Interaction and causality in digital signature exchange protocols

Jonathan Hayman *

Computer Laboratory, University of Cambridge, UK

Abstract. Causal reasoning is a powerful tool in analysing security protocols, as seen in the popularity of the strand space model. However, for protocols that branch, more subtle models are called for to capture the ways in which they can interact with a possibly malign environment. We study a model for security protocols encompassing causal reasoning and interaction, developing a semantics for a simple security protocol language based on concurrent games played on event structures. We show how it supports causal reasoning about a protocol for secure digital signature exchange. The semantics paves the way for the application of more sophisticated forms of concurrent game, for example including symmetry and probability, to the analysis of security protocols.

1 Introduction

The use of models that explicitly represent *causality* has proved highly useful in the analysis of security protocols, as seen in the importance of the strand space model [1]. Through the representation of causal dependency, it is possible to perform an analogue of Paulson's inductive method [2] to establish safety properties by deriving a contradiction to the existence of an earliest violating event, and analyses of causal dependency clarify the specification and proof of authentication properties.

Strand spaces as originally presented represent processes as sets of traces, thereby losing information on branching; this led to Crazzolara and Winskel's development of the semantics based on event structures and Petri nets of a security protocol language called SPL [3]. For the analysis of kinds of security protocol other than authentication protocols, and in particular for the analysis of digital signature exchange protocols, the ability of participants to choose between different forms of behaviour is at the core of the protocols. A key issue then becomes the representation of which participant forces the interaction down different branches: whether it is the process following the protocol or an adversary is of central importance, and this leads to the natural specification of their correctness properties in *game-based* models [4], albeit in past work at the price of no longer representing causality.

* The support of the ERC through the Advanced Grant ECSYM is gratefully acknowledged

In this paper, we study the application of *concurrent games* [5] in reasoning about a digital signature exchange protocol, demonstrating how the model supports *both* causal and game-based reasoning. This represents an exciting starting point where other features from the more abstract world of concurrent games, such as probability [6] and symmetry [7], might be brought to bear.

In more detail, we modestly extend SPL and use it to present the Asokan-Shoup-Waidner protocol [8] for digital signature exchange. We then present the semantics of SPL using concurrent games, showing how adversarial behaviour is constrained to the Dolev-Yao model. We conclude by studying the key correctness properties of the protocol.

2 SPL: A security protocol language

SPL, standing for Security Protocol Language, is introduced in [3] along with a semantics based on Petri nets. The work shows how a causal semantics, using a well-known model for concurrency, can be given to syntactically-represented processes that supports the kind of causal reasoning important in the strand space model. As is common when reasoning about security protocols, the language assumes asynchronous communication over a network where we assume messages to persist (for the benefit of any attacker). The language was initially designed for authentication protocols, so it is necessary to extend it slightly for the later study of digital signature exchange protocols by allowing nondeterministic choice and conditionals on input pattern matching.

We begin by assuming the following sets:

- the set **Entity** of *entities* or *participants*, ranged over by X,
- the set **Key** of *encryption keys*, ranged over by k. We assume that for every participant X there is a key $\mathit{Sig}(\mathsf{X})$ representing the signing key of X,
- the set **Hash** of *hash values*,
- the set **New** of *nonces*, representing long (secure pseudo-)randomly generated numbers, ranged over by n, and
- a set of *basic strings* ranged over by m, including for example the message to be signed and any control instructions to be sent to the third party.

We shall assume that the sets above are disjoint: though in principle a nonce value may be equal to a key, the probability of any encountered key being equal to a generated nonce is negligible.

Input in SPL will involve pattern matching: processes specify that they will accept a message matching a pattern, by which variables in the pattern are resolved to messages. We therefore define the set of *message patterns*, ranged over by M, which follow the grammar

$$M := m \mid \mathsf{X} \mid k \mid n \mid (M_1, M_2) \mid \{M\}_k \mid h(M) \mid \psi \mid x.$$

Above, $\{M\}_k$ represents the encryption of M using key k. The message $h(M)$ represents the application of a first -and second-preimage resistant cryptographic

hash function [9] to M. The symbols ψ and x range over *message variables*; we shall tend to use the former in patterns, explained next, where any message can match and the latter to suggest that we expect the message to be a nonce or hash value. We say that a message pattern is *closed*, or more simply is a *message*, if it contains no message variables.

We assume that we are given a function associating to every key k its inverse denoted by k^{-1}. Given the key k^{-1}, the message M can be recovered from $\{M\}_k$.

Processes are ranged over by p and b ranges over simple boolean expressions, with the standard definition of when a boolean holds where the atoms are equalities of (closed) messages, following the grammar

$$p := \mathsf{done(X)} \mid \mathsf{out\ new}\ x\, M.p \mid \mathsf{in\ pat}\ \boldsymbol{\psi}\ M\ \mathsf{where}\ b.p \mid \Big|\!\Big|_{i\in I}\, p_i \mid p + p \mid P$$

$$b := M = M' \mid b \wedge b \mid \neg b.$$

Above, $\boldsymbol{\psi}$ is a finite sequence of distinct message variables. In $\mathsf{out\ new}\ x\, M.p$, we view the variable x as bound, and in $\mathsf{in\ pat}\ \boldsymbol{\psi}\ M\ \mathsf{where}\ b.p$ we view the variables in $\boldsymbol{\psi}$ as bound. We require the free variables in M to be equal to the the variables in ψ, and write $M[\boldsymbol{N}/\boldsymbol{\psi}]$ for the substitution of each message N_i for free occurrences of ψ_i in M.

- $\mathsf{done(X)}$ is included for convenience in proofs, and indicates that the entity X has completed its role in the protocol.
- $\mathsf{out\ new}\ x\, M.p$ generates a new nonce n and then outputs the message $M[n/x]$ (using the newly generated nonce for x) before resuming as $p[n/x]$.
- $\mathsf{in\ pat}\ \boldsymbol{\psi}\ M\ \mathsf{where}\ b.p$ inputs, for any sequence of messages $\boldsymbol{N}$ the same length as $\boldsymbol{\psi}$, the message $M[\boldsymbol{N}/\boldsymbol{\psi}]$ providing the message is available on the network and the proposition $b[\boldsymbol{N}/\boldsymbol{\psi}]$ holds. The process then resumes as $p[\boldsymbol{N}/\boldsymbol{\psi}]$.
- $||_{i\in I}\, p_i$ is the parallel composition of processes indexed by the set I.
- $p_1 + p_2$ is the non-deterministic sum of p_1 and p_2: if p_1 can act to become the resumption p_1' then so can $p_1 + p_2$, and similarly for p_2.
- $P(\boldsymbol{M})$ is a process identifier predicated by a set of messages, and we assume a set of process definitions $Q(\boldsymbol{\psi}) = q$ allowing recursive definition of processes.

