),\\ &\mathsf{sig}(x_1, x_2, x_3), \mathsf{pcs}(x_1, x_2, x_3, x_4, x_5), \mathsf{fake}(x_1, x_2, x_3, x_4, x_5),\\ &\mathsf{sconvert}(x_1, x_2, x_3), \mathsf{tpconvert}(x_1, x_2, x_3),\\ &\mathsf{sigcheck}(x_1, x_2, x_3),\\ &\mathsf{sver}(x_1, x_2, x_3, x_4), \mathsf{tpver}(x_1, x_2, x_3, x_4), \mathsf{pcsver}(x_1, x_2, x_3, x_4, x_5)\}\end{aligned}$$

We set $\Sigma = \Sigma_{GJM}$, $\mathcal{S} = \mathcal{S}_{GJM}$, and $\mathcal{H} = \mathcal{H}_{GJM}$.

### 7.1 GJM Protocol is not Balanced

We analyse the following situation. We have one protocol instance of Alice, Bob and the TTP, where Alice and the TTP are specified by the pseudo code given in Figures 5 and 6, respectively. Bob is considered to be dishonest and will be modeled by the intruder.

We will now describe the formal setting in which we analyse the GJM protocol and state that in this setting the GJM protocol is unbalanced for Alice. That is, we let $ALL = \{A, B, T\}$, $HON = \{A, T\}$, $DIS = \{B\}$, $\mathcal{K} = \{A, B, T, \mathsf{sk}(B), \mathsf{text}, \mathsf{initiator}, \mathsf{responder}, \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{pk}(T), \mathsf{ok}\}$, and $\mathcal{G}_{GJM} = \mathrm{DY}[\{\mathcal{A}_i\}_{i\in HON}, HON, DIS, \mathcal{K}]$ be the Dolev-Yao system induced. As mentioned, the components for $A$ and $T$ are depicted in Figure 5 and 6. Note that $T$ gives priority to resolve if received at the same time as an abort. Let $\mathcal{I}$ denote the component corresponding to $B$ in $\mathcal{G}_{GJM}$.

Let $I = \{\mathcal{I}\}$. To formally define balance, we use tests to check whether a message has the form of a valid contract for Bob in the role of the responder and of a valid contract for Bob in the role of the initiator:

$$\theta^{\mathsf{contract}} = \theta^{\mathsf{contract}}_{\mathsf{initiator}} \vee \theta^{\mathsf{contract}}_{\mathsf{responder}} \ . \quad (27)$$

The tests $\theta^{\mathsf{contract}}_{\mathsf{initiator}}$ and $\theta^{\mathsf{contract}}_{\mathsf{responder}}$ can be defined in the same way as for the ASW-protocol, see Section 6. Using these tests, we can define in a straightforward manner the path property $\varphi_{\mathcal{I}}$ to describe that the intruder does get a valid contract.

$$\varphi_{\mathcal{I}} = \{m_0 s_0 \cdots \in (\mathcal{M}^{\mathbf{P}}_{\circ} \times S)^{\omega} \mid \exists i \exists m (m \in d_{\mathcal{S}}(\mathcal{K}(m_i, s_i)) \wedge m \models \theta^{\mathsf{contract}})\} \ .$$

The path property $\bar{\varphi}_A$ describes that Alice does not get a valid contract and is defined by

$$\bar{\varphi}_A = \{m_0 s_0 \cdots \in (\mathcal{M}_\circ^{\mathbf{P}} \times S)^\omega \mid \forall i(\mathsf{state}_A(s_i) \notin \{(i, \mathsf{contract}), (i, \mathsf{resolved})\})\}$$

where $\mathsf{state}_A(s_i)$ denotes the value of the variable $\mathsf{state}_A$ in state $s_i$ (see Figure 5).

Bob's view of the system is limited to his own history, that is, we use the view function $\mathsf{view}_\mathcal{I}$ defined by

$$\mathsf{view}_\mathcal{I}(m_0 s_0 \ldots m_l s_l) = m_0|_{\mathbf{In}_\mathcal{I} \cup \mathbf{Out}_\mathcal{I}} s_0(\mathcal{I}) \ldots m_l|_{\mathbf{In}_\mathcal{I} \cup \mathbf{Out}_\mathcal{I}} s_l(\mathcal{I}).$$

We now define the balance specifier $\beta_{GJM}$ to be $(\{\mathcal{I}\}, \mathsf{view}_\mathcal{I}, \bar{\varphi}_A, \varphi_\mathcal{I})$.

Lemma 4 characterizes the states of $\mathcal{G}_{GJM}$ that are $\beta_{GJM}$-unbalanced. Intuitively, the lemma states that the state of $\mathcal{G}_{GJM}$ after Alice has sent her promise of signature to the intruder is $\beta_{GJM}$-unbalanced.

Before we state this and other lemmas, we introduce abbreviations for messages of the GJM-protocol to increase readability.

First, we consider messages of the protocol instance where Alice is in the initiator role. The first message Alice sends as initiator in the exchange protocol is abbreviated by $\mathsf{ABposA}_r$, i.e., $\mathsf{ABposA}_r = \mathsf{pcs}(r, \mathsf{sk}(A), \langle \mathsf{c}, 1\rangle, \mathsf{pk}(B), \mathsf{pk}(T))$ for some randomness $r \in \mathcal{T}(\Sigma)\downarrow$. In the following abbreviations of messages subscripts allways denote randomnesses and these subscripts are read as 'for some randomness terms from $\mathcal{T}(\Sigma)\downarrow$'. The promise of signature from Bob in this instance is written as $\mathsf{ABposB}_r$, i.e., $\mathsf{ABposB}_r = \mathsf{pcs}(r, \mathsf{sk}(B), \langle \mathsf{c}, 2\rangle, \mathsf{pk}(A), \mathsf{pk}(T))$. The abort request message from Alice is abbreviated by $\mathsf{ABabort}_r$, i.e., $\mathsf{ABabort}_r = \mathsf{sig}(r, \mathsf{sk}(A), \langle \mathsf{c}, \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{abort}\rangle)$. The abort acknowledge message of the TTP is abbreviated by $\mathsf{ABabort\text{-}ack}_{r,r'}$, i.e.,

$$\mathsf{ABabort\text{-}ack}_{r,r'} = \mathsf{sig}(r, \mathsf{sk}(T), \mathsf{sig}(r', \mathsf{sk}(A), \langle \mathsf{c}, \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{abort}\rangle)).$$

The resolve request message of Alice and Bob are abbreviated by $\mathsf{ABresA}_{r,r',r''}$ and $\mathsf{ABresB}_{r,r',r''}$, respectively, i.e.,

$$\mathsf{ABresA}_{r,r',r''} = \langle \mathsf{sconvert}(r, \mathsf{sk}(A), \mathsf{pcs}(r', \mathsf{sk}(A), \langle \mathsf{c}, 1\rangle, \mathsf{pk}(B), \mathsf{pk}(T))), \mathsf{ABposB}_{r''}\rangle$$

and

$$\mathsf{ABresB}_{r,r',r''} = \langle \mathsf{ABposA}_r, \mathsf{sconvert}(r', \mathsf{sk}(B), \mathsf{pcs}(r'', \mathsf{sk}(B), \langle \mathsf{c}, 2\rangle, \mathsf{pk}(A), \mathsf{pk}(T)))\rangle.$$

The messages of the protocol instance where Alice is in the responder role are defined symmetrically. The first message Bob sends as initiator in the exchange protocol is abbreviated by $\mathsf{BAposB}_r$, i.e.,

$$\mathsf{BAposB}_r = \mathsf{pcs}(r, \mathsf{sk}(B), \langle \mathsf{c}, 1\rangle, \mathsf{pk}(A), \mathsf{pk}(T)).$$

The promise of signature from Alice is written as $\mathsf{BAposA}_r$, i.e.,

$$\mathsf{BAposA}_r = \mathsf{pcs}(r, \mathsf{sk}(A), \langle \mathsf{c}, 2\rangle, \mathsf{pk}(B), \mathsf{pk}(T)).$$

The abort request message from Bob is abbreviated by $\mathsf{BAabort}_r$, i.e.,

$$\mathsf{BAabort}_r = \mathsf{sig}(r, \mathsf{sk}(B), \langle \mathsf{c}, \mathsf{pk}(B), \mathsf{pk}(A), \mathsf{abort}\rangle).$$

The abort acknowledge message of the TTP is abbreviated by $\mathsf{BAabort\text{-}ack}_{r,r'}$, i.e.,

$$\mathsf{BAabort\text{-}ack}_{r,r'} = \mathsf{sig}(r, \mathsf{sk}(T), \mathsf{sig}(r', \mathsf{sk}(B), \langle \mathsf{c}, \mathsf{pk}(B), \mathsf{pk}(A), \mathsf{abort}\rangle)).$$

The resolve request message from Bob and Alice are abbreviated by $\mathsf{BAresB}_{r,r',r''}$ and $\mathsf{BAresA}_{r,r',r''}$, respectively, i.e.,

$$\mathsf{BAresB}_{r,r',r''} = \langle \mathsf{sconvert}(r, \mathsf{sk}(B), \mathsf{pcs}(r', \mathsf{sk}(B), \langle \mathsf{c}, 1\rangle, \mathsf{pk}(A), \mathsf{pk}(T))), \mathsf{BAposA}_{r''}\rangle$$

and

$$\mathsf{BAresA}_{r,r',r''} = \langle \mathsf{BAposB}_r, \mathsf{sconvert}(r', \mathsf{sk}(A), \mathsf{pcs}(r'', \mathsf{sk}(A), \langle \mathsf{c}, 2\rangle, \mathsf{pk}(B), \mathsf{pk}(T)))\rangle.$$

**Lemma 4.** *Let $(m, s)$ be a reachable state of $\mathcal{G}_{GJM}$. Then the following two conditions are equivalent:*

*1) $(m, s)$ is $\beta_{GJM}$-unbalanced.*
*2) i) $\mathsf{state}_A(s) = (i, \mathsf{pos\text{-}sent})$,*
*ii) $m(netout_A^B) \neq \mathsf{ABposB}_r$ for all $r \in \mathcal{T}(\Sigma)\downarrow$, and*
*iii) there is a run segment $m_0 s_0 \dots m_k s_k$ such that $(m_k, s_k) = (m, s)$ and for all $i \leq k$ and $r \in \mathcal{T}(\Sigma)\downarrow$ we have $m_i(sec_T^B) \neq \mathsf{ABresB}_r$.*

To prove Lemma 4 we need some technical lemmas. The proofs are very similar to the proof of Lemma 12 and are therefore omitted. Intuitively, the following lemma states that without knowing the secret key $\mathsf{sk}(A)$ of Alice one cannot derive a signature from Alice on a message $m$ unless the signature is present already.

**Lemma 5.** *Let $K \subseteq \mathcal{T}(\Sigma)\downarrow$ be $\{\mathsf{sk}(A)\}$-free. Let $r, m \in \mathcal{T}(\Sigma)\downarrow$. Then, the following conditions are equivalent:*

*1) $\mathsf{sig}(r, \mathsf{sk}(A), m) \notin d_{\mathcal{S}}(K)$.*
*2) For all $M \in K$ and $p \in Pos(M)$ such that $M|_p = \mathsf{sig}(r, \mathsf{sk}(A), m)$ there is a proper prefix $p'$ of $p$ such that $\mathsf{root}(M|_{p'}) \neq \langle,\rangle$.*

Lemma 6 states that without knowing the secret key $\mathsf{sk}(A)$ of Alice one cannot convert a $\mathsf{pcs}$-signature from Alice on a message $m$ to a universally verifiable signature on $m$ unless this converted signature is present already.

**Lemma 6.** *Let $K \subseteq \mathcal{T}(\Sigma)\downarrow$ be $\{\mathsf{sk}(A)\}$-free. Let $r, r', m \in \mathcal{T}(\Sigma)\downarrow$. Then, the following conditions are equivalent:*

*1) $\mathsf{sconvert}(r, \mathsf{sk}(A), \mathsf{pcs}(r', \mathsf{sk}(A), m, \mathsf{pk}(B), \mathsf{pk}(T))) \notin d_{\mathcal{S}}(K)$.*
*2) For all $M \in K$ and $p \in Pos(M)$ such that*

$$M|_p = \mathsf{sconvert}(r, \mathsf{sk}(A), \mathsf{pcs}(r', \mathsf{sk}(A), m, \mathsf{pk}(B), \mathsf{pk}(T)))$$

*there is a proper prefix $p'$ of $p$ such that $\mathsf{root}(M|_{p'}) \neq \langle,\rangle$.*

Lemma 7 is the corresponding lemma to 6 for the conversion of a pcs-signed message by the TTP . That is, without knowing the secret key $\mathsf{sk}(T)$ of the TTP one cannot convert a pcs-signature from Alice (where $T$ is the corresponding TTP) on a message $m$ to a universally verifiable signature on $m$ unless this converted signature is present already.

**Lemma 7.** *Let $K \subseteq \mathcal{T}(\Sigma)\downarrow$ be $\{\mathsf{sk}(T)\}$-free. Let $r, r', m \in \mathcal{T}(\Sigma)\downarrow$. Then, the following conditions are equivalent:*

*1) $\mathsf{tpconvert}(r, \mathsf{sk}(T), \mathsf{pcs}(r', \mathsf{sk}(A), m, \mathsf{pk}(B), \mathsf{pk}(T))) \notin d_{\mathcal{S}}(K)$.*
*2) For all $M \in K$ and $p \in Pos(M)$ such that*

$$M|_p = \mathsf{tpconvert}(r, \mathsf{sk}(T), \mathsf{pcs}(r', \mathsf{sk}(A), m, \mathsf{pk}(B), \mathsf{pk}(T)))$$

*there is a proper prefix $p'$ of $p$ such that $\mathsf{root}(M|_{p'}) \neq \langle , \rangle$ and*

Now we can prove Lemma 4.

*Proof:* In the protocol $\mathcal{G}_{GJM}$ Bob is modelled by the intruder. Therefore, in the following we write 'Bob' instead of 'the intruder'.

