On instance separation in the UC-framework

István Vajda

May 30, 2012

Abstract
The UC approach of Canetti offers the advantage of stand-alone analysis while keeping security guaranties for arbitrary complex environment. When we implement by this approach first we have to ensure secure instance separation and based on this condition, we are allowed to carry out a stand-alone analysis. In this report we propose three issues related to instance separation in UC-context:

We consider the problem of universal composability in cases, when we cannot assume independence of instances. Next we formalize the interleaving attack and a related security notion. In time-aware protocols time-based separation of instances is one of the standard implementation techniques. We propose an event-driven clock model towards purely symbolic analysis of time-aware protocols.

1. Introduction

The instance based approach of Canetti [5] emphasizes the importance of instance "independence". The security analysis of a protocol is simplified considerably, if we are allowed to carry out the analysis on a stand-alone model, where we analyze under the assumption that there are no other concurrently running instances from the same or other protocols. One of the main advantages of the UC approach is that it keeps the simplicity of the stand-alone analysis even in complex protocol environment.

In the alternative approach of Pfitzmann et al. [2] (BPW-approach) the requirement of instance separation is present in an implicit way. All instances of a protocol "live together" within the trusted host (*TH*) machine: the trusted host is a reactive ideal functionality storing all actions during the lifetime of the protocol. The essence of their approach is a reactive, composable Dolev-Yao type cryptolibrary, using which the protocol is abstracted into a completely symbolic version. Different invocations of ideal cryptographic primitives are separated by ideal access guarantees to the corresponding non-public stuff by so called handles. That is this separation approach works with finer granularity (at the level of crypto building blocks instead of complete protocol instances) and at the same time it provides inter- and inner-instance separation. For application examples of the BPW-approach see [9-13]. In this report we refer to the approach of Canetti [5] in Section 3, and to the BPW-approach in Section 4.

An ideal functionality F at Canetti is defined such a way that it excludes the possibility of interactions between different instances for honest or adversarial actions alike. It follows that a (UC-)secure implementation ρ of ideal functionality F must

Technical University of Budapest E-mail: vajda@hit.bme.hu

ensure similar protection. UC theorem [5] guarantees that, if an F- hybrid protocol π invokes (fresh) instances $F_1,...,F_m$, then the composed protocol π^ρ, where each F – instance is substituted by a (fresh) instance of ρ, securely implements protocol π (m is an arbitrary integer). Recall, universal composability means that UC-security is kept under composition with any protocol.

From the viewpoint of instance separation, the main part of the analysis is to show for an implementation ρ that an adversary is not able to carry out harmful interactions between concurrent instances.

There can also be the case that some kind of interactions are tolerated by the ideal functionality from cost or efficiency reasons, like the cases of "allowed leakage" or "allowed delay" in favor of the adversary. One expects that in this case the stand-alone approach of analysis will not work in general. In Section 3.1 we will examine the composition theorem for such tolerated dependences between instances.

Interleaving attacks are the usual attack types against those protocols, which cannot provide secure instance separation. We introduce the chosen instance attack (CIA) and a notion of security under CIA. This notion formalizes and generalizes interleaving attacks.

A standard technique for instance separation relies on time information. We will examine the related problem of time modeling and propose an abstract time model.

Matsuo [7] proposes a UC model for timestamps based on TTP real time clock both in the real and the ideal model, where in fact, UC approach refers to the security of request/reply communication with TTP clock. Buldas et. al. [4] proposes a UC-secure time-stamping scheme. Their time model is a quantized real time source: their Stamper works in rounds with time unit (hour, day, week, etc.). Backes [3] analyzes the Kerberos protocol in the BPW's UC model. Here the ideal model is purely symbolic and the analysis is carried out on a purely symbolic version of the protocol into which the Kerberos protocol is abstracted. They replaced the timestamps by nonces assumed known to the participants generating the timestamps, because timestamps are not modeled in the BPW's model. Note, this way they could model one characteristic property of time: it is changing. However, the time has a very important characteristic lost here: it is increasing continuously and as a consequence, the time order of any two events can be obtained. The proved (authentication) properties in [3] became weaker than the authentication Kerberos really offers, because by this simplification they could not grasp the purpose of timestamps in Kerberos.

In Section 4 we make an attempt to propose a way towards a purely symbolic time model by substituting the real-time clock with an event-driven clock.

2. Contributions

In this report we propose three issues related to instance separation in UC-context.