In the sequel, we adopt the notation $p_1 \parallel p_2$ for $||_{i\in\{1,2\}}\, p_i$ and write *nil* for the empty parallel composition. When the vector $\boldsymbol{\psi}$ is the set of free variables in M, we write $\mathsf{in}\ M\ \mathsf{where}\ b.p$ for $\mathsf{in\ pat}\ \boldsymbol{\psi}\ M\ \mathsf{where}\ b.p$, and when the condition b is a tautology we simply write $\mathsf{in}\ M.p$. When the variable x is not free in M or p, we write $\mathsf{out}\ M.p$ for $\mathsf{out\ new}\ x\, M.p$.

3 Optimistic signature exchange and the ASW protocol

A digital signature on a message by an entity indicates that the entity has seen and agreed to sign the message. For example, the message *"I agree to sell my house to Bob in exchange for £50"* may be signed by Alice by encrypting it

using her private key. Assuming that Alice has kept her private key safe and that everybody has access to her public key, Bob can prove to an arbiter than Alice has seen (and implicitly agreed to) this message.

Contracts require the *exchange* of messages, and this has to be done in a fair way: were Alice simply to send her signature directly to Bob, he could wait an indeterminate amount of time and then decide not to proceed. In the meantime, Alice would be left in limbo, unable to sell her house to anybody else.

Digital signature exchange protocols provide the means to fairly exchange signatures without giving either party an advantage. There are two classes of approach: *incremental* approaches in which the parties gradually release their signature to each other [10] and those based on a *trusted third party*.

With access to an entity that both parties trust to act in a prescribed way, the problem becomes much simpler: the two participants can simply send their signatures to the third party which only exchanges them once both are received. Though it can be shown that there is no non-incremental approach to signature exchange that does not involve a third party [11], the third party is potentially a bottleneck and represents a single point of failure; *optimistic* protocols, such as the Asokan-Shoup-Waidner (ASW) protocol, aim to do better, by only using the third party when necessary.

3.1 The ASW protocol in SPL

We now introduce the ASW protocol as given in [8], to which we refer the reader for a fuller account. The protocol begins after the two parties, say O acting as the originator and R acting as the responder, have agreed on the message m that they both wish to sign and that an entity T will act as a trusted third party. SPL terms representing both the originator, $orig(m, \mathsf{O}, \mathsf{R}, \mathsf{T})$, and responder, $resp(m, \mathsf{O}, \mathsf{R}, \mathsf{T})$, are presented in Figure 1.

The protocol begins with O sending to R the message M_1 defined in Fig. 1. The message is signed by O and acts as a promise to provide R with O's signature on m providing R follows the protocol. We specify now how the promise of O's signature can be fulfilled:

Definition 1. *A signature by* O *acting as originator for* R *on* m *is a pair of messages* n *and* $\{\mathsf{O}, \mathsf{R}, \mathsf{T}, m, h(n)\}_{Sig(\mathsf{O})}$.

In particular, the initial message M_1 can be translated to a signature by O when accompanied by the nonce x. Importantly, the pre-image resistance of the hash function h accompanied by the fact that x is a nonce (i.e. a long randomly-generated number) means that this translation of the promise to the signature causally depends on O revealing x.

If the protocol proceeds normally (the possibility of *aborting* or *resolving* is discussed below), after O has revealed its promise, the entity R then responds by sending O a message constituting a promise to provide O with R's signature. (Note that second preimage resistance of h prevents O from taking this to be the signature on some other message.)

Let $M_1 = \{\mathsf{O},\mathsf{R},\mathsf{T},m,h(x)\}_{Sig(\mathsf{O})}$ and $M_2 = \{\mathsf{O},\mathsf{R},\mathsf{T},m,z\}_{Sig(\mathsf{O})}$ in

$$
\begin{array}{ll}
orig(m,\mathsf{O},\mathsf{R},\mathsf{T}) \stackrel{\text{def}}{=} & resp(m,\mathsf{O},\mathsf{R},\mathsf{T}) \stackrel{\text{def}}{=} \\
\quad \mathsf{out\ new}\ x\ M_1. & \quad \mathsf{in}\ M_2. \\
\quad abort(M_1,\mathsf{O},\mathsf{T}) + & \quad \mathsf{out\ new}\ y\{h(M_2),h(y)\}_{Sig(\mathsf{R})}. \\
\qquad \mathsf{in}\ \{h(M_1),z\}_{Sig(\mathsf{R})}. & \qquad \mathsf{in}\ x\ \mathsf{where}\ h(x)=z. \\
\qquad \mathsf{out}\ x. & \qquad\quad \mathsf{out}\ y.\mathsf{done}(\mathsf{R}) \\
\qquad resolve(M_1,\{h(M_1),z\}_{Sig(\mathsf{R})},\mathsf{O},\mathsf{T},\mathsf{O}) + & + \quad resolve(M_2,\{h(M_2),h(y)\}_{Sig(\mathsf{R})},\mathsf{O},\mathsf{T},\mathsf{R}) \\
\qquad + \quad \mathsf{in}\ y\ \mathsf{where}\ h(y)=z.\mathsf{done}(\mathsf{O}) &
\end{array}
$$

$$
\begin{array}{ll}
abort(M,\mathsf{O},\mathsf{T}) \stackrel{\text{def}}{=} & resolve(M,M',\mathsf{O},\mathsf{T},\mathsf{X}) \stackrel{\text{def}}{=} \\
\quad \mathsf{out}\{\mathtt{abort},M\}_{Sig(\mathsf{O})}. & \mathsf{out}(\mathtt{resolve},M,M'). \\
\quad \mathsf{in}\ \{\mathtt{aborted},\{\mathtt{abort},M\}_{Sig(\mathsf{O})}\}_{Sig(\mathsf{T})}.\mathsf{done}(\mathsf{O}) & \quad \mathsf{in}\ \{\mathtt{aborted},\{\mathtt{abort},M\}_{Sig(\mathsf{O})}\}_{Sig(\mathsf{T})}. \\
\quad + \mathsf{in}\ \{M,\{h(M),z\}_{Sig(\mathsf{R})}\}_{Sig(\mathsf{T})}.\mathsf{done}(\mathsf{O}) & \qquad \mathsf{done}(\mathsf{X}) \\
 & + \ \mathsf{in}\ \{M,M'\}_{Sig(\mathsf{T})}.\mathsf{done}(\mathsf{X})
\end{array}
$$

Fig. 1. SPL terms for the originator and responder

Definition 2. *A signature by* R *acting as responder for* O *on* m *is a triple of messages* n *and* $M = \{\mathsf{O},\mathsf{R},\mathsf{T},m,n'\}_{Sig(\mathsf{O})}$ *and* $\{h(M),h(n)\}_{Sig(\mathsf{R})}$.