**1)⇒2):** We prove this implication by contraposition. First we show that if $\mathsf{state}_A(s) \neq (i, \mathsf{pos\text{-}sent})$, then $(m, s)$ is $\beta_{GJM}$-balanced. We distinguish between the possible values of $\mathsf{state}_A(s)$:

- $\mathsf{state}_A(s) = 0$: Obviously, Bob cannot force Alice to participate in the protocol. In particular, Alice can decide to stay in state $\mathsf{state}_A(s) = 0$. In such run Alice neither sends a promise of signature of the form $\mathsf{ABposA}_r$ ($r \in \mathcal{T}(\mathsf{Signature}_)\downarrow$) nor a signature $\mathsf{sconvert}(r', \mathsf{sk}(A), \mathsf{ABposA}_r)$ ($r, r' \in \mathcal{T}(\mathsf{Signature}_)\downarrow$) to Bob. By Lemma 5 and 6 we know that Bob neither can derive Alices' promise of signature nor her signature from his knowledge $\mathcal{K}$. Thus, he neither will get Alices' signature nor he can use the TTP to resolve the contract signing. (Note that Bob needs Alices' promise of signature $\mathsf{ABposA}_r$ for some $r \in \mathcal{T}(\Sigma)\downarrow$ to send a valid resolve request to the TTP.) By Lemma 7 we know that Bob cannot derive a message of the form of a contract issued by the TTP from $\mathcal{K}$.
- $\mathsf{state}_A(s) = \mathsf{initiator}$: Similar argument as in the previous case since Alice can decide not to send her promise of signature to Bob.
- $\mathsf{state}_A(s) = (i, \mathsf{sig\text{-}sent})$: Alice has the possibility to run the resolve subprotocol with the TTP to resolve the contract signing procedure since she has a promise of signature $\mathsf{ABposB}_r$ ($r \in \mathcal{T}(\Sigma)\downarrow$) from Bob. The TTP will issue a contract in response to Alices' resolve request since the TTP has not received an abort request for this instance of the protocol before: First, Alice has not sent an abort request to the TTP. Second, Bob cannot derive a valid abort request for this instance of the protocol because for this he would need to derive a signature of Alice on the message $\langle \mathsf{text}, \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{aborted} \rangle$ and this is impossible, see Lemma 5. (Also, Bob cannot send a message on $\mathsf{sec}_T^A$.) Thus, Bob does not have a strategy to prevent Alice from getting a contract.

- $\mathsf{state}_A(s) = (i, \mathsf{abort\text{-}req\text{-}sent})$: Let $m_0 s_0 \dots m_k s_k$ be a run segment of $\mathcal{G}_{GJM}$ that leads to $(m, s)$, i.e., $(m_k, s_k) = (m, s)$. Since we have $\mathsf{state}_A(s) = (i, \mathsf{abort\text{-}req\text{-}sent})$, we know that Alice has sent an abort request to the TTP in the last step, i.e., $m_k(\mathrm{sec}_T^A) = \mathsf{ABabort}_r$ for some $r \in \mathcal{T}(\Sigma)\downarrow$. We distinguish two cases:
  i) if the intruder has not sent an resolve request to the TTP, i.e., $m_i(\mathrm{sec}_T^B) \neq \mathsf{ABresB}_r$ for all $i \leq k$ and $r \in \mathcal{T}(\Sigma)\downarrow$, then he cannot get a contract anymore (the argument is similar as in the case of $\mathsf{state}_A(s) = 0$). So Bob does not have a strategy to get a contract.
  ii) if Bob has sent his resolve request to the TTP already, then Alice will get a contract in response to her abort request issued by the TTP. So Bob does not have a strategy to prevent Alice from getting a contract.
- $\mathsf{state}_A(s) = (i, \mathsf{aborted})$: Let $m_0 s_0 \dots m_k s_k$ be a run segment of $\mathcal{G}_{GJM}$ that leads to $(m, s)$, i.e., $(m_k, s_k) = (m, s)$. Since the TTP has sent an abort acknowledge message $\mathsf{ABabort\text{-}ack}_{r,r'}$ $(r, r' \in \mathcal{T}(\Sigma)\downarrow)$ to Alice, Alice must have sent an abort request $\mathsf{ABabort}_r$ to the TTP before, i.e., there is $i \leq k-2$ such that $m_i(\mathrm{sec}_T^A) = \mathsf{ABabort}_r$, $\mathsf{state}_A(s_{i+1}) = (i, \mathsf{abort\text{-}req\text{-}sent})$, $m_{i+1}(\mathrm{sec}_A^T) = \mathsf{ABabort\text{-}ack}_{r,r'}$, and $\mathsf{state}_A(s_{i+2}) = (i, \mathsf{aborted})$. If for some descendant $(m', s')$ of $(m_i, s_i)$ we have that $m'(\mathrm{sec}_T^B) = \mathsf{ABresB}_{u,v,w}$ for some $u, v, w \in \mathcal{T}(\Sigma)\downarrow$, then according to the definition of the TTP we have $m''(\mathrm{sec}_B^T) = \mathsf{ABabort\text{-}ack}_{r,r'}$ for the successor state $(m'', s'')$ of $(m', s')$. As in the case of $\mathsf{state}_A(s) = 0$, we can conclude from the fact that Alice neither has sent her signature $\mathsf{sconvert}(u, \mathsf{sk}(A), \mathsf{ABposA}_v)$ $(u, v \in \mathcal{T}(\Sigma)\downarrow)$ to Bob nor can use the TTP to get a contract, that Bob will not get a contract anymore. Hence, Bob does not have a strategy to get a contract.
- $\mathsf{state}_A(s) \in \{(i, \mathsf{contract}), (i, \mathsf{resolved})\}$: Alice has a contract so Bob does not have a strategy to prevent Alice from getting a contract.

Now we show that if $\mathsf{state}_A(s) = (i, \mathsf{pos\text{-}sent})$ and condition ii) or condition iii) does not hold we also have that $(m, s)$ is $\beta_{GJM}$-balanced. If condition ii) does not hold we have that $m(\mathrm{netout}_A^B) = \mathsf{ABposB}_r$ for some $r \in \mathcal{T}(\Sigma)\downarrow$ and according to the protocol specification Alice will send her signature to Bob and in the successor state $(m', s')$ of $(m, s)$ we will have that $\mathsf{state}_A(s') = (i, \mathsf{sig\text{-}sent})$. In $(m', s')$ Alice could send her resolve request to the TTP and will get a contract issued by the TTP back in response. Note that the intruder cannot send a valid abort request for this instance of the protocol since he cannot derive a message of the desired form (see case $\mathsf{state}_A(s) = (i, \mathsf{sig\text{-}sent})$). Thus, Bob cannot prevent Alice from getting a valid contract. If condition ii) holds but condition iii) does not, then we know that for the run segment $m_0 s_0 \dots m_k s_k$ such that $(m_k, s_k) = (m, s)$ we have that there is some $i \leq k$ and $r, r', r''$ such that $m_i(\mathrm{sec}_T^B) = \mathsf{ABresB}_{r,r',r''}$. Hence, the TTP gets the resolve request from Bob and will issue a valid contract. If Alice contacts the TTP by an abort request or a resolve request after $(m, s)$ she will get a contract issued by the TTP in response. Consequently, Bob cannot prevent Alice from getting a contract.

**2)⇒1):** We show that if conditions i),ii), and iii) are satisfied state $(m, s)$ is $\beta_{GJM}$-unbalanced. We have to show that Bob has two strategies a) a strategy to

get a contract and b) a strategy to prevent Alice from getting a contract. The strategy from Bob to prevent Alice from getting a contract is to do nothing. In this case Alice will not get the promise of signature from Bob that she would need to get a contract issued by the TTP and Alice will not get the signature from Bob directly. The strategy for Bob to get a contract is to send an resolve request to the TTP, i.e., send $\mathsf{ABresB}_{r,r',r''}$ for some $r, r', r'' \in \mathcal{T}(\Sigma)\downarrow$ to the TTP. By condition i) we know that Alice is in state $(i, \mathsf{pos\text{-}sent})$, i.e., $\mathsf{state}_A(s) = (i, \mathsf{pos\text{-}sent})$. Thus, Alice has not sent an abort request to the TTP, yet. Hence, if she sends her abort request to the TTP in state $(m, s)$, then, by the definition of the TTP (Figure 6), the resolve request from Bob will take priority over the abort request and the TTP will issue a contract to Bob. If Alice does not contact the TTP in $(m, s)$, then the TTP will also issue a contract to Bob. Thus, Bob will get a contract and thus his strategy will succeed. □

In a situation where Alice has just sent a promise of signature $\mathsf{ABposA}$ to the intruder, condition 2) of Lemma 4 is satisfied. Thus, as an immediate consequence of Lemma 4 we obtain:

**Theorem 4.** *The protocol $\mathcal{G}_{GJM}$ is $\beta_{GJM}$-unbalanced.*

As in case of the ASW-protocol, we also obtain that the system is unbalanced for the responder if the TTP gives priority to abort over resolvebe. If $\mathcal{G}'_{GJM}$ denotes the corresponding system and $\beta'_{GJM}$ the corresponding balance specifier, we obtain:

**Theorem 5.** *The protocol $\mathcal{G}'_{GJM}$ is $\beta'_{GJM}$-unbalanced.*

### 7.2 GJM Protocol is Abuse-free

For abuse-freeness, we imagine that Charlie assumes that there is only one instance of the GJM protocol running, but that he does not know whether Alice is the initiator or responder, which is a realistic assumption. Formally, we replace $\mathcal{A}_A$ by a variant of it, denoted $\mathcal{A}_{A'}$, which in the beginning decides whether it wants to play the role of the initiator or the responder and then sends a corresponding message to Bob, see pseudocode description in Figures 7 and 8. We set $\mathcal{G}^e_{GJM} = \mathrm{DY}[\{\mathcal{A}_i\}_{i\in\{A,T\}}, \{A', T\}, \{B\}, \mathcal{K}]$ with $\mathcal{K}$ as above, $\beta^e_{GJM} = (\{\mathcal{I}^e\}, \mathsf{view}_{\mathcal{I}}, \bar{\varphi}_A, \varphi_{\mathcal{I}})$, and $\mathcal{X} = \{A, B, T, \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{pk}(T), \mathsf{pk}(C), \mathsf{sk}(C), \mathsf{text}, \mathsf{ok}, \mathsf{pcsok}, \mathsf{tpok}, \mathsf{sok}\}$. Here, $\mathcal{I}^e$ denotes the intruder of $\mathcal{G}^e$. While here $T$ gives priority to resolve over abort, we also consider the situation where $T$ is replaced by $T'$ which gives priority to abort over resolve. The corresponding external view is denoted by $(\mathcal{G}'^e_{ASW}, \beta'^e_{ASW})$. We have:

**Theorem 6 (GJM is abuse-free).** *The external views $(\mathcal{G}^e_{GJM}, \beta^e_{GJM})$ and $(\mathcal{G}'^e_{GJM}, \beta'^e_{GJM})$ are $\mathcal{X}$-abusive-free.*

We present a detailed proof for $(\mathcal{G}^e_{GJM}, \beta^e_{GJM})$. The proof for $(\mathcal{G}'^e_{GJM}, \beta'^e_{GJM})$ is analogous.

In what follows for $r \in \mathcal{T}(\Sigma)$ we abbreviate $\mathsf{pcs}(r, \mathsf{sk}(A), \langle c, 1\rangle, \mathsf{pk}(B), \mathsf{pk}(T))$ by $\mathsf{PCS}_r$ and $\mathsf{fake}(r, \mathsf{sk}(B), \langle c, 1\rangle, \mathsf{pk}(A), \mathsf{pk}(T))$ by $\mathsf{FAKE}_r$.

For a state $(m, s)$ of $\mathcal{G}^e_{GJM}$ and $E \subseteq \mathcal{T}(\Sigma)$ we say that the *intruder knowledge in state* $(m, s)$ *is* $E$ if the set of messages the intruder can derive in $(m, s)$ is $d_{\mathcal{S}}(E)$.

We now state several lemmas and use them to prove Theorem 6. The proof of the lemmas then follow.

In the next Lemma the $\beta^e_{GJM}$-unbalanced states of $\mathcal{G}^e_{GJM}$ are characterised and it is stated what kind of knowledge the intruder has in these $\beta^e_{GJM}$-unbalanced states.

**Lemma 8.** *a) Let $(m, s)$ be a reachable state of $\mathcal{G}^e_{GJM}$. Then the following two conditions are equivalent:*
*1) $(m, s)$ is $\beta^e_{GJM}$-unbalanced.*
*2) i) $\mathsf{state}_A(s) = (i, \mathsf{pos\text{-}sent})$ and*
*ii) $m(\mathit{netout}^B_A) \neq \mathsf{ABposB}_r$ for all $r \in \mathcal{T}(\Sigma)\downarrow$, and*
*iii) there is a run segment $m_0 s_0 \dots m_k s_k$ such that $(m_k, s_k) = (m, s)$ and for all $i \leq k$ and $r \in \mathcal{T}(\Sigma)\downarrow$ we have $m_i(\mathit{sec}^B_T) \neq \mathsf{ABresB}_r$.*
*b) Let $(m, s)$ be a state of $\mathcal{G}^e_{GJM}$ that is $\beta^e_{GJM}$-unbalanced. Then the intruder knowledge in $(m, s)$ is $(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\})$ or $(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r, \mathsf{BAabort\text{-}ack}_{r',r''}\})$ for some $r, r', r'' \in \mathcal{T}(\Sigma)\downarrow$.*

Lemma 9 states that if the intruder derives a message $m$ from his knowledge in an $\beta^e_{GJM}$-unbalanced state, then Charlie neither can derive the secret key of Alice nor the secret key of the TTP from this message $m$.

**Lemma 9.** *We have that*

*a) Let $r \in \mathcal{T}(\Sigma)\downarrow$. If $m \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\})$, then $\mathcal{X} \cup \{m\}$ is $\{\mathsf{sk}(A), \mathsf{sk}(T)\}$-free.*
*b) Let $r, r', r'' \in \mathcal{T}(\Sigma)\downarrow$. If $m \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r, \mathsf{BAabort\text{-}ack}_{r',r''}\})$, then $\mathcal{X} \cup \{m\}$ is $\{\mathsf{sk}(A), \mathsf{sk}(T)\}$-free.*

The following Lemma states that if Charlie neither can derive the secret key of Alice nor the secret key of the TTP from a message $m$, then for $\mathcal{X}$-tests $\theta$ and messages $m$ there is an intruder randomness $r \in \mathcal{R}_{\mathcal{I}}$ such that if $m$ passes $\theta$, then $m_{|\mathsf{PCS}_{r'} \to \mathsf{FAKE}_r}\downarrow$ will also pass *theta*.