Section 3.1: We consider the task of composability in case of dependent instances, first of all to emphasize that UC and independence of instances are not "synonyms" of each other. We show that universal composability is possible also for dependent

instances, where realization ρ of ideal functionality F is provably secure for all tolerable interactions (dependences) between instances defined by the ideal functionality. However, the advantage of stand-alone analysis is lost, in general. (Proposition 1**)**

Section 3.2: We generalize and formalize the interleaving attack by introducing the definition of chosen instance attack (CIA) and a relation-based security notion under CIA. We examine its relationship to standard indistinguishability. (Proposition 2, Lemma 1, Lemma 2)

Section 4: We propose a step towards purely symbolic timing approach for the analysis of time-aware protocols. In particular, we propose an event-driven clock (e-time) and discuss a few properties of e-time relevant to an analysis in this model. (Property 1, Property 2)

3. Dependence of instances

First we show a simple example for the task of instance separation and the interleaving attack. (It is for illustration, can be skipped at first reading.)

Example 1: Consider the following two-party key exchange protocol:

1. $A \rightarrow B : \{K\}K_b$
2. $B \rightarrow A : \{N\}K$
3. $A \rightarrow B : \{sig_A(N)\}K$

Party *A* generates a fresh key *K* and encrypts it with the public key of party *B*. Party *B* chooses a fresh value (nonce) *N* and encrypts it with the new secret key *K*. Party *A* signs the nonce and encrypts the signature with secret key *K*. The wished goal of the protocol is that after the instance finishes the two parties and only those will be aware of the new secret key *K*. At first glance the protocol seems secure but it is not. Adversary *X* is able to implement an interleaving attack:

$A \rightarrow X : \{K\}K_x$
$\quad\quad \mathbf{X \rightarrow B : \{K\}K_b}$
$\quad\quad \mathbf{B \rightarrow X : \{N\}K}$
$X \rightarrow A : \{N\}K$
$A \rightarrow X : \{sig_A(N)\}K$
$\quad\quad \mathbf{X \rightarrow B : \{sig_A(N)\}K}$

The adversary takes part in two concurrent instances (for instance, by corrupting a party in both) and channels messages between them. At the end of the runs three parties *A*, *B* and *X* become aware of the same key *K*. The instances are not separated in this protocol. A security patch is the following:
1. $A \rightarrow B : \{K\}K_b$
2. $B \rightarrow A : \{N\}K$
3. $A \rightarrow B : \{sig_A(B, K, N)\}K$
□

3.1. UC for dependent instances

Instance separation is a technique of modular design and analysis of protocols. Indeed, if we can separate the instances in a secure way, then in the next step of the security assessment of the implementation, we can carry out a stand-alone analysis.

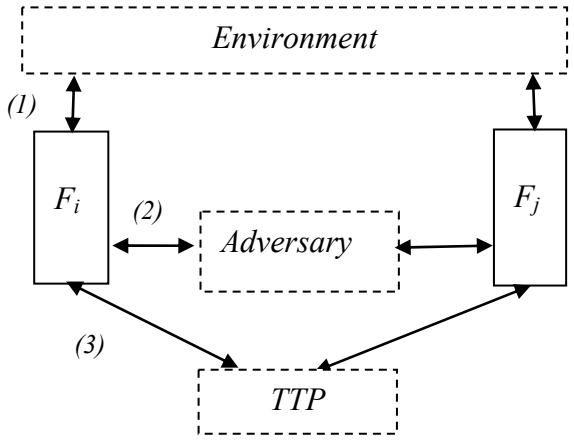

Fig.1.: Sources of potential interactions between instances

The ITI-model (Interactive Turing machine Instance [5]) of protocol instances has three communication ports (in Fig.1. instances F_i and F_j of an ideal functionality F):

- input/output ports to the environment ((1)),
- communication ports between instances via the adversary ((2)),
- input/output ports to subroutines ((3)).

Any interaction between different instances can be realized only via these ports. The UC-theorem of Canetti [5] is valid only if parties of subroutines (sub-parties) do not accept/send input/output from parties/sub-parties of any other instance:

Recall, the UC theorem [5] ensures that, if an F- hybrid protocol π invokes (fresh) instances $F_1,...,F_m$, then the composed protocol π^ρ , where each F –instance is substituted by a (fresh) instance of ρ, securely implements protocol π (m is an arbitrary integer). "Freshness" means not only that all random elements of an instance are chosen independently and uniformly but also that the instances must have disjoint local states.

A typical case of such kind of dependence between instances is, when instances rely on a single common instance corresponding to a TTP service (Fig.1). The JUC (Joint-state Universal Composition) theorem [6] provides a technique for getting rid of such dependence. Essentially: we produce virtual sub-instances of the single common instance by introducing sub-SIDs.