The promise by O is then fulfilled by sending R the nonce n chosen for x; this acts as proof that O either has received R's signature or can do so via the third party. Finally, R sends to O its nonce that allows conversion of its promise to an actual signature.

Let
$$
\begin{array}{rl}
M_2 = & \{\mathsf{O},\mathsf{R},\mathsf{T},m,z\}_{Sig(\mathsf{O})} \\
M_a = & \{\mathtt{abort},M_2\}_{Sig(\mathsf{O})} \\
M_r = & (\mathtt{resolve},M_2,\{h(M_2),w\}_{Sig(\mathsf{R})})
\end{array}
$$
in
$$
TTP(\mathsf{T}) = \Big|\Big|_{\mathsf{O},\mathsf{R}\in\mathbf{Entity}} \Big|\Big|_{m\in\mathbf{Message}} \Big|\Big|_{z\in\mathbf{Hash}} TTP_0(\mathsf{T},\mathsf{O},\mathsf{R},m,z)
$$
where
$$
\begin{array}{rcl}
TTP_0(\mathsf{T},\mathsf{O},\mathsf{R},m,z) & \stackrel{\text{def}}{=} & \mathsf{in}\ M_a.\quad \mathsf{out}\{\mathtt{aborted},M_a\}_{Sig(\mathsf{T})}.\quad TTP_a(\mathsf{T},\mathsf{O},\mathsf{R},m,z) \\
 & & +\ \mathsf{in}\ M_r.\quad \mathsf{out}\{M_r\}_{Sig(\mathsf{T})}.\quad TTP_r(\mathsf{T},\mathsf{O},\mathsf{R},m,z,M_r)
\end{array}
$$

$$
TTP_a(\mathsf{T},\mathsf{O},\mathsf{R},m,z) \stackrel{\text{def}}{=} \mathsf{in}\ M_r.\quad \mathsf{out}\{\mathtt{aborted},M_a\}_{Sig(\mathsf{T})}.\quad TTP_a(\mathsf{T},\mathsf{O},\mathsf{R},m,z)
$$

$$
\begin{array}{rcl}
TTP_r(\mathsf{T},\mathsf{O},\mathsf{R},m,z) & \stackrel{\text{def}}{=} & \mathsf{in}\ M_r.\quad \mathsf{out}\{M_r\}_{Sig(\mathsf{T})}.\quad TTP_r(\mathsf{T},\mathsf{O},\mathsf{R},m,z) \\
 & & +\ \mathsf{in}\ M_a.\quad \mathsf{out}\{M_r\}_{Sig(\mathsf{T})}.\quad TTP_r(\mathsf{T},\mathsf{O},\mathsf{R},m,z)
\end{array}
$$

Fig. 2. SPL terms for the TTP

The ASW protocol allows both the originator and responder to contact the trusted third party if their counterparty fails to respond in the expected way. This could, for example, be due to network failure or due to the other participant

(maliciously or not) failing to follow the protocol. The TTP handles two forms of request as part of the *abort* and *resolve* subprotocols. SPL terms for the TTP are presented in Figure 2.

The abort subprotocol is called by the originator if it fails to receive a promise from the responder after it has sent its promise to the responder. The resolve subprotocol can be called by either participant once it has received the promise from its counterparty.

The subprotocols' behaviour is subtle: it depends on whether there has been any prior request to abort or resolve. If the resolve subprotocol is invoked and the abort subprotocol has not earlier been invoked, the third party responds by providing a signature generated on behalf of the counterparty.

Definition 3. *A signature generated by* T *on behalf of either the originator* O *or responder* R *on message* m *is a message of the form*

$$\{\texttt{resolve}, M, \{h(M), n'\}_{Sig(\mathsf{R})}\}_{Sig(\mathsf{T})}$$

for some hash value n' *and* $M = \{\mathsf{O}, \mathsf{R}, \mathsf{T}, m, n\}_{Sig(\mathsf{O})}$ *for some hash value* n*.*

If, however, the abort subprotocol has been invoked prior to the resolve request, the third party simply informs the participant that neither party shall receive a signature from it and therefore that the participant can safely assume that its counterparty shall receive no form of signature; it does so by sending the message $\texttt{out}\{\texttt{aborted}, M_a\}_{Sig(\mathsf{T})}$, where M_a is the earlier message requesting that the exchange be aborted.

If the abort subprotocol is invoked by the originator and the resolve subprotocol for the message has not earlier been invoked, the TTP responds with a message informing the participant that the transaction has, as above, been aborted. If, otherwise, the other participant has earlier resolved the transaction, the TTP responds with a signature generated on its counterparty's behalf.

It is assumed that interaction with the TTP takes place over *resilient* communication channels: a communication channel is resilient if any message sent over it will eventually be received. An adversary can, however, observe messages sent over the channel and temporarily delay them. We note here that communication between the participants and the TTP is not encrypted; we have adhered to the original presentation since it is unnecessary for the correctness properties studied later. Encrypted communication could easily be added, and would address 'attacks' that are not normally considered where, for example, where the first two messages in a normal exchange are intercepted and then sent as a resolve request, meaning that the exchange certainly goes ahead.

4 Concurrent games

We now turn to giving a semantics to SPL, representing terms as concurrent strategies. Concurrent strategies are founded upon *event structures* [12].

Definition 4. *An event structure comprises three components* $(E, \leq, \mathrm{Con})$*, where* E *is the set of* events*,* $\leq \subseteq E \times E$ *is a partial order representing* causal dependency *and the non-empty set* $\mathrm{Con} \subseteq \mathcal{P}_{\mathrm{fin}}(E)$ *represents* consistency*, which jointly satisfy:*

1. $\{e' : e' \leq e\}$ *is finite for all* $e \in E$*,*
2. $\{e\} \in \mathrm{Con}$ *for all* $e \in E$*,*
3. *if* $X \in \mathrm{Con}$ *and* $Y \subseteq X$ *then* $Y \in \mathrm{Con}$*, and*
4. *if* $X \in \mathrm{Con}$ *and* $e \in X$ *and* $e' \leq e$ *then* $\{e'\} \cup X \in \mathrm{Con}$*.*

Events are viewed as atomic and can only occur once. A *configuration* of an event structure is a subset of events $x \subseteq E$ that is down-closed, meaning that $e \in x \;\&\; e' \leq e \implies e' \in x$, and consistent, meaning that, for all subsets $X \subseteq x$, if X is finite then $X \in \mathrm{Con}$. Write $\mathcal{C}^\infty(ES)$ for the configurations of an event structure ES, write $\mathcal{C}(ES)$ for its finite configurations, and write $x \overset{e}{\text{—}\!\subset} $ if $e \notin x$ and $x \in \mathcal{C}^\infty(ES)$ and $x \cup \{e\} \in \mathcal{C}^\infty(ES)$. An event structure is *elementary* if $\mathrm{Con} = \mathcal{P}_{\mathrm{fin}}(E)$.