**Lemma 10.** *Let $\theta$ be an $\mathcal{X}$-test, $m \in \mathcal{T}(\Sigma)\downarrow$ such that $\mathcal{X} \cup \{m\}$ is $\{\mathsf{sk}(A), \mathsf{sk}(T)\}$-free, $r \in \mathcal{T}(\Sigma)\downarrow$ and $r' \in \mathcal{R}_{\mathcal{I}}$ such that $r' \notin \mathit{Sub}(m)$. Then, $m \models \theta$ implies $m_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}}\downarrow \models \theta$.*

Lemma 11 shows that if the intruder can derive a message $m$ from his knowledge in an $\beta^e_{GJM}$-unbalanced state of $\mathcal{G}^e_{GJM}$ he can also derive some fake version $m_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}}$ from this knowledge reduced by the message $\mathsf{PCS}_r$.

**Lemma 11.** *Let $r \in \mathcal{T}(\Sigma)\downarrow$, $r' \in \mathcal{R}_{\mathcal{I}}$. Then, we have that*

*a) If $m \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\})$, then $m_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}}\downarrow \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}})$.*
*b) For all $u, v \in \mathcal{T}(\Sigma)\downarrow$ we have that if $m \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r, \mathsf{BAabort\text{-}ack}_{u,v}\})$, then $m_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}}\downarrow \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{BAabort\text{-}ack}_{u,v}\})$.*

Now we can prove Theorem 6.

*Proof:* To show that $\mathcal{G}^e_{GJM}$ is $\beta^e_{GJM}$ $\mathcal{X}$-abuse-free according to Definition 12 it suffices to show that there is no $\beta^e_{GJM}$-convincing $\mathcal{X}$-test. We prove this by contradiction. Assume that $\theta$ is a $\beta_G JM^e$-convincing $\mathcal{X}$-test. Then, by Definition 12 there is a $\theta$-possible state $(m, s)$ of $\mathcal{G}^e_{GJM}$ which is $\beta^e_{GJM}$-unbalanced. It suffices to show that there is a $\theta$-possible $\beta^e_{GJM}$-balanced state $(m', s')$ of $\mathcal{G}^e_{GJM}$ that is not a descendant of an $\beta^e_{GJM}$-unbalanced state. For this let $m''$ be a message that can be derived by the intruder in state $(m, s)$ such that $m''$ passes $\theta$. According to Lemma 8 part b) we know that $m'' \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\})$ or $m'' \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r, \mathsf{BAabort\text{-}ack}_{u,v}\})$ for some $r, u, v \in \mathcal{T}(\Sigma)\downarrow$. We distinguish between these two cases and show that in each of these cases there is a state $(m', s')$ with the conditions described above.

- $m'' \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\})$ for some $r \in \mathcal{T}(\Sigma)\downarrow$: Since $m''$ is a finite term there is $r' \in \mathcal{R}_{\mathcal{I}}$ such that $r' \notin \mathrm{Sub}(m)$. Then, by Lemma 9 and Lemma 10 we have that $m''_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}}\downarrow$ passes $\theta$. By Lemma 11 we have that $m''_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}}\downarrow \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}})$, so we can choose $(m', s')$ to be the initial state $(m_0, s_0)$ of $\mathcal{G}^e_{GJM}$. Since in $(m_0, s_0)$ the intruder can derive $m''_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}}\downarrow$ state $(m_0, s_0)$ is $\theta$-possible and by Lemma 8 part a) we know that $(m_0, s_0)$ is $\beta^e_{GJM}$-balanced and $(m_0, s_0)$ is obviously not a descendant of any $\beta^e_{GJM}$-unbalanced state. So in this case we are done.
- $m'' \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r, \mathsf{BAabort\text{-}ack}_{u,v}\})$ for some $r, u, v \in \mathcal{T}(\Sigma)\downarrow$: Since $m''$ is a finite term there is $r' \in \mathcal{R}_{\mathcal{I}}$ such that $r' \notin \mathrm{Sub}(m)$. Then by Lemma 9 and Lemma 10 we have that $m''_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}}\downarrow$ passes $\theta$. By Lemma 11 we have that $m''_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}}\downarrow \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{BAabort\text{-}ack}_{u,v}\})$. We can choose $(m', s')$ to be the state of $\mathcal{G}^e_{GJM}$ that is reached starting at the initial state $(m_0, s_0)$ of $\mathcal{G}^e_{GJM}$ after the intruder has sent the abort request as initiator to the TTP and got $\mathsf{BAabort\text{-}ack}$ back in response from the TTP. More formally, consider the run segment $m_0 s_0 m_1 s_1 m_2 s_2$, where $(m_0, s_0)$ is the initial state of $\mathcal{G}^e_{GJM}$, $\mathsf{state}_A(s_0) = \mathsf{state}_A(s_1) = \mathsf{state}_A(s_2) = 0$, $m_1(\mathrm{sec}^B_T) = \mathsf{BAabort}_v$, $m_2(\mathrm{sec}^T_B) = \mathsf{BAabort\text{-}ack}_{u,v}$, $m_1(p) = \circ$ for all ports $p \neq \mathrm{sec}^B_T$ of $\mathcal{G}^e_{GJM}$, and $m_2(p) = \circ$ for all ports $p \neq \mathrm{sec}^T_B$ of $\mathcal{G}^e_{GJM}$. In state $(m_2, s_2)$ the knowledge of the intruder exactly is $\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{BAabort\text{-}ack}_{u,v}\}$. By Lemma 8 part a) we know that $(m_i, s_i)$ for $i \in \{0, 1, 2\}$ are $\beta^e_{GJM}$-balanced so we can choose $(m', s')$ to be $(m_2, s_2)$ and we are done. □

In the next sections we give the proofs of the above mentioned lemmas.

**Proof of Lemma 8** Lemma 8 characterizes the set of unbalanced states of the protocol that is assumed by Charlie. The proof of this lemma is similiar to the proof of Lemma 4 which characterizes the unbalanced states of the protocol that is actually run between Alice, Bob, and the TTP.

*Proof of Lemma 8:*

a) In the protocol $\mathcal{G}^e_{GJM}$ Bob is modelled by the intruder. Therefore, in the following we write 'Bob' instead of 'the intruder'.

**1)⇒2):** We prove this implication by contraposition. First we show that if $\mathsf{state}_A(s) \neq (i, \mathsf{pos\text{-}sent})$, then $(m, s)$ is $(I, \mathsf{view}, \varphi_1, \varphi_2)$-balanced. We distinguish between the possible values of $\mathsf{state}_A(s)$. For

$$\mathsf{state}_A(s) \in \{0, \mathsf{initiator}, (i, \mathsf{sig\text{-}sent}), (i, \mathsf{abort\text{-}req\text{-}sent}), (i, \mathsf{aborted}), (i, \mathsf{resolve\text{-}req\text{-}sent}), (i, \mathsf{contract}), (i, \mathsf{resolved})\}$$

the arguments are similar as in the corresponding cases of Lemma 4. Thus, we only have to consider the values of $\mathsf{state}_A(s)$ that correspond to the responder part for Alice:

- $\mathsf{state}_A(s) = \mathsf{responder}$: Bob does not have a strategy to get a contract since Alice has the possibility to ignore a promise of signature from Bob if she wants to abort the contract signing. More specifically, if $m'(\text{netout}_A^B) = \mathsf{BAposB}_r$ for some $r \in \mathcal{T}(\Sigma)\downarrow$, $m' = m$ or a descendant $(m', s')$ of $(m, s)$ such that $\mathsf{state}_A(s') = \mathsf{responder}$, then Alice could take the transition such that we have $\mathsf{state}_A(s'') = \mathsf{not\text{-}interested}$ for the successor $(m'', s'')$ of $(m', s')$. Since Alice neither has sent her promise of signature nor her signature to Bob Bob neither can use the TTP to resolve the contract singning nor he has the signature of Alice on the contract and cannot derive this signature on his own (similar argument as in case $\mathsf{state}_A(s) = 0$ in Lemma 4). Thus, Bob does not have a strategy to get a contract.
- $\mathsf{state}_A(s) = (r, \mathsf{pos\text{-}sent})$: If Alice receives Bobs signature in state $(m, s)$, then obviously, Bob does not have a strategy to prevent Alice from getting a contract.
  Otherwise, if Alice does not receive Bobs signature in state $(m, s)$ but Bob has sent his abort request as initiator to the TTP already, i.e., $m_i(\text{sec}_T^B) = \mathsf{BAabort}_r$ for some $i < k$ and $r \in \mathcal{T}(\Sigma)\downarrow$, then if Alice decides to send an resolve request to the TTP, that is $m'(\text{sec}_T^A) = \mathsf{ABresA}_{r',r'',r'''}$ for some $r', r'', r''' \in \mathcal{T}(\Sigma)\downarrow$ and $\mathsf{state}_A(s') = (r, \mathsf{resolve\text{-}req\text{-}sent})$ for the successor state $(m', s')$ of $(m, s)$, then if any Alice receives any message from Bob later on she will not react on this message since she simply waits for the response from the TTP to her resolve request. Since she did not send her signature to Bob and Bob cannot derive Alices signature on his own and the TTP has already aborted the contract signing Bob does not have a strategy to get a contract.
  And in case Alice does not receive Bobs signature in state $(m, s)$ and Bob has not sent his abort request to the TTP yet, Alice could send an resolve request to the TTP, that is, $m'(\text{sec}_T^A) = \mathsf{BAresA}_{r',r'',r'''}$ for some $r', r'', r''' \in \mathcal{T}(\Sigma)\downarrow$ and $\mathsf{state}_A(s') = (r, \mathsf{resolve\text{-}req\text{-}sent})$ for the successor state $(m', s')$ of $(m, s)$ then the TTP will issue a contract in response, note that a resolve request takes priority over an abort request sent by Bob in state $(m', s')$, so even if Bob has sent his abort request in $(m', s')$ the TTP will issue a contract. Hence, Bob does not have a strategy to prevent Alice from getting a contract.
- $\mathsf{state}_A(s) = (r, \mathsf{resolve\text{-}req\text{-}sent})$: We know that $m(\text{sec}_T^A) = \mathsf{BAresA}_{r,r',r''}$ for some $r, r', r'' \in \mathcal{T}(\Sigma)\downarrow$. If for some $i < k$ we have that $m_i(\text{sec}_T^B) =$

$\mathsf{BAabort}_{r'''}$ for some $r''' \in \mathcal{T}(\Sigma)\downarrow$, then Bob has no strategy to get a contract anymore since the TTP will not resolve the contract signing and Alice has not sent her signature to Bob explicitly and Bob cannot derive Alices' signature on the contract on his own. If Bob has not sent an abort request yet, i.e., for all $r''' \in \mathcal{T}(\Sigma)\downarrow$ we have $m_i(\mathrm{sec}_T^B) \neq \mathsf{BAabort}_{r'''}$ for $i < k$ then the TTP will issue a contract for Alice, note that even if $m(\mathrm{sec}_T^B) = \mathsf{BAabort}_{r'''}$ for some $r''' \in \mathcal{T}(\Sigma)\downarrow$ the resolve request from Alice will take priority over the abort request from Bob. So Alice will get a contract issued by the TTP and so Bob does not have a strategy to prevent Alice from getting a contract.

- $\mathsf{state}_A(s) \in \{(r, \mathsf{contract}), (r, \mathsf{resolved})\}$: Alice has a contract so Bob does not have a strategy to prevent Alice from getting a contract.
- $\mathsf{state}_A(s) = (r, \mathsf{aborted})$: Since the TTP has sent BAabort-ack message in response to Alices resolve request Bob cannot use the TTP to get a valid contract anymore. So Bob does not have a strategy to get a contract.

b) We know that $\mathsf{state}_A(s) = (i, \mathsf{pos\text{-}sent})$. If the intruder has sent an abort request to the TTP and got an abort acknowledge message $\mathsf{BAabort\text{-}ack}_{r',r''}$ for some $r', r'' \in \mathcal{T}(\Sigma)\downarrow$ back in response it is easy to see that the knowledge of the intruder in $(m, s)$ is $\mathsf{know}(m, s) = \mathcal{K}_\mathcal{I} \cup \{\mathsf{PCS}_r, \mathsf{BAabort\text{-}ack}_{r',r''}\}$ for some $r \in \mathcal{T}(\Sigma)\downarrow$. Otherwise, if the the intruder has not got an abort acknowledge message from the TTP it is easy to see that the of the intruder is $\mathcal{K}_\mathcal{I} \cup \{\mathsf{PCS}_r\}$ for some $r \in \mathcal{T}(\Sigma)\downarrow$. □

**Proof of Lemma 9** In order to prove Lemma 9 we need a characterization of the sets $K \subseteq \mathcal{T}(\Sigma)$ from which messages of the form $\mathsf{sk}(t)$ for $t \in \mathcal{T}(\Sigma)\downarrow$ can be derived. The following lemma states that a message of the form $\mathsf{sk}(t)$ can be derived from $K$ iff $\mathsf{sk}(t)$ can be obtained by applying a sequence of projections to some message $m \in K$, i.e., $\mathsf{sk}(t) = \pi_\kappa$ for some $\kappa \in \{1, 2\}^*$.