The adversary may be active in several different instances and may try to channel information (whole messages or parts of them) between different instances (via communication port (2) in Fig.1). Recall, a corresponding type of attack is the interleaving attack (Example 1, above).

This is the standard type of attempted dependence, which is to be foiled by techniques of instance separation: the definition of ideal functionality F forbids such an interaction and in case of a secure realization any such attack attempt leads to the abortion of the target instance. Essentially: in the ideal functionality, protocol messages carry a unique instance identifier (SID), which cannot be tampered with by the adversary and any attempts to channel message from any other instance is detected by at least one of honest parties based on the instance SID.

Note, these scenarios do not cover all sources of potential dependence between instances. It may be the case that we allow "legal" interactions between instances via communication port (2). Tolerable interaction is similar to "allowed leakage" or "allowed delay" in favor of the adversary and it is defined also by the ideal functionality. Below, we consider universal composability in case of such interactions between instances.

We mention that dependence between instances may happen also from further reasons: for example, via dependence between inputs of different instances. For instance, a corresponding scenario is the following: there exists a statistical dependence between the time samples of a process, where the samples correspond to the input values of instances (port (1) in Fig.1). The adversary as an a priori knowledge may be aware of this dependence. Leakage of input information in one instance can be a useful predicate for a corresponding quantity in a concurrent instance.

Here we would like to emphasize that universal composability, in principle, need not assume independence of instances, informally, universal composability and secure separation of instances are not "synonyms" of each other. However, in general, we loose the simplicity of stand-alone analysis of an implementation ρ of ideal functionality F:

Proposition 1: Universal composability is possible also for dependent instances, where realization ρ of ideal functionality F is provably secure for all tolerable interactions between instances defined by the ideal functionality. However, the advantage of stand-alone analysis is lost, in general.

Proof: (sketch)
We can apply the technique of Canetti [5]. Below we emphasize only those issues, which are particular to the statement.

Assume that protocol ρ UC-realizes ideal functionality F, where no distinguishing environment with dummy adversary and black box simulation can distinguish a single instance of ρ from a single instance F. Recall, this type of distinguishing environment is very powerful: it incorporates the usual adversary, it has access not only to the standard input/output interfaces of the parties but it has full access also to the communication between parties of instances.

Recall, the environment models arbitrary protocol environment under computational constraint. In particular, it models also an environment of an arbitrary set of concurrent instances of ρ, where the environment is able to carry out interactions between those instances and the instance target of the distinguishing effort.

This way, when assessing whether a realization ρ of an ideal functionality F with tolerable interaction is UC-secure or not, the distinguishing environment checks all security consequences of tolerable interactions.

In the hybrids argument (within an indirect proof, like in [5]), distinguishability of hybrid protocol π running with t-tuple of instances of ideal functionality F from composed protocol π^{ρ} with t-tuple of instances of protocol ρ is reduced to the distinguishability of single instances of ρ and F. The distinguishing "l-th hybrid – environment" for the single instance is just one possible inter-instance constellation, under which protocol ρ is UC-secure by the above argument and as such, the ρ-instance is indistinguishable from an instance of ideal functionality F.
□

A related research is to explore those security tasks where potential interactions between concurrent instances appear as tolerable imperfections, and as such, are part of the definition of corresponding ideal functionalities.

3.2. A standard definition for interaction-proof property of protocols

Informally, secure separation of instances means that different instances cannot have "observable" impact on each other's "performance". Here we give a definition for instance separation by defining the "observable impact on performance". This way we give a definition also for protocols secure against interleaving attacks. Referring to the previous section, here we assume that ideally we do not allow interaction between instances, i.e. our aim here is to propose a standard definition for interaction-proof implementation.

We introduce the notion of chosen instance attack (CIA) and the definition of security under CIA. A few notations follow:

A target instance is attacked by the adversary. The target instance is chosen with input according to a distribution D over the input space. Let M denote the set of protocol messages of an instance with output $O(M)$. Let S denote the output space of a non-attacked instance. Consider an attack against the target instance, which is done by modifying message set M. If the modification is detected by the (honest) parties of the target instance, we say it is aborted. The probability of an event E under non-abort condition will be denoted by $\widehat{P}(E) = P(E \mid non-abort)$.

Let S' ($S' \supset S$) denote the output after running the target instance under attack in non-abort cases, i.e. when there is no attack or the attack remains undetected by the honest parties of the instance. Here we do not describe the representation of the output explicitly; we assume that it can be done by some appropriate way: e.g. by giving what the participants (the adversary is included) of all involved instances know relative to their a priori knowledge after the run of the instance. For an example see Example 1.