We often make use of the binary relations of immediate causal dependency and conflict between events. Immediate causal dependency $e \rightarrowtriangle e'$ means that $e \neq e'$ and $e \leq e'$, and for any u such that $e \leq u \leq e'$ either $e = u$ or $e' = u$. Often, the consistency of a set of events can be determined just by examining pairs of events. In such a case, the consistency relation can be replaced by a *conflict* relation placing two events in conflict, written $e \# e'$, if there is no configuration containing them both. Finally, we say that an event e of an event structure ES is said to be *initial* if $e' \leq e$ implies $e' = e$.

Given a subset of events $V \subseteq E$ of an event structure $ES = (E, \leq, \mathrm{Con})$, the *projection* $(V, \leq_V, \mathrm{Con}_V)$ of ES to V has events V, causal dependency $e \leq_V e'$ iff $e \leq_{ES} e'$ and $X \in \mathrm{Con}_V$ iff $X \in \mathrm{Con}$ for any finite $X \subseteq V$.

Morphisms on event structures will be used to exhibit how the behaviour in one event structure can be simulated by another.

Definition 5. *A (partial)* morphism *from an event structure* $ES = (E, \leq, \mathrm{Con})$ *to an event structure* $ES' = (E', \leq', \mathrm{Con}')$ *is a partial function* $f : E \rightharpoonup E'$ *such that* $fx \in \mathcal{C}(ES')$ *for all configurations* $x \in \mathcal{C}(ES)$*, and*

$$\textit{if } e, e' \in x \textit{ and } f(e), f(e') \textit{ both defined then } f(e) \neq f(e').$$

The latter condition enforces the view that events are indivisible. A morphism is said to be *total* if f is total and *rigid* if it is total and preserves causal dependency in the sense that $f(e) \leq' f(e')$ whenever $e \leq e'$. A *rigid inclusion*, written $\hookrightarrow$, is a rigid map that is also an inclusion.

When defining event structures, it is necessary to ensure that every event has a unique causal history. This aspect can be lightened by building event structures out of *rigid families*.

Definition 6. *A rigid family* $\mathcal{F}$ *is a non-empty set of finite partial orders that are down-closed under rigid inclusions: if* $q' \in \mathcal{F}$ *and* $q \hookrightarrow q'$ *is a rigid inclusion, viewing* q *and* q' *as elementary event structures, then* $q \in \mathcal{F}$*.*

Concretely, down-closure of $\mathcal{F}$ stipulates that for any $q \in \mathcal{F}$, if e is maximal in q then $q \setminus e \in \mathcal{F}$, where $q \setminus e$ is the partial order with e removed.

Clearly, any event structure determines a rigid family where the orders are its finite configurations ordered by causal dependency; we call the elements of this family *ordered configurations* of the event structure. Conversely, an event structure can be obtained from a rigid family by taking its events to be primes i.e. the partial orders with a *unique* maximal element. Causal dependency between primes $p \leq p'$ is determined by whether there is a rigid inclusion $p \hookrightarrow p'$. A set of primes X is consistent in the event structure if it is finite and it has a supremum, with respect to the partial order of rigid inclusion, in the rigid family.

Example 1. Let the rigid family $\mathcal{F}$ be the down-closure (w.r.t. rigid inclusions) of the two partial orders on the left below. Their corresponding event structure is drawn on the right, where the arrows represent causal dependency and the wavy line represents conflict. The event c_1 is the prime partial order $a \to c$ and c_2 is the prime partial order $c \leftarrow b$.

We make use of a number of constructions on event structures. Given an event structure ES, we now adopt the convention of writing E for its set of events, $\leq$ for its conflict relation and Con for its consistency relation. When ES has a subscript, we add the subscript to its components.

Firstly, the simple parallel composition of event structures $\|_{i \in I} ES_i$ has events $\bigcup_{i \in I} \{i\} \times E_i$, causal dependency $(i, e) \leq (i', e')$ iff $i = i'$ and $e \leq_i e'$ and a set of events X is consistent iff $\{e : (i, e) \in X\} \in \mathrm{Con}_i$ for every $i \in I$. The event structure $!ES$ consists of countably-many copies of ES placed in parallel with each other, $!ES = \|_{i \in \mathbb{N}} ES$.

The *augmentation* of an event structure $\mathrm{aug}(ES)$ is defined via a rigid family. A partial order q is in the rigid family if its underlying set $|q|$ is a member of $\mathcal{C}(ES)$ and the order is an extension of the order on the configuration in ES: it satisfies $e \leq e' \implies e \leq_q e'$ for all $e, e' \in |q|$.

Note that there are total functions from the events of $\mathrm{aug}(ES)$ and $!ES$ to the events of ES taking any event in either $\mathrm{aug}(ES)$ or $!ES$ to the event that generated it.

4.1 Concurrent games and winning strategies

An *event structure with polarity* is an event structure with a total function $\mathrm{pol} : E \to \{+, -\}$ attaching a polarity, either $+$ for player or $-$ for opponent, to every event. The intuition is that the state of an interaction between the player and the opponent is a configuration of an event structure with polarity called the *game*. The player can extend the configuration by playing any of the enabled $+$ events and the opponent can extend the configuration by playing any of the enabled $-$ events. The occurrence of any event can affect what the other player

can do: it can enable events in the game that causally depend upon it and it prohibits the occurrence of events that are inconsistent with it.

The constructions on event structures above extend straightforwardly to event structures with polarity, with polarity being inherited from the event structures from which they are constructed. A further important operation is the dual $ES^{\perp}$, which is $(E, \leq, \mathrm{Con}, \mathrm{pol}')$ where $\mathrm{pol}'(e) = +$ iff $\mathrm{pol}(e) = -$.

We now introduce *strategies* on event structures with polarity as introduced in [5], which are potentially non-deterministic specifications of how the player is to act. Their definition is guided by properties that are desired when strategies are composed; we briefly mention an application of composition in the conclusion.

Definition 7. *A strategy for the player is a total morphism $\sigma : S \to A$ between event structures with polarity s.t.*

- *(Polarity preservation)* $\mathrm{pol}_S(e) = \mathrm{pol}_A(\sigma(e))$ *for all* $e \in E_S$,
- *(Receptivity) for any* $x \in \mathcal{C}^{\infty}(S)$, *if* $\sigma(s) \stackrel{e}{\multimap} \subset$ *and* $\mathrm{pol}(e) = -$ *then there exists unique* s *such that* $x \stackrel{s}{\multimap} \subset$ *and* $\sigma(s) = e$, *and*
- *(Innocence) if* $e \rightarrowtriangle e'$ *in* S *then either* $\sigma(e) \rightarrowtriangle \sigma(e')$ *in* A *or* $\mathrm{pol}(e) = -$ *and* $\mathrm{pol}(e') = +$

Correctness properties shall be expressed with reference to *winning strategies* [13].