**Lemma 12.** *Let $K \subseteq \mathcal{T}(\Sigma)\downarrow$ and $t \in \mathcal{T}(\Sigma)\downarrow$. Then the following conditions are equivalent:*

*1) $\mathsf{sk}(t) \notin d_\mathcal{S}(K)$*
*2) For all $M \in K$ and $p \in Pos(M)$ such that $M|_p = \mathsf{sk}(t)$ there is a proper prefix $p'$ of $p$ such that $\mathsf{root}(M|_{p'}) \neq \langle, \rangle$.*

**Proof:** Let $m \in \mathcal{T}(\Sigma)\downarrow$.

– 2) ⇒ 1): Assume that 2) holds. We show by induction on the structure of $N$ that for every $\mathcal{S}$-term $N$ and each $K$-instance $\hat{N}$ of $N$ we have that $\{\hat{N}\downarrow\}$ satisfies condition 2). From this 1) follows immediately.
If $N$ is a variable, then $\hat{N} \in K$ and the claim follows directly from condition 2).
If $N$ is of the form $f(N_1, \ldots, N_l)$ for $\mathcal{S}$-terms $N_1, \ldots, N_l$ and $\hat{N}$ is a $K$-instance of $N$ there are $t_1, \ldots, t_n \in K$ such that $\hat{N}{=}N[t_1, \ldots, t_n/x_1, \ldots, x_n]$,

where $\{x_1, \ldots, x_n\} = \mathcal{V}(N)$. Then we have

$$\begin{aligned}
&\hat{N}\downarrow = N[t_1, \ldots, t_n/x_1, \ldots, x_n]\downarrow \\
&= f(N_1, \ldots, N_l)[t_1, \ldots, t_n/x_1, \ldots, x_n]\downarrow \\
&= f(N_1[t_1, \ldots, t_n/x_1, \ldots, x_n], \ldots, N_l[t_1, \ldots, t_n/x_1, \ldots, x_n])\downarrow \\
&= f(N_1[t_1, \ldots, t_n/x_1, \ldots, x_n]\downarrow, \ldots, N_l[t_1, \ldots, t_n/x_1, \ldots, x_n]\downarrow)\downarrow .
\end{aligned}$$

By induction we know that $\{\hat{N}_i\}$ satisties condition 2) where

$$\hat{N}_i = N_i[t_1, \ldots, t_n/x_1, \ldots, x_n]\downarrow .$$

We need to show that $\{\hat{N}\downarrow\}$ satisfies condition 2). We need to show that if $\hat{N}\downarrow |_p = \mathsf{sk}(t)$, then there is a proper prefix $p'$ of $p$ such that $\mathsf{root}(\hat{N}\downarrow |_{p'}) \neq \langle,\rangle$. If $f(\hat{N}_1, \ldots, \hat{N}_l)$ is in normal form we have that $\hat{N}\downarrow$ satisfies condition 2) since $\hat{N}_1, \ldots, \hat{N}_l$ satisfy condition 2) and $f \neq \mathsf{sk}$ (note $\mathsf{sk}(x) \notin \mathcal{S}$). If $f(\hat{N}_1, \ldots, \hat{N}_l)$ is not in normal form we know that

$$f \in \{\pi_1, \pi_2, \mathsf{sconvert}, \mathsf{tpconvert}, \mathsf{sigcheck}, \mathsf{pcsver}, \mathsf{sver}, \mathsf{tpver}\} :$$

Since $\hat{N}_1, \ldots, \hat{N}_l$ are in normal form, an $\mathcal{H}$-identity that is applied to $f(\hat{N}_1, \ldots, \hat{N}_l)$ is applied at the root position of $f(\hat{N}_1, \ldots, \hat{N}_l)$. We now distinguish between the possible cases:

- $f = \pi_1$: Then, we have that $\pi_1(\hat{N}_1) \to_{\mathcal{H}} \hat{N}\downarrow$ is $\pi_1(\langle x, y\rangle) = x$. So, we have that $\hat{N}_1$ is of the form $\hat{N}_1 = \langle u_1, u_2\rangle$ and $\hat{N}\downarrow = u_1$. Let $\hat{N}\downarrow |_p = \mathsf{sk}(t)$. Then, we have that $\hat{N}_1|_{1p} = \mathsf{sk}(t)$. By induction, it follows that there is a proper prefix $p'$ of $1p$ such that $\mathsf{root}(\hat{N}_1|_{p'}) \neq \langle,\rangle$. Since $\mathsf{root}(\hat{N}_1) = \langle,\rangle$ we know that $p'$ is of the form $1p''$ where $p''$ is a proper prefix of $p'$. So, $\langle,\rangle \neq \mathsf{root}(\hat{N}_1|_{1p''}) = \mathsf{root}(u_1|_{p''}) = \mathsf{root}(\hat{N}\downarrow)$.
- $f = \pi_2$: Analogous argument.
- $f \in \{\mathsf{sigcheck}, \mathsf{pcsver}, \mathsf{sver}, \mathsf{tpver}\}$: Then, we have that $\hat{N}\downarrow \in \{\mathsf{ok}, \mathsf{pcsok}, \mathsf{sok}, \mathsf{tpok}\}$ and thus, $\mathsf{root}(\hat{N}\downarrow) \neq \langle,\rangle$.

– 1) $\Rightarrow$ 2): Easily shown by contraposition. □

Now we can turn to the proof of Lemma 9 which states that the intruder cannot derive a message from his knowledge in an unbalanced state of $\mathcal{G}^e_{GJM}$ from which Charlie can derive the secret key from Alice or the TTP.

*Proof of Lemma 9:*

a) Let $m \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\})$. We have to show that $\mathsf{sk}(A), \mathsf{sk}(T) \notin d_{\mathcal{S}}(\mathcal{X} \cup \{m\})$. It suffices to show that for $K = \mathcal{X} \cup \{m\}$ and $t \in \{A, T\}$ condition 2) of Lemma 12 is satisfied.
For all $M \in \mathcal{X}$ and $p \in \mathrm{Pos}(M)$ we obviously have that if $M|_p = \mathsf{sk}(t)$ there is a proper prefix $p'$ of $p$ such that $\mathsf{root}(M|_{p'}) \neq \langle,\rangle$.
For $M = m$ we know that $M \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\})$. Since for all $M' \in \mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\}$ and $p \in \mathrm{Pos}(M')$ we have that if $M'|_p = \mathsf{sk}(t)$ there is a proper prefix $p'$ of $p$ such that $\mathsf{root}(M'|_{p'}) \neq \langle,\rangle$ we know by Lemma 12 (2) $\Rightarrow$ 1)) that $\mathsf{sk}(t) \notin d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\})$. Since $M \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\})$ we have that $d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r, M\}) = d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\})$ and so $\mathsf{sk}(t) \notin d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r, M\})$. By Lemma 12 (1) $\Rightarrow$ 2)) it follows that $M$ satisfies the desired condition.

b) Analogously. □

**Proof of Lemma 10** One of the main lemmas used in our proof of Theorem 6 is Lemma 10. It states that if from message $m$ Charlie neither can derive the secret key from Alice nor the secret key from the TTP, then Charlie cannot perform a test $\theta$ such that $m$ passes $\theta$ but the message $m'$ that is obtained from $m$ by replacing $\mathsf{PCS}_r$ by a message of the form $\mathsf{FAKE}_{r'}$ does not pass $\theta$. For the proof of Lemma 10 we need a series of technical lemmas.

Lemma 13 states that if one applies an $\mathcal{H}$-identity to an instance of an $(\mathcal{S}\cup\mathcal{X})$-term, then one get an instance of some other $(\mathcal{S}\cup\mathcal{X})$-term of the same kind. This statement is needed for proofs by induction in subsequent lemmas.

**Lemma 13.** *Let $\mathcal{M}\subseteq\mathcal{T}(\Sigma)\downarrow$ be $\{\mathsf{sk}(A),\mathsf{sk}(T)\}$-free. Let $N$ be an $(\mathcal{S}\cup\mathcal{X})$-term. Let $\hat{N}$ be an $\mathcal{M}$-instance of $N$. Let $T\in\mathcal{T}(\Sigma)$ such that $\hat{N}\to_{\mathcal{H}} T$. Then $T$ is an $\mathcal{M}'$-instance of some $(\mathcal{S}\cup\mathcal{X})$-term $N'$, where $\mathcal{M}'\subseteq\mathcal{T}(\Sigma)\downarrow$ is $\{\mathsf{sk}(A),\mathsf{sk}(T)\}$-free.*

**Proof:** By induction on the size of position $p\in\mathbf{N}^*$ we show that for each $\{\mathsf{sk}(A),\mathsf{sk}(T)\}$-free $\mathcal{M}\subseteq\mathcal{T}(\Sigma)\downarrow$, $(\mathcal{S}\cup\mathcal{X})$-term $N$, $\mathcal{M}$-instance $\hat{N}$ of $N$, and $T\in\mathcal{T}(\Sigma)$ such that $\hat{N}\to^p_{\mathcal{H}} T$ we have that $T$ is an $\mathcal{M}'$-instance of some $(\mathcal{S}\cup\mathcal{X})$-term $N'$ where $\mathcal{M}'$ is $\{\mathsf{sk}(A),\mathsf{sk}(T)\}$-free and $\mathcal{M}\subseteq\mathcal{M}'\subseteq\mathcal{T}(\Sigma)\downarrow$.

Since we have that $\mathcal{M}\subseteq\mathcal{T}(\Sigma)\downarrow$ we know that $N$ is not a variable.

First, assume that $p=\epsilon$. We distinguish between the possible root symbols of $N$.

- $\mathsf{root}(N)\in\{\mathsf{pk},\langle,\rangle,\mathsf{sig},\mathsf{fake},\mathsf{pcs}\}$: This is not possible since there is no $\mathcal{H}$-identity with appropriate left-hand side.
- $\mathsf{root}(N)=\pi_1$: The $\mathcal{H}$-identity that is applied in step $\hat{N}\to_{\mathcal{H}} T$ is $\pi_1(\langle x,y\rangle)=x$ so $\hat{N}|_1$ is of the form $\hat{N}|_1=\langle t_1,t_2\rangle$ for some $t_1,t_2\in\mathcal{T}(\Sigma)$. We distinguish two cases.
  - $N|_1$ is a variable: Then we have that $\langle t_1,t_2\rangle\in\mathcal{M}$ since $\hat{N}$ is an $\mathcal{M}$-instance of $N$. Hence, $T=t_1$ and obviously we have that $\mathcal{M}'=\mathcal{M}\cup\{t_1\}$ is $\{\mathsf{sk}(A),\mathsf{sk}(T)\}$-free. So, $T$ is an $\mathcal{M}'$-instance of the $(\mathcal{S}\cup\mathcal{X})$-term $x_1\in\overline{\mathcal{V}}$.
  - $N|_1$ is of the form $N|_1=\langle t_1',t_2'\rangle$: Then we have that $t_1'$ is an $(\mathcal{S}\cup\mathcal{X})$-term and $T=t_1$ is an $\mathcal{M}$-instance of $t_1'$.
- $\mathsf{root}(N)=\pi_2$: Analogous argument.
- $\mathsf{root}(N)\in\{\mathsf{sigcheck},\mathsf{sver},\mathsf{tpver},\mathsf{pcsver}\}$: Then, we have that $T\in\{\mathsf{ok},\mathsf{pcsok},\mathsf{sok},\mathsf{tpok}\}$. Hence, $T$ is an $\mathcal{M}$-instance of the $(\mathcal{S}\cup\mathcal{X})$-term $T$.

Now, assume that $p$ is of the form $p=ip'$ for some $p'$ and $N$ is of the form $N=f(t_1',\ldots,t_l')$. Then, we have that $\hat{N}$ is of the form $\hat{N}=f(t_1,\ldots,t_l)$ where $t_j$ is an $\mathcal{M}$-instance of $t_j'$ for $j\in\{1,\ldots,l\}$. We have that $T$ is of the form $T=f(\overline{t}_1,\ldots,\overline{t}_l)$ where $\overline{t}_j=t_j$ for $j\neq i$ and $t_i\to^{p'}_{\mathcal{H}}\overline{t}_i$. By induction, we have that $\overline{t}_i$ is an $\mathcal{M}'$-instance of some $(\mathcal{S}\cup\mathcal{X})$-term $t$, where $\mathcal{M}'$ is $\{\mathsf{sk}(A),\mathsf{sk}(T)\}$-free and $\mathcal{M}\subseteq\mathcal{M}'\subseteq\mathcal{T}(\Sigma)\downarrow$. So $T$ is an $\mathcal{M}'$-instance of the $(\mathcal{S}\cup\mathcal{X})$-term $N[t]_i$. □

The next lemma states that if we instantiate an $\mathcal{S}$ term $N$ by some terms $t_1,\ldots,t_n$ from an $\{\mathsf{sk}(A),\mathsf{sk}(T)\}$-free set and then replace a message of the form $\mathsf{PCS}_r$ by some message of the form $\mathsf{FAKE}_{r'}$ we get the same term as if we instantiate $N$ by the terms $t_{1|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}},\ldots,t_{n|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}}$.

**Lemma 14.** *Let $\mathcal{M} \subseteq \mathcal{T}(\Sigma)\downarrow$ be $\{\mathsf{sk}(A), \mathsf{sk}(T)\}$-free and $N$ an $\mathcal{S}$-term. Let $r \in \mathcal{T}(\Sigma)\downarrow$ and let $r' \in \mathcal{R}_{\mathcal{I}}$. Then for all $t_1, \ldots, t_n \in \mathcal{M}$ we have that*

$$N[t_1, \ldots, t_n/x_1, \ldots, x_n]_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}} = N[t_{1|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}}, \ldots, t_{n|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}}/x_1, \ldots, x_n],$$

*where $\{x_1, \ldots, x_n\} = \mathcal{V}(N)$.*

**Proof:** In the following we will write $\mathsf{PCS}$ instead of $\mathsf{PCS}_r$ and $\mathsf{FAKE}$ instead of $\mathsf{FAKE}_{r'}$. We prove this lemma by induction on the structure of $N$:

- $N$ is a constant: We have $N[t_1, \ldots, t_n/x_1, \ldots, x_n] = N$ for all $t_1, \ldots, t_n \in \mathcal{T}(\Sigma)$ and $N_{|\mathsf{PCS}\to\mathsf{FAKE}} = N$.
- $N = x$ for some variable $x \in \overline{\mathcal{V}}$: We have

  $$N[m/x]_{|\mathsf{PCS}\to\mathsf{FAKE}} = \hat{m} = N[\hat{m}/x]$$

  where for $t \in \mathcal{T}(\Sigma)$ we denote $t_{|\mathsf{PCS}\to\mathsf{FAKE}}$ by $\hat{t}$ and we are done.
- $N$ is of the form $N = f(N_1, \ldots, N_l)$ for $\mathcal{S}$-terms $N_1, \ldots, N_l$ where $f \neq \mathsf{pcs}$: We have that

  $$\begin{aligned} & N[t_1, \ldots, t_n/x_1, \ldots, x_n]_{|\mathsf{PCS}\to\mathsf{FAKE}} \\ = {} & f(N_1[t_1, \ldots, t_n/x_1, \ldots, x_n], \ldots, N_l[t_1, \ldots, t_n/x_1, \ldots, x_n])_{|\mathsf{PCS}\to\mathsf{FAKE}} \\ = {} & f(N_1[t_1, \ldots, t_n/x_1, \ldots, x_n]_{|\mathsf{PCS}\to\mathsf{FAKE}}, \ldots, N_l[t_1, \ldots, t_n/x_1, \ldots, x_n]_{|\mathsf{PCS}\to\mathsf{FAKE}}) \end{aligned}$$

  and by induction we proceed

  $$\begin{aligned} & f(N_1[t_1, \ldots, t_n/x_1, \ldots, x_n]_{|\mathsf{PCS}\to\mathsf{FAKE}}, \ldots, N_l[t_1, \ldots, t_n/x_1, \ldots, x_n]_{|\mathsf{PCS}\to\mathsf{FAKE}}) \\ = {} & f(N_1[\hat{t}_1, \ldots, \hat{t}_n/x_1, \ldots, x_n], \ldots, N_l[\hat{t}_1, \ldots, \hat{t}_n/x_1, \ldots, x_n]) \\ = {} & f(N_1, \ldots, N_l)[\hat{t}_1, \ldots, \hat{t}_n/x_1, \ldots, x_n] \\ = {} & N[\hat{t}_1, \ldots, \hat{t}_n/x_1, \ldots, x_n]. \end{aligned}$$