Let $R : SxS' \to \{0,1\}$ denote an efficiently computable relation, where $R(a,b) = 1$ means, that $a \in S$ and $b \in S'$ are in relation R. For brevity, below we use notation $R(a,b)$ for equality $R(a,b) = 1$.

Definition 1. (*chosen instance attack, CIA*)

The goal of adversary X^{CIA} is the modification of the protocol messages of the target instance, such a way, that the resulted output is in relation *R* with the output of the non-attacked target instance. Adversary X^{CIA} has access to a CIA-oracle, where the CIA-oracle is the following:

The adversary is allowed to request the invocation of at most *r* instances, where the set of parties and their inputs are given in the request. □

Example 2: Consider Example 1:

$M = \{\{K\}K_X, \{N\}K, \{sig_A(N)\}K\}$,

$O(M) = \{$after the run the two parties *A* and *X* of the instance and no one else are aware of key $K\} \in S$,

$O(X^{CIA}(M)) = \{$the two parties *A* and *X* of the target instance and party *B* from a concurrent instance are aware of the same key $K\} \in S' \setminus S$.

□

The strength of an adversary rapidly grows with the number of corrupted parties. This is especially true for interleaving attacks, where the adversary builds bridges between two or more instances via corrupted parties. For simplicity of parametrization, let $c \geq 0$ denote the level of corruption in an instance of the protocol, i.e. in each interacting instance during the attack.

Here we consider an arbitrary efficiently computable protocol. It may happen that some input information is carried to the output by the protocol as publicly accessible plaintext, where, in addition, the integrity of this information is not protected. Such protocol could be attacked easily by an X^{CIA} adversary by manipulating the plaintext characters. Excluding such protocols does not seem to weaken our definitional approach.

Now we define the security under chosen instance attack, which is the advantage of an adversary carrying out modification attack against the target instance with the help of the CIA-oracle relative to the success of a simulator having access only to public a priori information:

Definition 2. (*security under CIA*)

A protocol is (ε, t, r, c)-secure under chosen instance attack, if for arbitrary probability distribution *D* over the input space, for arbitrary efficiently computable relation *R* and for any adversarial algorithm X^{CIA} with complexity limit *t*, request limit *r* and corruption level *c*, there exists a simulator *X'* with complexity limit *t* such that

$$\hat{P}\left\{R\left(O(M), O\left(X^{CIA}(M)\right)\right)\right\} - \hat{P}\left\{R\left(O(M), O\left(X'(pub)\right)\right)\right\} \leq \varepsilon \qquad (1)$$

where *pub* denotes all publicly available a priori information and where the probability is calculated over all random variables (randomness used to set up the instances in (1) and the internal random elements of algorithms *X* and *X'*). □

One might expect that probability $\hat{P}\left(R\left(O(M),O\left(X'(pub)\right)\right)\right)$ is negligibly small. However, this probability depends on the considered protocol: assume a two-party protocol which is only a public key encryption of the input message sent by one of the parties and decoded by the other party. Now, if the set of input messages contains only two elements, then the considered probability is exactly ½, for the only meaningful relation, the dissimilarity.

Definition 2 formally resembles the notion of non-malleability (NM) for public key encryption, therefore, for brevity, we refer to it as (ε,t,r,c) NM-security of a protocol against adversary X^{CIA}.

Corollary of Definition 2: (ε,t,r,c) NM-security of a protocol implies its (ε,t,c) "stand alone" security.

Proof:
When an X^{CIA} adversary does not send requests to the CIA oracle, it simplifies to a "stand-alone" adversary.
□

Definition 2 is illustrated in Fig. 2: if we choose an instance *M* according to input distribution *D* and choose an X^{CIA} adversary with complexity parameters (t,r,c), then instance $X^{CIA}(M)$ may fall into set abort or into set non-abort. In case of non-abort, *M* and $X^{CIA}(M)$ can be distinguishable or non-distinguishable by a standard distinguisher (see Definition 3 below).

Note, abort/non-abort decision is carried out by the set of honest participants by running the rules of the protocol. A standard distinguisher is much more powerful: it is allowed to run arbitrary distinguishing algorithm under complexity limit *t*.

Recall, for public key encryption the implication NM-CPA → IND-CPA stands: if the encryption leaks, the obtained information on the plaintext can be used to generate a ciphertext with a relation based on this information. Here we guess – formally - similar implication for "typical" protocols.