Definition 8. *Consider a strategy $\sigma : S \to A$. With reference to a set $W \subseteq \mathcal{C}^{\infty}(A)$ of* winning configurations*, the strategy σ is* winning *if every $+$-maximal configuration of S (i.e. every configuration $x \in \mathcal{C}^{\infty}(S)$ for which there is no s satisfying $x \stackrel{s}{\multimap} \subset$ and $\mathrm{pol}(s) = +$) has $\sigma x \in W$.*

4.2 Semantics of SPL

The semantics of a closed SPL term shall be given as a strategy on a game denoted by *SPL*, in which players perform *actions* successively and potentially repeatedly. Let **Msg** denote the set of all (closed) messages and let the set of actions be

$$\begin{aligned} Act = &\{\mathsf{in}\ M : M \in \mathbf{Msg}\} \cup \{\mathsf{out}\ M : M \in \mathbf{Msg}\} \\ &\cup \{\mathsf{done}(X) : X \in \mathbf{Entity}\} \cup \{\mathsf{new}\ n : n \in \mathbf{New}\}. \end{aligned}$$

Given an event structure Y and $p \in \{+, -\}$, let Y^p denote the event structure with polarity where the underlying event structure is Y and all events have polarity p. Viewing Act as an event structure with trivial causal dependency and consistency relations, we define the game $SPL_0 = \mathrm{aug}(!Act^-) \parallel \mathrm{aug}(!Act^+)$. Let the function sending an event of SPL_0 to its image in Act be denoted by $act : SPL_0 \to Act$. The game *SPL* is specified as a rigid family: its partial order configurations are configurations x in $\mathcal{C}(SPL_0)$ inheriting order from SPL_0 for which, for any $n \in \mathbf{New}$, there is at most one $e \in x$ such that $act(e) = \mathsf{new}\ n$.

Actions of the process to which we give semantics are represented by events with positive polarity. Events with negative polarity represent any other process that may act in parallel with the process to which we give semantics. The definition of the game *SPL* allows the player to have strategies that successively perform actions in response to those of the opponent; in particular, the use of augmentation is necessary for the strategy to be innocent. The difference between the games SPL_0 and *SPL* is that, in *SPL*, a nonce can be generated only once.

We now proceed to give the semantics of SPL terms. We omit the formal definition of recursion, which is a straightforward adaption of the recursive definition of event structures [14]. By induction on the size (the number of actions) of terms, we shall define an event structure with polarity S_p and strategy $\sigma_p : S_p \to SPL$. In each case, the set of negative events in S_p shall be equal to the set of negative events in *SPL*. Any two negative events in S_p are causally dependent iff they are causally dependent in *SPL*. For any finite set of negative events X of S_p, we shall have $X \in \mathrm{Con}_p$ iff X is consistent in *SPL*.

A first useful operation is *prefixing* by a positive action. Given a strategy $\sigma_p : S_p \to SPL$ and $\alpha \in Act$, and assuming first that $\alpha \neq \mathsf{new}\ n$ for $n \in \mathbf{New}$, we define $\alpha.\sigma_p : \alpha.S_p \to SPL$ to have domain $\alpha.S_p$ formed with events the disjoint union of those from S_p and a new event, say a, with positive polarity. Causal dependency in $\alpha.S_p$ extends that of S_p with $a \leq e$ for all events e with positive polarity. A subset of events X of $\alpha.S_p$ is consistent iff $X \setminus \{a\}$ is consistent in S_p. Let a' be the first[1] initial positive α-event in *SPL* and let $SPL \uparrow a'$ be the projection of *SPL* to all negative events along with positive events that causally depend on but are not equal to a'. There is an isomorphism $\phi : SPL \cong SPL \uparrow a'$ that acts as the identity on events with negative polarity. The morphism $\alpha.\sigma_p$ sends a to a' and $e \neq a$ to $\phi(e)$.

Now, if $\alpha = \mathsf{new}\ n$, it is necessary to remove from $\alpha.S_p$ all positive events u for which there exists $e \leq u$ satisfying $e \neq a$ and $\mathrm{pol}(e) = +$ and $act(e) = \mathsf{new}\ n$. A subset of events X of $\alpha.S_p$ is consistent iff $X \setminus \{a\}$ is consistent in S_p and there is no $e \in X$ such that $\mathrm{pol}(e) = -$ and $act(e) = \mathsf{new}\ n$.

A second convenient operation is the *generalized sum* of strategies as described above (*i.e.* strategies with negative events those from *SPL*). Given a non-empty family of strategies $\sigma_i : S_i \to SPL$ for $i \in I$ where $S_i = (E_i, \leq_i, \mathrm{Con}_i, \mathrm{pol}_i)$, their sum $\sum_i \sigma_i : \sum_i S_i \to SPL$ is defined as follows. The event structure $\sum_i S_i = (E, \leq, \mathrm{Con}, \mathrm{pol})$ has negative events and causality dependency within them, as specified above, from *SPL*. The positive events form the set $\{(i, e) : i \in I \ \&\ e \in E_i\}$. No negative event depends on any positive event (as required for innocence), but for a positive event (i, e) we have $u \leq (i, e)$ iff either u is positive and there exists e' s.t. $u = (i, e')$ and $e' \leq_i e$ or u is negative and $u \leq_i e$. For a subset of events $X \subseteq E$, we have $X \in \mathrm{Con}$ iff there exists $i \in I$ and $Y \in \mathrm{Con}_i$ such that $X = \{e \in Y : \mathrm{pol}_i(e) = -\} \cup \{(i, e) : e \in Y \ \&\ \mathrm{pol}_i(e) = +\}$.

Input The strategy for $\mathsf{in\ pat}\ \boldsymbol{\psi}\ M\ \mathsf{where}\ b.p$ is as follows. Let A be the set of events e in *SPL* with negative polarity for which there exists M_0 such that

[1] this choice is arbitrary and can be alleviated by adding symmetry: see the conclusion.

$act(e) = \mathsf{out}\, M_0$ and there exists a sequence of messages $\boldsymbol{N}$ the same length as $\boldsymbol{\psi}$ satisfying $M_0 = M[\boldsymbol{N}/\boldsymbol{\psi}]$ and $b[\boldsymbol{N}/\boldsymbol{\psi}]$ holds. Note that any such sequence is unique for e. For any $e \in A$, by prefixing we form the strategy $\mathsf{in}\ M_0.\sigma_{p[\boldsymbol{N}/\boldsymbol{\psi}]}$ where $\sigma_{p[\boldsymbol{N}/\boldsymbol{\psi}]}$ is the strategy inductively obtained for $p[\boldsymbol{N}/\boldsymbol{\psi}]$. From this strategy, we obtain a strategy σ_e by adding a single immediate causal dependency $e \rightarrowtriangle a$, where a is the event for the initial input, the least positive event in $\mathsf{in}\ M_0.S_{p[\boldsymbol{N}/\boldsymbol{\psi}]}$. Finally, we define $\sigma = \sum_{e \in A} \sigma_e$.