- $N$ is of the form $N = \mathsf{pcs}(N_1, N_2, N_3, N_4, N_5)$: By the fact that $\mathcal{M}$ is $\{\mathsf{sk}(A), \mathsf{sk}(T)\}$-free we know that $N_2[t_1, \ldots, t_n/x_1, \ldots, x_n] \neq \mathsf{sk}(A)$ and so we have

  $$\begin{aligned} & N[t_1, \ldots, t_n/x_1, \ldots, x_n]_{|\mathsf{PCS}\to\mathsf{FAKE}} \\ = {} & \mathsf{pcs}(N_1[t_1, \ldots, t_n/x_1, \ldots, x_n]_{|\mathsf{PCS}\to\mathsf{FAKE}}, \ldots, N_4[t_1, \ldots, t_n/x_1, \ldots, x_n]_{|\mathsf{PCS}\to\mathsf{FAKE}}) \end{aligned}$$

  and by induction

  $$\begin{aligned} & \mathsf{pcs}(N_1[t_1, \ldots, t_n/x_1, \ldots, x_n]_{|\mathsf{PCS}\to\mathsf{FAKE}}, \ldots, N_4[t_1, \ldots, t_n/x_1, \ldots, x_n]_{|\mathsf{PCS}\to\mathsf{FAKE}}) \\ = {} & \mathsf{pcs}(N_1[\hat{t}_1, \ldots, \hat{t}_n/x_1, \ldots, x_n], \ldots, N_4[\hat{t}_1, \ldots, \hat{t}_n/x_1, \ldots, x_n]) \\ = {} & N[\hat{t}_1, \ldots, \hat{t}_n/x_1, \ldots, x_n]. \end{aligned}$$

□

One of the main lemmas needed in our proof of Lemma 10 is the following one. It states that if we can get from an instance $\hat{N}$ of a particular form of some $(\mathcal{S} \cup \mathcal{X})$-term to a term $T$ by applying a sequence of $\mathcal{H}$-identities, then we can get from $\hat{N}_{|\mathsf{PCS}\to\mathsf{FAKE}}$ to $T_{|\mathsf{PCS}\to\mathsf{FAKE}}$ by applying a similar sequence of $\mathcal{H}$-identities. For this proof proof to work it is crucial that for the $\mathcal{H}$-indentity $\mathsf{pcsver}(w, \mathsf{pk}(x), \mathsf{pk}(y), \mathsf{pk}(z), \mathsf{pcs}(u, \mathsf{sk}(x), w, \mathsf{pk}(y), \mathsf{pk}(z))) = \mathsf{pcsok}$ there is a corresponding $\mathcal{H}$-identity

$$\mathsf{pcsver}(w, \mathsf{pk}(x), \mathsf{pk}(y), \mathsf{pk}(z), \mathsf{fake}(u, \mathsf{sk}(y), w, \mathsf{pk}(x), \mathsf{pk}(z))) = \mathsf{pcsok}.$$

**Lemma 15.** *Let $\mathcal{M} \subseteq \mathcal{T}(\Sigma)\downarrow$ be $\{\mathsf{sk}(A), \mathsf{sk}(T)\}$-free. Let $r \in \mathcal{T}(\Sigma)\downarrow$ and let $r' \in \mathcal{R}_{\mathcal{I}}$. Then, for all $(\mathcal{S} \cup \mathcal{X})$-terms $N$, $\mathcal{M}$-instances $\hat{N}$ of $N$, and $T \in \mathcal{T}(\Sigma)$ we have that $\hat{N} \xrightarrow{*}_{\mathcal{H}} T$ implies $\hat{N}_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}} \xrightarrow{*}_{\mathcal{H}} T_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}}$.*

**Proof:** In the following we will write $\mathsf{PCS}$ instead of $\mathsf{PCS}_r$ and $\mathsf{FAKE}$ instead of $\mathsf{FAKE}_{r'}$. We show the lemma by induction on the size of $\hat{N}$.

If $\hat{N}$ is a constant we have that $\hat{N}_{|\mathsf{PCS}\to\mathsf{FAKE}} = \hat{N} = T = T_{|\mathsf{PCS}\to\mathsf{FAKE}} = T$.

Now, let $\hat{N}$ be of the form $\hat{N} = f(t_1, \ldots, t_l)$. If $T = \hat{N}$ we are done. Otherwise, there is $\overline{N} \in \mathcal{T}(\Sigma)$ such that

$$\hat{N} \to^{p}_{\mathcal{H}} \overline{N} \xrightarrow{*}_{\mathcal{H}} T$$

for some $p \in \mathrm{Pos}(\hat{N})$. We consider different cases for $p$:

- $p = \epsilon$: We know that $N$ is not a variable since $\mathcal{M} \subseteq \mathcal{T}(\Sigma)\downarrow$. We distinguish between the possible root symbols of $\hat{N}$.
  - $\mathsf{root}(\hat{N}) = \pi_1$: We know that $\hat{N}$ is of the form $\hat{N} = \pi_1(\langle t_1, t_2\rangle)$, $\overline{N} = t_1$, and $N$ is of the form $N = \pi_1(t)$ for some $(\mathcal{S} \cup \mathcal{X})$-term $t$. We have that $|\overline{N}| = |t_1| < |\hat{N}|$ and by Lemma 13 we know that $\overline{N}$ is an $\mathcal{M}'$-instance of some $(\mathcal{S}\cup\mathcal{X})$-term $t'_1$, such that $\mathcal{M}'$ is $\{\mathsf{sk}(A), \mathsf{sk}(T)\}$-free. By induction, it follows that
    $$t_{1\,|\mathsf{PCS}\to\mathsf{FAKE}} \xrightarrow{*}_{\mathcal{H}} T_{|\mathsf{PCS}\to\mathsf{FAKE}}.$$
    We now have to show that
    $$\hat{N}_{|\mathsf{PCS}\to\mathsf{FAKE}} \xrightarrow{*}_{\mathcal{H}} t_{1\,|\mathsf{PCS}\to\mathsf{FAKE}}.$$
    We have that $\hat{N} \to_{\mathcal{H}} t_1$ by applying the identity $\pi_1(\langle x, y\rangle) = x$ at the root of $\hat{N}$. Hence, there is a substitution $\sigma$ such that $\hat{N} = \sigma(\pi_1(\langle x, y\rangle))$ and $\sigma(x) = t_1$. With $\hat{\sigma}(x) = \sigma(x)_{|\mathsf{PCS}\to\mathsf{FAKE}}$ and $\hat{\sigma}(y) = \sigma(y)_{|\mathsf{PCS}\to\mathsf{FAKE}}$ we have that $\hat{N}_{|\mathsf{PCS}\to\mathsf{FAKE}} = \hat{\sigma}(\pi_1(\langle x, y\rangle))$ and $t_{1\,|\mathsf{PCS}\to\mathsf{FAKE}} = \hat{\sigma}(x)$.
  - $\mathsf{root}(\hat{N}) = \pi_2$: Analogous argument.
  - $\mathsf{root}(\hat{N}) = \mathsf{pcsver}$: We have that $T = \mathsf{pcsok}$ and $\hat{N}$ is of the form $\hat{N} = \mathsf{pcsver}(t_1, t_2, t_3, t_4, t_5)$. If $t_5 \neq \mathsf{PCS}$ the argument is similar as in the previous cases. If $t_5 = \mathsf{PCS}$ we have that
    $$\hat{N}_{|\mathsf{PCS}\to\mathsf{FAKE}} = \mathsf{pcsver}(\mathsf{text}, \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{pk}(T), \mathsf{FAKE})$$
    and thus we can apply the $\mathcal{H}$-identity
    $$\mathsf{pcsver}(w, \mathsf{pk}(x), \mathsf{pk}(y), \mathsf{pk}(z), \mathsf{fake}(v, \mathsf{sk}(y), w, \mathsf{pk}(x), \mathsf{pk}(z))) = \mathsf{pcsok}$$
    to show that $\hat{N}_{|\mathsf{PCS}\to\mathsf{FAKE}} \to_{\mathcal{H}} \mathsf{pcsok} = T_{|\mathsf{PCS}\to\mathsf{FAKE}}$.
  - $\mathsf{root}(\hat{N}) \in \{\mathsf{sig}, \mathsf{pk}, \mathsf{sk}, \langle,\rangle, \mathsf{fake}, \mathsf{pcs}\}$: Impossible since there is no $\mathcal{H}$-identity with an appropriate left-hand side.

- $\mathsf{root}(\hat{N}) = \mathsf{sigcheck}$: We have that $T = \mathsf{ok}$ and

$$\hat{N} = \sigma(\mathsf{sigcheck}(x, \mathsf{sig}(v, \mathsf{sk}(y), x), \mathsf{pk}(y)))$$

for some substitution $\sigma$. Then it is easy to see that

$$\hat{N}_{|\mathsf{PCS}\to\mathsf{FAKE}} = \hat{\sigma}(\mathsf{sigcheck}(x, \mathsf{sig}(v, \mathsf{sk}(y), x), \mathsf{pk}(y)))$$

where $\hat{\sigma}(z) = \sigma(z)_{|\mathsf{PCS}\to\mathsf{FAKE}}$ for all $z \in \mathsf{dom}(\sigma)$. Thus, we have that

$$\hat{N}_{|\mathsf{PCS}\to\mathsf{FAKE}} \to_{\mathcal{H}} \mathsf{ok} = T_{|\mathsf{PCS}\to\mathsf{FAKE}}.$$

- $\mathsf{root}(\hat{N}) \in \{\mathsf{sver}, \mathsf{tpver}\}$: Analogous argument.

– $p = ip'$ for some $i$: We know that $N$ is no variable since $\mathcal{M} \subseteq \mathcal{T}(\Sigma)\downarrow$. Thus, $N$ is of the form $N = f(t'_1, \ldots, t'_l)$ for some $(\mathcal{S} \cup \mathcal{X})$-terms $t'_1, \ldots, t'_l$. We have that $\overline{N}$ is of the form $\overline{N} = f(\overline{t}_1, \ldots, \overline{t}_l)$, where $\overline{t}_j = t_j$ for $j \neq i$ and $t_i \to_{\mathcal{H}} \overline{t}_i$. By induction, we know that $t_{i|\mathsf{PCS}\to\mathsf{FAKE}} \xrightarrow{*}_{\mathcal{H}} \overline{t}_{i|\mathsf{PCS}\to\mathsf{FAKE}}$. By Lemma 13 we have that $\overline{N}$ is an $\mathcal{M}'$-instance of some $(\mathcal{S} \cup \mathcal{X})$-term $N'$ for some $\{\mathsf{sk}(A), \mathsf{sk}(T)\}$-free set $\mathcal{M}' \subseteq \mathcal{T}(\Sigma)\downarrow$. Since the size of $\overline{N}$ is smaller than the size of $\hat{N}$ (by definition of $\mathcal{H}$), by induction we have that

$$\overline{N}_{|\mathsf{PCS}\to\mathsf{FAKE}} \xrightarrow{*}_{\mathcal{H}} T_{|\mathsf{PCS}\to\mathsf{FAKE}}.$$

Using Lemma 14 it is easy to see that

$$\hat{N}_{|\mathsf{PCS}\to\mathsf{FAKE}} = f(t_{1|\mathsf{PCS}\to\mathsf{FAKE}}, \ldots, t_{l|\mathsf{PCS}\to\mathsf{FAKE}}).$$

Thus, we have that

$$\hat{N}_{|\mathsf{PCS}\to\mathsf{FAKE}} \to^{p}_{\mathcal{H}} \overline{N}_{|\mathsf{PCS}\to\mathsf{FAKE}} \xrightarrow{*}_{\mathcal{H}} T_{|\mathsf{PCS}\to\mathsf{FAKE}}.$$

□

The preceding lemmas said something about how replacing a term of the form $\mathsf{PCS}_r$ by a term of the form $\mathsf{FAKE}_{r'}$ in a message $m$ affect the instantiation of $(\mathcal{S} \cup \mathcal{X})$-terms and the application of $\mathcal{H}$-identities. The following three lemmas are their counterparts for the converse replacement, i.e., the replacement of a term of the form $\mathsf{FAKE}_{r'}$ by a term $\mathsf{PCS}_r$ and how this replacement affects instantiation of $(\mathcal{S} \cup \mathcal{X})$-terms and the application of $\mathcal{H}$-identities. The proofs of these lemmas is very similar to the corresponding preceding lemmas and are therefor omitted.

**Lemma 16.** *Let $r \in \mathcal{R}_{\mathcal{I}}$. Let $\mathcal{M} \subseteq \mathcal{T}(\Sigma)\downarrow$ be $\{r\}$-free. Let $N$ be an $(\mathcal{S} \cup \mathcal{X})$-term. Let $\hat{N}$ be an $\mathcal{M}$-instance of $N$. Let $T \in \mathcal{T}(\Sigma)$ such that $\hat{N} \to_{\mathcal{H}} T$. Then $T$ is an $\mathcal{M}'$-instance of some $(\mathcal{S} \cup \mathcal{X})$-term $N'$, where $\mathcal{M}' \subseteq \mathcal{T}(\Sigma)\downarrow$ is $\{r\}$-free.*

The following lemma corresponds to Lemma 14 and it is proven in the same way as Lemma 14.

**Lemma 17.** *Let $r \in \mathcal{T}(\Sigma)\downarrow$. Let $r' \in \mathcal{R}_\mathcal{I}$ Let $\mathcal{M} \subseteq \mathcal{T}(\Sigma)\downarrow$ be $\{r'\}$-free and $N$ an $\mathcal{S}$-term. Then for all $t_1, \ldots, t_n \in \mathcal{M}$ we have that*

$$N[t_1, \ldots, t_n/x_1, \ldots, x_n]_{|\mathsf{FAKE}_{r'} \to \mathsf{PCS}_r} = N[t_{1|\mathsf{FAKE}_{r'} \to \mathsf{PCS}_r}, \ldots, t_{n|\mathsf{FAKE}_{r'} \to \mathsf{PCS}_r}/x_1, \ldots, x_n],$$

*where $\{x_1, \ldots, x_n\} = \mathcal{V}(N)$.*

The following lemma corresponds to Lemma 18 and it is proven in the same way as Lemma 18.