Output Considering $\mathsf{out\,new}\ x\, M.p$, let n be any nonce. By induction, we have a strategy $\sigma_{p[n/x]}$ and so, by prefixing twice, a strategy $\sigma_{\mathsf{new}\ n\ .\ \mathsf{out}\ M[n/x].p[n/x]}$. The strategy for $\mathsf{out\,new}\ x\, M.p$ is defined to be $\sum_{n \in \mathbf{New}} \sigma_n$.

Parallel composition The strategy for the parallel composition $\|_{i \in I}\, p_i$ is relatively simple: we do not at this stage introduce any causal dependencies between the processes since we do now know with which other processes $\|_{i \in I}\, p_i$ shall be composed. Formally, we define the strategy $\sigma : S \to \mathit{SPL}$ where $S = (E, \leq, \mathrm{Con})$ as follows. For $i \in \{I\}$, let $\sigma_i : S_i \to \mathit{SPL}$ be the strategy strategy for p_i where $S_i = (E_i, \leq_i, \mathrm{Con}_i, \mathrm{pol}_i)$, and let $\mathit{SPL} = (E_{SPL}, \leq_{SPL}, \mathrm{Con}_{SPL}, \mathrm{pol}_{SPL})$.

$$E = \{e \in E_{SPL} : \mathrm{pol}_{SPL}(e) = -\} \cup \textstyle\bigcup_{i \in I}(\{i\} \times \{e : e \in E_i \ \&\ \mathrm{pol}_i(e) = +\})$$

$$\begin{aligned} e \leq e' \iff & \mathrm{pol}(e) = - \ \&\ \mathrm{pol}(e') = - \ \&\ e \leq_{SPL} e' \\ \text{or} \ & \mathrm{pol}(e) = - \ \&\ \mathrm{pol}(e') = + \ \&\ \exists i, u' : e' = (i, u') \ \&\ e \leq_i u' \\ \text{or} \ & \mathrm{pol}(e) = + \ \&\ \mathrm{pol}(e') = + \ \&\ \exists i, u, u' : e = (i, u) \ \&\ e' = (i, u) \ \&\ u \leq_i u' \end{aligned}$$

$$\begin{aligned} X \in \mathrm{Con} \iff & \{e \in X : \mathrm{pol}(e) = -\} \in \mathrm{Con}_{SPL} \ \text{ and for all } i \in I : \\ & \{u : (\mathrm{pol}(u) = - \ \&\ u \in X) \text{ or } ((i, u) \in X \ \&\ \mathrm{pol}(i, u) = +)\} \in \mathrm{Con}_i \end{aligned}$$

Nondeterministic sum The sum $p_1 + p_2$ has strategy $\sum_{i \in \{1,2\}} \sigma_i$, where σ_i is the strategy for p_i.

Theorem 1. *σ_p, as defined above, is a strategy for any closed term p.*

There is an asymmetry in the semantics above that warrants explanation: Any player input event in the strategy causally depends on an opponent event that outputs the same message, but there are no causal dependencies of the opponent events on player events (such dependencies would, indeed, violate the innocence condition). Causal dependency of opponent events on those of the process are central to reasoning about security protocols: for example, an initially secret message may be replayed but not guessed by an attacker, so any attacker event that outputs the secret message causally depends on some output by the process. As we proceed to describe, these causal dependencies are introduced after we have inductively obtained the strategy σ_p on *SPL*; the reason why it is not defined as we give the semantics for each term is that we do not know with which other processes the processes under consideration shall be composed and therefore what messages it will be possible for the attacker to output.

4.3 Constraining the attacker

We now move from from strategies that can be composed to form semantics to ones that are used to reason about the behaviour of adversaries.

The key principle is that any message that the attacker outputs has to be justified, in the sense of the Dolev-Yao [15] model, from messages to which it has access. For a set of messages s, we write $s \vdash M$ if M is justified by s, defined to be the least relation satisfying the following rules:

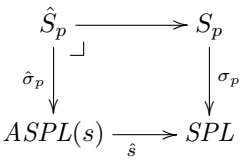
$s \vdash M$ if $M \in s$ $\quad$ $s \vdash (M_1, M_2)$ if $s \vdash M_1$ and $s \vdash M_2$
$s \vdash M_1$ and $s \vdash M_2$ if $s \vdash (M_1, M_2)$ $\quad$ $s \vdash M$ if $s \vdash k^{-}1$ and $s \vdash \{M\}_k$
$s \vdash \{M\}_k$ if $s \vdash M$ and $s \vdash k$ $\quad$ $s \vdash h(M)$ if $s \vdash M$

We write $M \prec M'$ if M is a sub-message of M', the least reflexive transitive relation such that $M_1 \prec (M_1, M_2)$ and $M_1 \prec \{M_1\}_k$. Given a set of messages s that is initially on the network, we now refine the game SPL so that all opponent outputs are justified and all opponent inputs of messages not in s depend on on corresponding outputs. We also ensure that any generated nonce is not a submessage in s. The game, denoted by $ASPL(s)$, is defined to be the event structure obtained from the following rigid family of configurations (inheriting polarity from SPL):

Definition 9. *For any configuration x of an event structure with polarity, subset of messages $s \subseteq$ **Msg** and $\alpha : x \to Act$, the configuration x is defined to be* secured *w.r.t. α iff for any e such that $\mathrm{pol}(e) = -$ and there exists M such that $\alpha(e) = \mathsf{out}\, M$, we have $s \cup \{N : \exists e' \leq e \;\&\; \mathrm{pol}(e') = - \;\&\; act(e') = \mathsf{in}\, N\} \vdash M$. A* source map *for x and α is a partial function $\kappa : x \rightharpoonup x$ that is defined on an event e iff $\mathrm{pol}(e) = -$ and there exists $M \notin s$ such that $e = \mathsf{in}\, M$, in which case we require that $act(f(e)) = \mathsf{out}\, M$.*

*The rigid family defining $ASPL(s)$ consists of finite partial orders $\leq$ over configurations $x \in \mathcal{C}(SPL)$ that contain no event e such that $act(e) = \mathsf{new}\, n$ for any $n \in$ **New** if $n \prec M \in s$, that are secured w.r.t. $act : x \to Act$ and for which there exists a source map κ such that $\leq$ is the transitive closure of $(\leq_{SPL} \cap (x \times x)) \cup \{(\kappa(e), e) : e \in x \;\&\; \kappa(e)$ defined$\}$.*

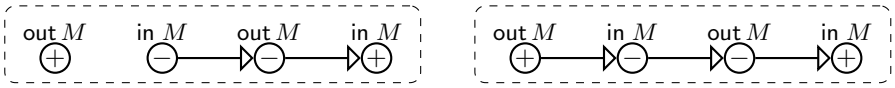
There is a morphism of event structures $\hat{s} : ASPL(s) \to SPL$. Given a strategy $\sigma_p : S_p \to SPL$ representing the semantics of p, we can form a strategy $\hat{\sigma}_p : \hat{S}_p \to ASPL(s)$ by pullback as drawn to the right, since the pullback of a strategy against a morphism that preserves polarity is also a strategy [6]. Concretely, the strategy $\hat{\sigma}_p$ only has opponent events that are justified by what the opponent may have intercepted according to the Dolev-Yao model. The pullback can be viewed as minimally modifying the domain S_p so that it meets the additional causal constraints in $ASPL(s)$. From now on, we denote by $(\!|p|\!)_s$ the strategy $\hat{\sigma}_p : \hat{S}_p \to ASPL(s)$. A particular instance of the pullback is obtained as follows.



Theorem 2. *A finite partial order $\leq_x \subseteq x \times x$ is an order configuration of $\hat{S}_p$ iff there is no $e \in x$ such that $act(e) = \mathsf{new}\ n$ and $n \prec M \in s$, there exists an order configuration $\leq_y \subseteq y \times y$ of S_p such that y is secured w.r.t. $act \circ \sigma_p : y \to Act$ and there exists a source map $\kappa : y \rightharpoonup y$ such that $\leq_x$ is the transitive closure of the following relation: $\leq_y \cup \{(\kappa(e), e) : e \in y \ \&\ \kappa(e)\ \mathit{defined}\}$.*

Note that the configurations are required to be partial orders: if the source map creates a cyclic dependency, it will not result in an order configuration.

The pullback construction has the effect of introducing causal dependencies of negative events on positive ones. For example, an order configuration of S_p of the form on the left below (where we label events with their actions) gives rise to the configuration in $\hat{S}_p$ on the right.



This kind of pattern is always encountered when one player event outputs a message that the attacker does not initially know and another player event receives it. It has the correct causal dependency of the player input on the player output. The intermediate opponent events are useful: they mean that we give the attacker the ability to intercept or delay messages.

5 Correctness properties

We now use the semantics to formulate correctness properties for the ASW protocol. The properties that we consider are expressed in terms of strategies being winning with respect to particular sets of winning configurations.

It is in the formulation of correctness properties that we capture the notion of resilient communication between the third party and the participants: the protocol will win by default if the resilience assumption is not satisfied. Since we do not explicitly represent channels of communication, we directly characterize the messages intended to be between participants and the third party T as those either under the key $Sig(\mathsf{T})$ or containing either `abort` or `resolve`.

The following predicates on configurations x of $ASPL(s)$ will be useful.

- $x \models \mathsf{Sig}^p_{\mathsf{Orig}}(\mathsf{X}, \mathsf{Y}, \mathsf{T}, m)$ if $\{M : \exists e \in x : act(x) = \mathsf{in}\ M\ \&\ \mathrm{pol}(e) = p\}$ contains a signature by X acting as originator for Y on m with TTP T
- $x \models \mathsf{Sig}^p_{\mathsf{Resp}}(\mathsf{X}, \mathsf{Y}, \mathsf{T}, m)$ if $\{M : \exists e \in x : act(x) = \mathsf{in}\ M\ \&\ \mathrm{pol}(e) = p\}$ contains a signature by Y acting as responder for X on m with TTP T
- $x \models \mathsf{Sig}^p_{\mathsf{TTP}}(\mathsf{X}, \mathsf{Y}, \mathsf{T}, m)$ if $\{M : \exists e \in x : act(x) = \mathsf{in}\ M\ \&\ \mathrm{pol}(e) = p\}$ contains a signature by TTP T on behalf of X or Y
- $x \models \mathsf{Resil}(\mathsf{T})$ if for all events $e \in x$ such that $\mathrm{pol}(e) = +$ and $act(e) = \mathsf{out}\ M$ and M is a message to/from the third party T, there exists $e' \in x$ such that $\mathrm{pol}(e') = -$ and $act(e') = \mathsf{out}\ M$

The first property is *effectiveness*: that whenever the protocol completes without invoking the third party, each party has the signature of the other.

Theorem 3 (Effectiveness). *Let s be any set of messages such that $s \not\vdash Sig(\mathsf{O})$ and $s \not\vdash Sig(\mathsf{R})$ and $s \not\vdash \{\mathsf{O}, \mathsf{R}, \mathsf{T}, m, M\}_{Sig(\mathsf{O})}$ for any message M. The strategy $(\!|orig(m, \mathsf{O}, \mathsf{R}, \mathsf{T}) \parallel resp(m, \mathsf{O}, \mathsf{R}, \mathsf{T}) \parallel TTP(\mathsf{T})|\!)_s$ is winning with respect to winning configurations x of $ASPL(s)$ that satisfy either:*

- *$x \models \mathsf{Sig}^+_{\mathsf{Orig}}(\mathsf{O}, \mathsf{R}, \mathsf{T}, m)$ and $\mathsf{Sig}^+_{\mathsf{Resp}}(\mathsf{O}, \mathsf{R}, \mathsf{T}, m)$,*
- *there exists $e \in x$ and M such that $\mathrm{pol}(e) = +$ and $act(e) = \mathsf{out}\, M$ but there is no $e' \in x$ such that $\mathrm{pol}(e') = -$ and $act(e') = \mathsf{out}\, M$, or*
- *there exists $e \in x$ such that $\mathrm{pol}(e) = +$ that carries a message containing either* `abort` *or* `resolve`*.*

Note that, since we trust how the third party will behave, it is included in the process being considered. The proof of the theorem is omitted, but it makes use of causal reasoning: the key aspect is that no attacker can interfere with the protocol since any such action would causally depend on some process event that outputs the signing key for either O or T, of which there is none.

The statement of effectiveness makes use of the notion of winning strategy to ensure that the environment (assumed to be hostile) does not indefinitely block transmission of a message. In particular, a configuration is winning if the process outputs a message but there is no corresponding negative event outputting the same message. The same kind of definition is used later to ensure that the attacker always releases messages to/from the third party: the other correctness properties do not require *all* messages eventually to get through.

Two further correctness properties for the ASW protocol are *fairness* and *timeliness*. Fairness asserts that if one party gains a signature of the other then the other party will gain a signature of the first if it continues with the protocol. Timeliness asserts that each party can proceed with the protocol no matter how the other party acts: it will not be stuck indefinitely waiting for the other. It is convenient to prove fairness and timeliness together. The properties for the originator and responder are stated separately since they hold even when the other party is dishonest (*i.e.* doesn't follow the protocol), and we only present them for the originator; those for the responder are similar.