**Lemma 18.** *Let $r \in \mathcal{T}(\Sigma)\downarrow$ and $r' \in \mathcal{R}_\mathcal{I}$. Let $\mathcal{M} \subseteq \mathcal{T}(\Sigma)\downarrow$ be $\{r'\}$-free. Then for all $(\mathcal{S} \cup \mathcal{X})$-terms $N$, $\mathcal{M}$-instances $\hat{N}$ of $N$, and $T \in \mathcal{T}(\Sigma)$ we have that $\hat{N} \xrightarrow{*}_\mathcal{H} T$ implies $\hat{N}_{|\mathsf{FAKE}_{r'} \to \mathsf{PCS}_r} \xrightarrow{*}_\mathcal{H} T_{|\mathsf{FAKE}_{r'} \to \mathsf{PCS}_r}$.*

Lemma 19 states that if one first replace a term of the form $\mathsf{FAKE}_{r'}$ by a term of the form $\mathsf{PCS}_r$ in a message $m$ and after this replace in the resulting message $\mathsf{PCS}_r$ by $\mathsf{FAKE}_{r'}$ one gets the original message $m$ if some properties are satisfied. The proof of this lemma is obvious.

**Lemma 19.** *Let $m \in \mathcal{T}(\Sigma)\downarrow$. Let $r \in \mathcal{R}_\mathcal{I}$ such that $r \notin Sub(m)$. Then for all $t \in \mathcal{T}(\Sigma)\downarrow$ we have that $(m_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}})_{|\mathsf{FAKE}_{r'} \to \mathsf{PCS}_r} = m$.*

The following lemma states for atomic $(\mathcal{S} \cup \mathcal{X})$-tests that under certain assumptions if a message $m$ passes $\theta$, then the message $m'$ that is constructed from $m$ by replacing a term of the form $\mathsf{PCS}_r$ by a term $\mathsf{FAKE}_{r'}$ (or vice versa) also passes $\theta$.

**Lemma 20.** *Let $m \in \mathcal{T}(\Sigma)\downarrow$. Let $\theta$ be an atomic $(\mathcal{S} \cup \mathcal{X})$-test. Let $r \in \mathcal{T}(\Sigma)\downarrow$ and let $r' \in \mathcal{R}_\mathcal{I}$.*

*a) If $\mathcal{X} \cup \{m\}$ is $\{\mathsf{sk}(A), \mathsf{sk}(T)\}$-free, then $m \models \theta$ implies $m_{|\mathsf{PCS}_r \to \mathsf{FAKE}-r'}\downarrow \models \theta$.*
*b) If $\mathcal{X} \cup \{m\}$ is $\{r'\}$-free, then $m \models \theta$ implies $m_{|\mathsf{FAKE}_{r'} \to \mathsf{PCS}_r}\downarrow \models \theta$.*

*Proof:* In the following we will write $\mathsf{PCS}$ instead of $\mathsf{PCS}_r$ and $\mathsf{FAKE}$ instead of $\mathsf{FAKE}_{r'}$.

a) Assume $m \models \theta$. It suffices to show that $\hat{m} = m_{|\mathsf{PCS} \to \mathsf{FAKE}}$ passes $\theta$. Let $M, N$ be simple $(\mathcal{S} \cup \mathcal{X})$-terms such that $\theta = (M, N)$. We have to show that

$$M[\hat{m}/x] \equiv_\mathcal{H} N[\hat{m}/x].$$

Since $m$ passes $\theta$ we know that

$$M[m/x] \equiv_\mathcal{H} N[m/x],$$

and hence, we know that $M[m/x] \xrightarrow{*}_\mathcal{H} m'$ and $N[m/x] \xrightarrow{*}_\mathcal{H} m'$ for some $m' \in \mathcal{T}(\Sigma)\downarrow$. By Lemma 15 we know that $M[m/x]_{|\mathsf{PCS} \to \mathsf{FAKE}} \xrightarrow{*}_\mathcal{H} m'_{|\mathsf{PCS} \to \mathsf{FAKE}}$ and $N[m/x]_{|\mathsf{PCS} \to \mathsf{FAKE}} \xrightarrow{*}_\mathcal{H} m'_{|\mathsf{PCS} \to \mathsf{FAKE}}$. Hence, we have $M[m/x]_{|\mathsf{PCS} \to \mathsf{FAKE}} \equiv_\mathcal{H} N[m/x]_{|\mathsf{PCS} \to \mathsf{FAKE}}$. By Lemma 14 we have $M[m/x]_{|\mathsf{PCS} \to \mathsf{FAKE}} = M[\hat{m}/x]$ and $N[m/x]_{|\mathsf{PCS} \to \mathsf{FAKE}} = N[\hat{m}/x]$. Thus, $\hat{m} \models \theta$.

b) Assume $m \models \theta$. It suffices to show that $\hat{m} = m_{|\mathsf{FAKE}\to\mathsf{PCS}}$ passes $\theta$. Let $M, N$ be simple $(\mathcal{S} \cup \mathcal{X})$-terms such that $\theta = (M, N)$. We have to show that

$$M[\hat{m}/x] \equiv_{\mathcal{H}} N[\hat{m}/x].$$

Since $m$ passes $\theta$ we know that

$$M[m/x] \equiv_{\mathcal{H}} N[m/x],$$

and hence, we know that $M[m/x] \xrightarrow{*}_{\mathcal{H}} m'$ and $N[m/x] \xrightarrow{*}_{\mathcal{H}} m'$ for some $m' \in \mathcal{T}(\Sigma)\downarrow$. By Lemma 18 we know that $M[m/x]_{|\mathsf{FAKE}\to\mathsf{PCS}} \xrightarrow{*}_{\mathcal{H}} m'_{|\mathsf{FAKE}\to\mathsf{PCS}}$ and $N[m/x]_{|\mathsf{FAKE}\to\mathsf{PCS}} \xrightarrow{*}_{\mathcal{H}} m'_{|\mathsf{FAKE}\to\mathsf{PCS}}$. Hence, we have $M[m/x]_{|\mathsf{FAKE}\to\mathsf{PCS}} \equiv_{\mathcal{H}} N[m/x]_{|\mathsf{FAKE}\to\mathsf{PCS}}$. By Lemma 17 we have $M[m/x]_{|\mathsf{FAKE}\to\mathsf{PCS}} = M[\hat{m}/x]$ and $N[m/x]_{|\mathsf{FAKE}\to\mathsf{PCS}} = N[\hat{m}/x]$. Thus, $\hat{m} \models \theta$.

□

Now, we can turn to the proof of Lemma 10.

*Proof of Lemma 10:* Let $\theta$ be an $(\mathcal{S} \cup \mathcal{X})$-test, $m \in \mathcal{T}(\Sigma)\downarrow$ such that $\mathcal{X} \cup \{m\}$ is $\{\mathsf{sk}(A), \mathsf{sk}(T)\}$-free, $r \in \mathcal{T}(\Sigma)\downarrow$ and $r' \in \mathcal{R}_{\mathcal{I}}$ such that $r' \notin \mathrm{Sub}(m)$.

Since $\theta$ is a (infinite) boolean combination of atomic $(\mathcal{S} \cup \mathcal{X})$-tests it suffices to show that for every atomic $(\mathcal{S} \cup \mathcal{X})$-test $\theta'$ we have we have that $m \models \theta'$ iff $m_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}} \models \theta'$. First, assume that $m \models \theta'$. Then, by Lemma 20a) we get $m_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}} \models \theta$.

Now, assume that $m_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}} \models \theta'$. Then, by Lemma 20b) we get

$$(m_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}})_{|\mathsf{FAKE}_{r'}\to\mathsf{PCS}_r} \models \theta'.$$

By Lemma 19 we have $(m_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}})_{|\mathsf{FAKE}_{r'}\to\mathsf{PCS}_r} = m$. □

**Proof of Lemma 11** The following Lemma is slightly more general than Lemma 11. It gives a sufficient condition for a set $E \subseteq \mathcal{T}(\Sigma)\downarrow$ that allows to conclude that whenever a message $m$ is derivable from $E$ the message $m'$ that is constructed from $m$ by replacing a term of the form $\mathsf{PCS}_r$ by a term $\mathsf{FAKE}_{r'}$ can be derived from $E_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}}$.

**Lemma 21.** *Let $E \subseteq \mathcal{T}(\Sigma)\downarrow$ be $\{\mathsf{sk}(A), \mathsf{sk}(T)\}$-free. Let $r \in \mathcal{T}(\Sigma)\downarrow$ and let $r' \in \mathcal{R}_{\mathcal{I}}$. Then, $m \in d_{\mathcal{S}}(E)\downarrow$ implies $m_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}}\downarrow \in d_{\mathcal{S}}(E_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}})$.*

**Proof:** Let $m \in d_{\mathcal{S}}(E)\downarrow$. Then there is an $\mathcal{S}$-term $N$ and an $E$-instance $\hat{N}$ of $N$ such that $\hat{N} \to^*_{\mathcal{H}} m$. According to Lemma 15 we have that $\hat{N}_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}} \to^*_{\mathcal{H}} m_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}}$ and using Lemma 14 we can conclude that $\hat{N}_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}}$ is an $E_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}}$-instance of $N$, so $m_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}} \in d_{\mathcal{S}}(E_{|\mathsf{PCS}_r\to\mathsf{FAKE}_{r'}})$. □

Now we can prove Lemma 11.

*Proof of Lemma 11:*

a) Let $m \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\})$. Since for $K = \mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\}$ and $t \in \{A, T\}$ condition 2) of Lemma 12 is satisfied, by Lemma 12 we obtain that $\mathsf{sk}(A), \mathsf{sk}(T) \notin d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\})$, and hence, $\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{PCS}_r\}$ is $\{\mathsf{sk}(A), \mathsf{sk}(T)\}$-free. By Lemma 21, we have that $m_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}} \downarrow \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}} \cup \{\mathsf{FAKE}_{r'}\})$. Since $\mathsf{FAKE}_{r'} \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}})$ we have that $m_{|\mathsf{PCS}_r \to \mathsf{FAKE}_{r'}} \downarrow \in d_{\mathcal{S}}(\mathcal{K}_{\mathcal{I}})$.
b) Similar argument as in a). □

## 8 Conclusion

We have proposed a new definition of abuse-freeness which involves as key features (i) a specifically designed notion of test performed by the outside party and (ii) a formalization of the assumptions of the outside party by the notion of external view. We have applied our definition to the ASM and GJM protocol, where for the latter protocol we have developed an equational theory to describe the semantics of private contract signatures.

In view of the results in [1, 8, 11], an interesting question is whether abuse-freeness as defined in this paper is decidable.

## References

1. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. In *Proceedings of the 31st International Colloqium on Automata, Languages, and Programming (ICALP 2004)*, volume 3142 of *Lecture Notes in Computer Science*, pages 46–58. Springer, 2004.
2. M. Abadi and C. Fournet. Mobile Values, New Names, and Secure Communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages (POPL 2001)*, pages 104–115. ACM Press, 2001.
3. N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99, 1998.
4. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
5. R. Chadha, M.I. Kanovich, and A.Scedrov. Inductive methods and contract-signing protocols. In P. Samarati, editor, *8-th ACM Conference on Computer and Communications Security (CCS 2001)*, pages 176–185. ACM Press, 2001.
6. R. Chadha, S. Kremer, and A. Scedrov. Formal analysis of multi-party contract signing. In R. Focardi, editor, *17th IEEE Computer Security Foundations Workshop (CSFW-17)*, pages 266–279. IEEE Computer Society Press, 2004.
7. R. Chadha, J.C. Mitchell, A. Scedrov, and V. Shmatikov:. Contract Signing, Optimism, and Advantage. In R.M. Amadio and D. Lugiez, editors, *CONCUR 2003 - Concurrency Theory, 14th International Conference*, volume 2761 of *Lecture Notes in Computer Science*, pages 361–377. Springer, 2003.
8. Y. Chevalier and M. Rusinowitch. Combining Intruder Theories. In *Proceedings of the 32st International Colloqium on Automata, Languages, and Programming (ICALP 2005)*, volume 3580 of *Lecture Notes in Computer Science*, pages 639–651. Springer, 2005.

9. D. Dolev and A.C. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
10. J.A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Advances in Cryptology – CRYPTO'99, 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 449–466. Springer-Verlag, 1999.
11. D. Kähler, R. Küsters, and Th. Wilke. Deciding Properties of Contract-Signing Protocols. In Volker Diekert and Bruno Durand, editors, *Proceedings of the 22nd Symposium on Theoretical Aspects of Computer Science (STACS 2005)*, number 3404 in Lecture Notes in Computer Science, pages 158–169. Springer-Verlag, 2005.
12. S. Kremer and J.-F. Raskin. Game analysis of abuse-free contract signing. In *Computer Security Foundations Workshop 2002 (CSFW 2002)*, pages 206–220. IEEE Computer Society, 2002.
13. V. Shmatikov and J.C. Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science (TCS), special issue on Theoretical Foundations of Security Analysis and Design*, 283(2):419–450, 2002.