Theorem 4 (Fairness and timeliness for the originator). *Let s be any set of messages such that $s \not\vdash Sig(\mathsf{O})$ and $s \not\vdash Sig(\mathsf{T})$ and $s \not\vdash \{\mathsf{O}, \mathsf{R}, \mathsf{T}, m, M\}_{Sig(\mathsf{O})}$ for any M. The strategy $(\!|orig(m, \mathsf{O}, \mathsf{R}, \mathsf{T}) \parallel TTP(\mathsf{T})|\!)_s$ is winning with respect to the set of winning configurations x of $ASPL(s)$ that satisfy either $x \not\models \mathsf{Resil}(\mathsf{T})$ or both*

1. *if $x \models \mathsf{Sig}^-_{\mathsf{Orig}}(\mathsf{O}, \mathsf{R}, \mathsf{T}, m) \vee \mathsf{Sig}^-_{\mathsf{TTP}}(\mathsf{O}, \mathsf{R}, \mathsf{T}, m)$ then $x \models \mathsf{Sig}^+_{\mathsf{Resp}}(\mathsf{O}, \mathsf{R}, \mathsf{T}, m) \vee \mathsf{Sig}^+_{\mathsf{TTP}}(\mathsf{O}, \mathsf{R}, \mathsf{T}, m)$, and*
2. *x contains a player event $\mathsf{done}(\mathsf{O})$.*

Fairness is the first point above and timeliness the second. Again, causal reasoning comes to the fore in their proofs. The key properties for fairness are that the key $Sig(\mathsf{O})$ is never available to the adversary, as before, and that if the adversary gains a signature either by O or on behalf of O then the signature must have come from either the O or the third party, and these events causally depend on other events that allow the generation of R's signature.

6 Conclusion and related work

We have given a semantics for SPL using concurrent games and used this to apply causal reasoning when studying correctness of the ASW protocol, providing what we feel are more intuitive and direct proofs than those obtained by inductive techniques based on interleaving structures. The work provides a starting point for application of concurrent games and their extensions to provide a rich foundation for the semantics of security protocols, where both interaction and causality are explicitly represented.

Formal methods have been applied previously to analyse the ASW and GJM protocols in [16, 4, 17]. [16] describes the use of the Murφ model checker, and implicitly assumes the fairness of runs (in the sense that it is assumed that the entities terminate) to study protocol fairness and other correctness properties. [4] introduced the idea of specifying fairness through the use of strategies and used the MOCHA model checker to study them. Notably, their games are played over interleaving structures, and fairness constraints on runs have to be added to deal, for example, with resilience. Finally, [17] studies the GJM (Garay-Jakobbsson-MacKenzie) protocol using a combination of inductive methods and interleaved tree structures to represent the game.

There has not been space to present the details here, but the semantics can also be applied to give an account of the GJM protocol [18], which uses a cryptographic primitive called private contract signatures to provide a property called *abuse freeness*: that there is no reachable configuration where one entity can prove to an external entity that it has to ability to choose unilaterally either to exchange signatures or to abort the protocol. In previous work [16, 17], this has been simplified to considering *balance*, which is that there is no reachable configuration where one entity can choose between exchange of signatures or aborting. Balance can be formulated in the framework presented here by saying that there is no configuration where there are two counterstrategies [13] to either the originator or responder in parallel with the third party, where one counterstrategy wins if it obtains the signature and the other counter strategy wins if it or the TTP do not reveal its signature.

There are a number of variations on the basic game structures that we intend to study. Firstly, rather than directly representing input moves in the game, we can model them using 'neutral' events in a 'partial strategy' [19]. Composition of partial strategies (such as those for the originator and responder) then has neutral events representing a global session type [20]. Composition may also play a role when probabilistic strategies [6] are studied, using probability to discuss the likelihood of either breaking a key or being able to do no better than simply guessing what value is encrypted beneath a key. Finally, the current work motivates the study of multiplayer games to allow us to reason about TTP accountability and to study multiparty exchange in the GJM protocol.

References

1. Thayer, F.J., Herzog, J.C., Guttman, J.D.: Strand spaces: Proving security protocols correct. Journal of Computer Security **7**(1) (1999) 191–230
2. Paulson, L.C.: Proving properties of security protocols by induction. In: Proc. 10th IEEE workshop on Computer Security Foundations. CSFW (1997)
3. Crazzolara, F., Winskel, G.: Events in security protocols. In: ACM Conference on Computer and Communications Security
4. Kremer, S., Raskin, J.F.: Game analysis of abuse-free contract signing. In: Proc. 15th IEEE workshop on Computer Security Foundations. CSFW (2002)
5. Rideau, S., Winskel, G.: Concurrent strategies. In: Proc. LICS, IEEE Computer Society (2011) 409–418
6. Winskel, G.: Distributed probabilistic and quantum strategies. In: Proc. MFPS. Volume 298 of ENTCS. (2013)
7. Castellan, S., Clairambault, P., Winskel, G.: Symmetry in concurrent games. In: Proc. LICS. (2014)
8. Asokan, N., Shoup, V., Waidner, M.: Asynchronous protocols for optimistic fair exchange. In: Proc. 1998 IEEE Symp. on Security and Privacy. (May 1998) 86–99
9. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press (2001)
10. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM **28**(6) (June 1985)
11. Pagnia, H., Gärtner, F.C.: On the impossibility of fair exchange without a trusted third party. Technical report, Darmstadt University of Technology (1999)
12. Winskel, G.: Event structures. In: Advances in Petri Nets. Volume 255 of LNCS., Springer (1986) 325–392
13. Clairambault, P., Gutierrez, J., Winskel, G.: The winning ways of concurrent games. In: Proc. LICS, IEEE (2012)
14. Winskel, G.: Event structure semantics for CCS and related languages. In: Proc. ICALP. Volume 140 of LNCS., Springer (1982) 561–576
15. Dolev, D., Yao, A.C.C.: On the security of public key protocols. IEEE Transactions on Information Theory **29**(2) (1983) 198–207
16. Shmatikov, V., Mitchell, J.C.: Finite-state analysis of two contract signing protocols. TCS **283**(2) (2002)
17. Chadha, R., Kanovich, M., Scedrov, A.: Inductive methods and contract-signing protocols. In: ACM Conference on Computer and Communications Security. (2001)
18. Garay, J.A., Jakobsson, M., MacKenzie, P.: Abuse-free optimistic contract signing. In: Advances in Cryptology 99. Volume 1666 of LNCS. (1999)
19. Castellan, S., Hayman, J., Winskel, G.: Strategies as concurrent processes. Available from http://www.cl.cam.ac.uk/∼gw104/
20. Takeuchi, K., Honda, K., Kubo, M.: An interaction-based language and its typing system. In: Proc. PARLE '94. Volume 817 of LNCS. (1994)