// decide nondeterministically whether Alice wants to participate in the protocol or not
$\mathsf{state}_A = 0$:

- $\mathsf{state}_A = 0$
- $\mathsf{state}_A = \mathsf{initiator}$

// Alice can decide to send her promise of signature to Bob at any time
$\mathsf{state}_A = \mathsf{initiator}$:

- $\mathsf{state}_A = \mathsf{initiator}$
- $\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$
  $N_A = \mathsf{newnonce}()$
  $m_1 = \langle \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{pk}(T), \mathsf{text}, \mathsf{hash}(N_A) \rangle$
  $s_1 = \mathsf{sig}(\mathsf{sk}(A), m_1)$
  $me_1 = \langle m_1, s_1 \rangle$
  $\mathrm{netin}_B^A = me_1$

// receive promise of signature from Bob
$\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$ and
$\pi_1(\pi_1(\mathrm{netout}_A^B)) = me_1$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{netout}_A^B), \pi_2(\mathrm{netout}_A^B), \mathsf{pk}(B)) = \mathsf{ok}$:

- $\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$
  $me_2 = \mathrm{netout}_A^B$
  $\mathrm{netin}_B^A = N_A$

// receive signature from Bob
$\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$ and
$\mathsf{hash}(\mathrm{netout}_A^B) = \pi_2(\pi_1(me_2))$:

- $\mathsf{state}_A = (i, \mathsf{contract})$

// decide to abort
$\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$ and
( $\pi_1(\pi_1(\mathrm{netout}_A^B)) \neq me_1$ or
$\mathsf{sigcheck}(\pi_1(\mathrm{netout}_A^B), \pi_2(\mathrm{netout}_A^B), \mathsf{pk}(B)) \neq \mathsf{ok}$: )

- $\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$
- $\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$
  $m_2 = \langle \mathsf{aborted}, me_1 \rangle$
  $ma_1 = \langle m_2, \mathsf{sig}(\mathsf{sk}(A), m_2) \rangle$
  $\mathrm{sec}_T^A = ma_1$

// receive abort confirmation from TTP
$\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$ and
$\pi_1(\mathrm{sec}_A^T) = \langle \mathsf{aborted}, ma_1 \rangle$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{sec}_A^T), \pi_2(\mathrm{sec}_A^T), \mathsf{pk}(T))$

- $\mathsf{state}_A = (i, \mathsf{aborted})$

// receive replacement contract from TTP
$\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$ and
$\pi_1(\pi_1(\mathrm{sec}_A^T)) = me_1$ and
$\pi_1(\pi_1(\pi_2(\pi_1(\mathrm{sec}_A^T)))) = me_1$ and
$\mathsf{sigcheck}(\pi_1(\pi_2(\pi_1(\mathrm{sec}_A^T))), \pi_2(\pi_2(\pi_1(\mathrm{sec}_A^T))), \mathsf{pk}(B)) = ok$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{sec}_A^T), \pi_2(\mathrm{sec}_A^T), \mathsf{pk}(T)) = \mathsf{ok}$

- $\mathsf{state}_A = (i, \mathsf{resolved})$

// resolve request
$\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$ and
$\mathsf{hash}(\mathrm{netout}_A^B) \neq \pi_2(\pi_1(me_2))$:

- $\mathsf{state}_A = (i, \mathsf{resolve\text{-}req\text{-}sent})$
  $\mathrm{sec}_T^A = \langle me_1, me_2 \rangle$

// receive replacement contract from TTP
$\mathsf{state}_A = (i, \mathsf{resolve\text{-}req\text{-}sent})$ and
$\pi_1(\mathrm{sec}_A^T) = \langle me_1, me_2 \rangle$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{sec}_A^T), \pi_2(\mathrm{sec}_A^T), \mathsf{pk}(T)) = ok$

- $\mathsf{state}_A = (i, \mathsf{resolved})$

**Fig. 1.** This is the pseudocode description of Alice in the ASW-protocol

| status for instance $\langle \mathsf{text}, \mathsf{pk}(A), \mathsf{pk}(B) \rangle$ | from $A$ | from $B$ | status for instance $\langle \mathsf{text}, \mathsf{pk}(A), \mathsf{pk}(B) \rangle$ | to $A$ | to $B$ | DB-action for instance $\langle \mathsf{text}, \mathsf{pk}(A), \mathsf{pk}(B) \rangle$ |
|---|---|---|---|---|---|---|
| $\perp$ | ABabort | $\circ$ | aborted | ABabort-ack | $\circ$ | store ABabort-ack |
| $\perp$ | ABresA | $\circ$ | resolved | ABrepl-contract | $\circ$ | store ABrepl-contract |
| $\perp$ | $\circ$ | ABresB | resolved | $\circ$ | ABrepl-contract | store ABrepl-contract |
| $\perp$ | ABabort | ABresB | resolved | ABrepl-contract | ABrepl-contract | store ABrepl-contract |
| $\perp$ | ABresA | ABresB | resolved | ABrepl-contract | ABrepl-contract | store ABrepl-contract |
| aborted | ABabort | $\circ$ | aborted | stored value | $\circ$ | |
| aborted | ABresA | $\circ$ | aborted | stored value | $\circ$ | |
| aborted | $\circ$ | ABresB | aborted | $\circ$ | stored value | |
| aborted | ABabort | ABresB | aborted | stored value | stored value | |
| aborted | ABresA | ABresB | aborted | stored value | stored value | |
| resolved | ABabort | $\circ$ | resolved | stored value | $\circ$ | |
| resolved | ABresA | $\circ$ | resolved | stored value | $\circ$ | |
| resolved | $\circ$ | ABresB | resolved | $\circ$ | stored value | |
| resolved | ABabort | ABresB | resolved | stored value | stored value | |
| resolved | ABresA | ABresB | resolved | stored value | stored value | |

**Fig. 2.** This is TTP of the ASW-protocol described as a transition table

// decide nondeterministically whether Alice wants to participate in the protocol and in which role
$\mathsf{state}_A = 0$:

- $\mathsf{state}_A = 0$
- $\mathsf{state}_A = \mathsf{responder}$
  $\mathrm{netin}_B^A = \mathsf{responder}$
- $\mathsf{state}_A = \mathsf{initiator}$
  $\mathrm{netin}_B^A = \mathsf{initiator}$

// Alice can decide to send her promise of signature to Bob at any time
$\mathsf{state}_A = \mathsf{initiator}$:

- $\mathsf{state}_A = \mathsf{initiator}$
- $\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$
  $N_A = \mathsf{newnonce}()$
  $m_1 = \langle \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{pk}(T), \mathsf{text}, \mathsf{hash}(N_A) \rangle$
  $s_1 = \mathsf{sig}(\mathsf{sk}(A), m_1)$
  $me_1 = \langle m_1, s_1 \rangle$
  $\mathrm{netin}_B^A = me_1$

// receive promise of signature from Bob
$\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$ and
$\pi_1(\pi_1(\mathrm{netout}_A^B)) = me_1$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{netout}_A^B), \pi_2(\mathrm{netout}_A^B), \mathsf{pk}(B)) = \mathsf{ok}$:

- $\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$
  $me_2 = \mathrm{netout}_A^B$
  $\mathrm{netin}_B^A = N_A$

// receive signature from Bob
$\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$ and
$\mathsf{hash}(\mathrm{netout}_A^B) = \pi_2(\pi_1(me_2))$:

- $\mathsf{state}_A = (i, \mathsf{contract})$

// decide to abort
$\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$ and
( $\pi_1(\pi_1(\mathrm{netout}_A^B)) \neq me_1$ or
$\mathsf{sigcheck}(\pi_1(\mathrm{netout}_A^B), \pi_2(\mathrm{netout}_A^B), \mathsf{pk}(B)) \neq \mathsf{ok}$: )

- $\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$
- $\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$
  $m_2 = \langle \mathsf{aborted}, me_1 \rangle$
  $ma_1 = \langle m_2, \mathsf{sig}(\mathsf{sk}(A), m_2) \rangle$
  $\mathrm{sec}_T^A = ma_1$

// receive abort confirmation from TTP
$\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$ and
$\pi_1(\mathrm{sec}_A^T) = \langle \mathsf{aborted}, ma_1 \rangle$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{sec}_A^T), \pi_2(\mathrm{sec}_A^T), \mathsf{pk}(T))$

- $\mathsf{state}_A = (i, \mathsf{aborted})$

// receive replacement contract from TTP
$\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$ and
$\pi_1(\pi_1(\mathrm{sec}_A^T)) = me_1$ and
$\pi_1(\pi_1(\pi_2(\pi_1(\mathrm{sec}_A^T)))) = me_1$ and
$\mathsf{sigcheck}(\pi_1(\pi_2(\pi_1(\mathrm{sec}_A^T))), \pi_2(\pi_2(\pi_1(\mathrm{sec}_A^T))), \mathsf{pk}(B)) = ok$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{sec}_A^T), \pi_2(\mathrm{sec}_A^T), \mathsf{pk}(T)) = \mathsf{ok}$

- $\mathsf{state}_A = (i, \mathsf{resolved})$

// resolve request
$\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$ and
$\mathsf{hash}(\mathrm{netout}_A^B) \neq \pi_2(\pi_1(me_2))$:

- $\mathsf{state}_A = (i, \mathsf{resolve\text{-}req\text{-}sent})$
  $\mathrm{sec}_T^A = \langle me_1, me_2 \rangle$

// receive replacement contract from TTP
$\mathsf{state}_A = (i, \mathsf{resolve\text{-}req\text{-}sent})$ and
$\pi_1(\mathrm{sec}_A^T) = \langle me_1, me_2 \rangle$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{sec}_A^T), \pi_2(\mathrm{sec}_A^T), \mathsf{pk}(T)) = ok$

- $\mathsf{state}_A = (i, \mathsf{resolved})$

**Fig. 3.** This is the pseudocode description of Alice in the ASW-protocol as assumed by Charlie (initiator part)

// Alice responds to promise of signature from Bob in the responder role
$\mathsf{state}_A = \mathsf{responder}$ and
$\pi_1(\pi_1(\mathrm{netout}_A^B)) = \mathsf{pk}(B)$ and
$\pi_1(\pi_2(\pi_1(\mathrm{netout}_A^B))) = \mathsf{pk}(A)$ and
$\pi_1(\pi_2(\pi_2(\pi_1(\mathrm{netout}_A^B)))) = \mathsf{pk}(T)$ and
$\pi_1(\pi_2(\pi_2(\pi_2(\pi_1(\mathrm{netout}_A^B))))) = \mathsf{text}$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{netout}_A^B), \pi_2(\mathrm{netout}_A^B), \mathsf{pk}(B)) = \mathsf{ok}$

- $\mathsf{state}_A = \mathsf{not\text{-}interested}$
- $\mathsf{state}_A = (r, \mathsf{pos\text{-}sent})$
  $me_1 = \mathrm{netout}_A^B$
  $N_A = \mathsf{newnonce}()$
  $m_1 = \langle me_1, \mathsf{hash}(N_A)\rangle$
  $s_1 = \mathsf{sig}(\mathsf{sk}(A), m_1)$
  $me_2 = \langle m_1, s_1\rangle$
  $\mathrm{netin}_B^A = me_2$

// receive signature from Bob and send own signature
$\mathsf{state}_A = (r, \mathsf{pos\text{-}sent})$ and
$\pi_2(\pi_2(\pi_2(\pi_2(\pi_1(me_1))))) = \mathsf{hash}(\mathrm{netout}_A^B)$

- $\mathsf{state}_A = (r, \mathsf{contract})$

// send resolve request
$\mathsf{state}_A = (r, \mathsf{pos\text{-}sent})$ and
$\pi_2(\pi_2(\pi_2(\pi_2(\pi_1(me_1))))) \neq \mathsf{hash}(\mathrm{netout}_A^B)$

- $\mathsf{state}_A = (r, \mathsf{pos\text{-}sent})$
- $\mathsf{state}_A = (r, \mathsf{resolve\text{-}req\text{-}sent})$
  $m_2 = \langle me_1, me_2\rangle$
  $s_2 = \mathsf{sig}(\mathsf{sk}(A), m_2)$
  $mr_1 = \langle m_2, s_2\rangle$
  $\mathrm{sec}_T^A = mr_1$

// receive replacement contract
$\mathsf{state}_A = (r, \mathsf{resolve\text{-}req\text{-}sent})$ and
$\pi_1(\mathrm{sec}_A^T) = m_2$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{sec}_A^T), \pi_2(\mathrm{sec}_A^T), \mathsf{pk}(T)) = \mathsf{ok}$

- $\mathsf{state}_A = (r, \mathsf{resolved})$

// receive abort message
$\mathsf{state}_A = (r, \mathsf{resolve\text{-}req\text{-}sent})$ and
$\pi_1(\pi_1(\mathrm{sec}_A^T)) = \mathsf{aborted}$ and
$\mathsf{sigcheck}(\pi_1(\pi_2(\pi_1(\mathrm{sec}_A^T))), \pi_2(\pi_2(\pi_1(\mathrm{sec}_A^T))), \mathsf{pk}(T)) = \mathsf{ok}$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{sec}_A^T), \pi_2(\mathrm{sec}_A^T), \mathsf{pk}(T)) = \mathsf{ok}$

- $\mathsf{state}_A = (r, \mathsf{aborted})$

**Fig. 4.** This is the pseudocode description of Alice in the ASW-protocol as assumed by Charlie (responder part)

// Alice decides she wants to participate in the protocol or not
$\mathsf{state}_A = 0$:

- $\mathsf{state}_A = 0$
- $\mathsf{state}_A = \mathsf{initiator}$

// Alice can decide to send her promise of signature to Bob at any time
$\mathsf{state}_A = \mathsf{initiator}$:

- $\mathsf{state}_A = \mathsf{initiator}$
- $\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$
  $m_1 = \langle \mathsf{c}, 1 \rangle$
  $p_1 = \mathsf{pcs}(\mathsf{random}(), \mathsf{sk}(A), m_1, \mathsf{pk}(B), \mathsf{pk}(T))$
  $me_1 = \langle m_1, p_1 \rangle$
  $\mathrm{netin}_B^A = me_1$

// Alice receives promise of signature from Bob and sends her signature
$\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$ and
$\mathsf{pcsver}(\pi_1(\mathrm{netout}_A^B), \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{netout}_A^B)) = \mathsf{pcsok}$ and $\pi_1(\mathrm{netout}_A^B) = \langle \mathsf{c}, 2 \rangle$:

- $\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$
  $me_2 = \mathrm{netout}_A^B$
  $m_2 = \langle \mathsf{c}, 1 \rangle$
  $s_2 = \mathsf{sconvert}(\mathsf{random}(), \mathsf{sk}(A), p_1)$
  $me_3 = \langle m_2, s_2 \rangle$
  $\mathrm{netin}_B^A = me_3$

// Alice receives signature from Bob
$\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$ and
$\mathsf{sver}(\pi_1(\mathrm{netout}_A^B), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{netout}_A^B)) = \mathsf{sok}$ and $\pi_1(\mathrm{netout}_A^B) = \langle \mathsf{c}, 2 \rangle$:

- $\mathsf{state}_A = (i, \mathsf{contract})$

// Alice sends abort request to TTP
$\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$ and
$(\mathsf{pcsver}(\pi_1(\mathrm{netout}_A^B), \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{netout}_A^B)) \neq \mathsf{pcsok}$ or $\pi_1(\mathrm{netout}_A^B) \neq \langle \mathsf{c}, 2 \rangle)$:

- $\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$
- $\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$
  $m_3 = \langle \mathsf{text}, \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{aborted} \rangle$
  $s_3 = \mathsf{sig}(\mathsf{random}(), \mathsf{sk}(A), m_3)$
  $ma_1 = \langle m_3, s_3 \rangle$
  $\mathrm{sec}_T^A = ma_1$

// Alice receives abort confirmation from TTP
$\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{sec}_A^T), \pi_2(\mathrm{sec}_A^T), \mathsf{pk}(T)) = \mathsf{ok}$ and $\pi_1(\mathrm{sec}_A^T) = ma_1$

- $\mathsf{state}_A = (i, \mathsf{aborted})$

// Alice receives resolved contract from TTP in response to abort request
$\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$ and
$\mathsf{sver}(\pi_1(\mathrm{sec}_A^T), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{sec}_A^T)) = \mathsf{sok}$ and $\pi_1(\mathrm{sec}_A^T) = \langle \mathsf{c}, 2 \rangle$

- $\mathsf{state}_A = (i, \mathsf{resolved})$

// Alice sends resolve request to TTP
$\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$ and
$(\mathsf{sver}(\pi_1(\mathrm{netout}_A^B), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{netout}_A^B)) \neq \mathsf{sok}$ or $\pi_1(\mathrm{netout}_A^B) \neq \langle \mathsf{c}, 2 \rangle)$:

- $\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$
- $\mathsf{state}_A = (i, \mathsf{resolve\text{-}req\text{-}sent})$
  $m_4 = \langle \mathsf{c}, 1 \rangle$
  $s_4 = \mathsf{sconvert}(\mathsf{random}(), \mathsf{sk}(A), p_1)$
  $mr_1 = \langle m_4, s_4, \pi_2(me_2) \rangle$
  $\mathrm{sec}_T^A = mr_1$

// Alice receives resolved contract from TTP
$\mathsf{state}_A = (i, \mathsf{resolve\text{-}req\text{-}sent})$ and
$((\mathsf{sver}(\pi_1(\mathrm{sec}_A^T), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{sec}_A^T)) = \mathsf{sok}$ and $\pi_1(\mathrm{sec}_A^T) = \langle \mathsf{c}, 2 \rangle)$ or
$(\mathsf{tpver}(\pi_1(\mathrm{sec}_A^T), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{sec}_A^T)) = \mathsf{tpok}$ and $\pi_1(\mathrm{sec}_A^T) = \langle \mathsf{c}, 2 \rangle))$

- $\mathsf{state}_A = (i, \mathsf{resolved})$

**Fig. 5.** This is the pseudocode description of Alice (initiator), in the GJM-protocol

| status for instance $\langle \mathsf{text}, \mathsf{pk}(A), \mathsf{pk}(B) \rangle$ | from $A$ | from $B$ | status for instance $\langle \mathsf{text}, \mathsf{pk}(A), \mathsf{pk}(B) \rangle$ | to $A$ | to $B$ | DB-action for instance $\langle \mathsf{text}, \mathsf{pk}(A), \mathsf{pk}(B) \rangle$ |
|---|---|---|---|---|---|---|
| $\perp$ | ABabort | $\circ$ | aborted | ABabort-ack | $\circ$ | store ABabort-ack |
| $\perp$ | ABresA | $\circ$ | resolved | ABttpB | $\circ$ | store ABcontractA |
| $\perp$ | $\circ$ | ABresB | resolved | $\circ$ | ABttpA | store ABcontractB |
| $\perp$ | ABabort | ABresB | resolved | ABcontractB | ABcontractA | store ABcontractB |
| $\perp$ | ABresA | ABresB | resolved | ABcontractB | ABcontractA | store ABcontractB |
| aborted | ABabort | $\circ$ | aborted | stored value | $\circ$ | |
| aborted | ABresA | $\circ$ | aborted | stored value | $\circ$ | |
| aborted | $\circ$ | ABresB | aborted | $\circ$ | stored value | |
| aborted | ABabort | ABresB | aborted | stored value | stored value | |
| aborted | ABresA | ABresB | aborted | stored value | stored value | |
| resolved | ABabort | $\circ$ | resolved | stored value | $\circ$ | |
| resolved | ABresA | $\circ$ | resolved | stored value | $\circ$ | |
| resolved | $\circ$ | ABresB | resolved | $\circ$ | stored value | |
| resolved | ABabort | ABresB | resolved | stored value | stored value | |
| resolved | ABresA | ABresB | resolved | stored value | stored value | |

**Fig. 6.** This is the description of the TTP in the GJM-protocol described as a transition table

// decide nondeterministically whether Alice wants to participate in the protocol and in which role
$\mathsf{state}_A = 0$:

- $\mathsf{state}_A = 0$
- $\mathsf{state}_A = \mathsf{responder}$
  $\mathrm{netin}_B^A = \mathsf{responder}$
- $\mathsf{state}_A = \mathsf{initiator}$
  $\mathrm{netin}_B^A = \mathsf{initiator}$

// Alice can decide to send her promise of signature to Bob at any time
$\mathsf{state}_A = \mathsf{initiator}$:

- $\mathsf{state}_A = \mathsf{initiator}$
- $\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$
  $m_1 = \langle \mathsf{c}, 1 \rangle$
  $p_1 = \mathsf{pcs}(\mathsf{random}(), \mathsf{sk}(A), m_1, \mathsf{pk}(B), \mathsf{pk}(T))$
  $me_1 = \langle m_1, p_1 \rangle$
  $\mathrm{netin}_B^A = me_1$

// Alice receives promise of signature from Bob and sends her signature
$\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$ and
$\mathsf{pcsver}(\pi_1(\mathrm{netout}_A^B), \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{netout}_A^B)) = \mathsf{pcsok}$ and $\pi_1(\mathrm{netout}_A^B) = \langle \mathsf{c}, 2 \rangle$:

- $\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$
  $me_2 = \mathrm{netout}_A^B$
  $m_2 = \langle \mathsf{c}, 1 \rangle$
  $s_2 = \mathsf{sconvert}(\mathsf{random}(), \mathsf{sk}(A), p_1)$
  $me_3 = \langle m_2, s_2 \rangle$
  $\mathrm{netin}_B^A = me_3$

// Alice receives signature from Bob
$\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$ and
$\mathsf{sver}(\pi_1(\mathrm{netout}_A^B), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{netout}_A^B)) = \mathsf{sok}$ and $\pi_1(\mathrm{netout}_A^B) = \langle \mathsf{c}, 2 \rangle$:

- $\mathsf{state}_A = (i, \mathsf{contract})$

// Alice sends abort request to TTP
$\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$ and
$(\mathsf{pcsver}(\pi_1(\mathrm{netout}_A^B), \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{netout}_A^B)) \neq \mathsf{pcsok}$ or $\pi_1(\mathrm{netout}_A^B) \neq \langle \mathsf{c}, 2 \rangle)$:

- $\mathsf{state}_A = (i, \mathsf{pos\text{-}sent})$
- $\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$
  $m_3 = \langle \mathsf{text}, \mathsf{pk}(A), \mathsf{pk}(B), \mathsf{aborted} \rangle$
  $s_3 = \mathsf{sig}(\mathsf{random}(), \mathsf{sk}(A), m_3)$
  $ma_1 = \langle m_3, s_3 \rangle$
  $\mathrm{sec}_T^A = ma_1$

// Alice receives abort confirmation from TTP
$\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{sec}_A^T), \pi_2(\mathrm{sec}_A^T), \mathsf{pk}(T)) = \mathsf{ok}$ and $\pi_1(\mathrm{sec}_A^T) = ma_1$

- $\mathsf{state}_A = (i, \mathsf{aborted})$

// Alice receives resolved contract from TTP in response to abort request
$\mathsf{state}_A = (i, \mathsf{abort\text{-}req\text{-}sent})$ and
$\mathsf{sver}(\pi_1(\mathrm{sec}_A^T), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{sec}_A^T)) = \mathsf{sok}$ and $\pi_1(\mathrm{sec}_A^T) = \langle \mathsf{c}, 2 \rangle$

- $\mathsf{state}_A = (i, \mathsf{resolved})$

// Alice sends resolve request to TTP
$\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$ and
$(\mathsf{sver}(\pi_1(\mathrm{netout}_A^B), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{netout}_A^B)) \neq \mathsf{sok}$ or $\pi_1(\mathrm{netout}_A^B) \neq \langle \mathsf{c}, 2 \rangle)$:

- $\mathsf{state}_A = (i, \mathsf{sig\text{-}sent})$
- $\mathsf{state}_A = (i, \mathsf{resolve\text{-}req\text{-}sent})$
  $m_4 = \langle \mathsf{c}, 1 \rangle$
  $s_4 = \mathsf{sconvert}(\mathsf{random}(), \mathsf{sk}(A), p_1)$
  $mr_1 = \langle m_4, s_4, \pi_2(me_2) \rangle$
  $\mathrm{sec}_T^A = mr_1$

// Alice receives resolved contract from TTP
$\mathsf{state}_A = (i, \mathsf{resolve\text{-}req\text{-}sent})$ and
$((\mathsf{sver}(\pi_1(\mathrm{sec}_A^T), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{sec}_A^T)) = \mathsf{sok}$ and $\pi_1(\mathrm{sec}_A^T) = \langle \mathsf{c}, 2 \rangle)$ or
$(\mathsf{tpver}(\pi_1(\mathrm{sec}_A^T), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{sec}_A^T)) = \mathsf{tpok}$ and $\pi_1(\mathrm{sec}_A^T) = \langle \mathsf{c}, 2 \rangle))$

- $\mathsf{state}_A = (i, \mathsf{resolved})$

**Fig. 7.** This is the pseudocode description of Alice in the GJM-protocol as assumed by Charlie (initiator part)

// Alice responds to promise of signature from Bob in the responder role
$\mathsf{state}_A = \mathsf{responder}$ and
$\mathsf{pcsver}(\pi_1(\mathrm{netout}_A^B), \mathsf{pk}(B), \mathsf{pk}(A), \mathsf{pk}(T), \pi_2(\mathrm{netout}_A^B)) = \mathsf{pcsok}$ and $\pi_1(\mathrm{netout}_A^B) = \langle \mathsf{c}, 1\rangle$:

- $\mathsf{state}_A = \mathsf{not\text{-}interested}$
- $\mathsf{state}_A = (r, \mathsf{pos\text{-}sent})$
  $me_1 = \mathrm{netout}_A^B$
  $m_1 = \langle \mathsf{c}, 2\rangle$
  $p_1 = \mathsf{pcs}(\mathsf{random}(), \mathsf{sk}(A), m_1, \mathsf{pk}(B), \mathsf{pk}(T))$
  $me_2 = \langle m_1, p_1\rangle$
  $\mathrm{netin}_B^A = me_2$

// Alice receives signature from Bob and sends her own signature in response
$\mathsf{state}_A = (r, \mathsf{pos\text{-}sent})$ and
$\mathsf{sver}(\pi_1(\mathrm{netout}_A^B), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{netout}_A^B)) = \mathsf{sok}$ and $\pi_1(\mathrm{netout}_A^B) = \langle \mathsf{c}, 1\rangle$:

- $\mathsf{state}_A = (r, \mathsf{contract})$
  $m_2 = \langle \mathsf{c}, 1\rangle$
  $s_2 = \mathsf{sconvert}(\mathsf{random}(), \mathsf{sk}(A), p_1)$
  $me_4 = \langle m_2, s_2\rangle$
  $\mathrm{netin}_B^A = me_4$

// Alice sends resolve request to TTP
$\mathsf{state}_A = (r, \mathsf{pos\text{-}sent})$ and
$(\mathsf{sver}(\pi_1(\mathrm{netout}_A^B), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{netout}_A^B)) \neq \mathsf{sok}$ or $\pi_1(\mathrm{netout}_A^B) \neq \langle \mathsf{c}, 1\rangle)$:

- $\mathsf{state}_A = (r, \mathsf{pos\text{-}sent})$
- $\mathsf{state}_A = (r, \mathsf{resolve\text{-}req\text{-}sent})$
  $m_3 = \langle \mathsf{c}, 1\rangle$
  $s_3 = \mathsf{sconvert}(\mathsf{random}, \mathsf{sk}(A), p_1)$
  $mr_1 = \langle m_3, s_3, \pi_2(me_1)\rangle$
  $\mathrm{sec}_T^A = mr_1$

// Alice receives resolved contract from TTP
$\mathsf{state}_A = (r, \mathsf{resolve\text{-}req\text{-}sent})$ and
$((\mathsf{sver}(\pi_1(\mathrm{sec}_A^T), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{secout}_A^T)) = \mathsf{sok}$ and $\pi_1(\mathrm{sec}_A^T) = \langle \mathsf{c}, 1\rangle)$ or
$(\mathsf{tpver}(\pi_1(\mathrm{sec}_A^T), \mathsf{pk}(B), \mathsf{pk}(T), \pi_2(\mathrm{sec}_A^T)) = \mathsf{tpok}$ and $\pi_1(\mathrm{sec}_A^T) = \langle \mathsf{c}, 1\rangle))$

- $\mathsf{state}_A = (r, \mathsf{resolved})$

// Alice receives abort acknowlege message from TTP in response to her resolve request
$\mathsf{state}_A = (r, \mathsf{resolve\text{-}req\text{-}sent})$ and
$\mathsf{sigcheck}(\pi_1(\mathrm{sec}_A^T), \pi_2(\mathrm{sec}_A^T), \mathsf{pk}(T)) = \mathsf{ok}$ and
$\pi_1(\pi_1(\pi_1(\mathrm{sec}_A^T))) = \mathsf{text}$ and
$\pi_1(\pi_2(\pi_1(\pi_1(\mathrm{sec}_A^T)))) = \mathsf{pk}(B)$ and
$\pi_1(\pi_2(\pi_2(\pi_1(\pi_1(\mathrm{sec}_A^T))))) = \mathsf{pk}(A)$ and
$\pi_2(\pi_2(\pi_2(\pi_1(\pi_1(\mathrm{sec}_A^T))))) = \mathsf{aborted}$ and
$\mathsf{sigcheck}(\pi_1(\pi_1(\mathrm{sec}_A^T)), \pi_2(\pi_1(\mathrm{sec}_A^T)), \mathsf{pk}(B)) = \mathsf{ok}$

- $\mathsf{state}_A = (r, \mathsf{aborted})$

**Fig. 8.** This is the pseudocode description of Alice in the GJM-protocol as assumed by Charlie (responder part)