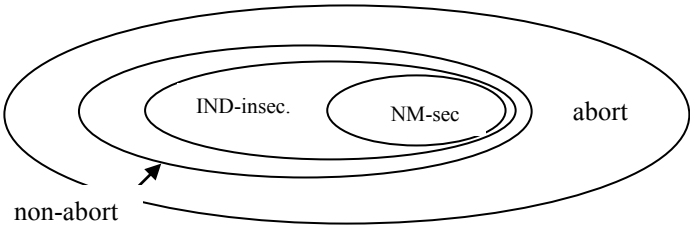

Fig.2. Guessed relationship of standard indistinguishability and relation-based security notions for "typical" protocols

Definition 3. (*IND-insecure*)
We call a protocol (ε,t,r,c) IND-insecure, if there exists a probability distribution *D* over the input space and an adversarial algorithm X^{CIA} with complexity parameters (t,r,c), such that for all distinguishing algorithms *Y* with complexity limit *t*:

$$\widehat{P}_{b\leftarrow_r\{0,1\}}\left\{Y\left(M_b\right)=b\right\}\le \tfrac{1}{2}+\varepsilon/2\,, \qquad (2)$$

for instance pair $\left(M_0, M_1 = X^{CIA}(M_0)\right)$, $M_0 \neq M_1$, where the input of M_0 is according to distribution *D*.
□

For emphasizing the decision aspect, in formula (2) only the random selection of *b* is designated; the probability is calculated over the following random variables: coin flipping variable *b*, randomness used to set up instance M_0, internal random elements of algorithms X^{CIA} and *Y*.

Now, we state a partial result under the following assumptions:
A1.) Consider protocols such that $\widehat{P}\left(R\left(O(M),O\left(X'(pub)\right)\right)\right)$ is negligible for arbitrary efficiently computable relation *R* and arbitrary simulator *X'* with complexity limit *t*.
A2.) Adversary X^{CIA} fabricates instances with output in set $S'\setminus S$ if not aborted.

Both assumptions seem plausible for "typical" protocols and adversarial goal

Proposition 2: Under assumptions A1 and A2, if a protocol is (ε,t,r,c) IND-secure, then it is $(\sim\varepsilon,t,r,c)$ NM-insecure. (Here $\sim\mu$ means a value within a negligible distance to μ.)

Proof: A protocol is IND-secure if for any probability distribution D over the input space, for any adversarial algorithm X^{CIA} with complexity parameters (t,r,c), there exists a distinguishing algorithm *Y* with complexity limit *t*, such that

$$\widehat{P}_{b\leftarrow_r\{0,1\}}\left\{Y\left(M_b\right)=b\right\}>\tfrac{1}{2}+\varepsilon/2\,. \qquad (3)$$

We show that there exists an efficiently computable relation *R* such that adversary X^{CIA} $(\sim\varepsilon,t,r,c)$-breaks NM-security.

Relation *R* is the following: $R:Sx(S'\setminus S)\to 1$, $R:Sx(\overline{S'\setminus S})\to 0$. Relation *R* is efficiently computable: by assumption A2 algorithm *Y* is able to decide if its input instance M_b produces an output in set $S'\setminus S$ or in set *S* with success probability (3). Here follows $\widehat{P}(O(M_1)\in S'\setminus S)>\varepsilon$ and $\widehat{P}\left(R(O(M_0),O(M_1))=1\right)>\varepsilon$. Taking into account assumption A1 and Definition 2, we arrive to the claim.
□

The following technical lemma gives hint on our expectation that set IND is strictly larger than set NM.

Lemma 1: Two random variables ξ and η over space U may be perfectly indistinguishable while perfectly related by an efficiently computable relation R.

Proof: Let U={00,01,10,11} and random variables ξ and η have uniform distribution over U. If
$P(\xi=(01)\,|\,\eta=(00))=P(\xi=(10)\,|\,\eta=(00))=1/2$,
$P(\xi=(00)\,|\,\eta=(01))=P(\xi=(11)\,|\,\eta=(01))=1/2$,
$P(\xi=(00)\,|\,\eta=(10))=P(\xi=(11)\,|\,\eta=(10))=1/2$,
$P(\xi=(01)\,|\,\eta=(11))=P(\xi=(10)\,|\,\eta=(11))=1/2$,
then random variables ξ and η are perfectly indistinguishable and at the same time perfectly related by opposite parities.
□

Note, if we consider arbitrary efficient protocol, then we can get any output random variable produced by an efficient algorithm. Note, furthermore, relations are able to explore dependencies given in two-dimensional distributions, which standard distinguishers cannot on marginal distributions.

Our intuitive feeling is that non-negligible standard distinguishability implies a non-negligible and efficiently computable relation between the considered pair of random variables. The next technical lemma shows a corresponding result for the case of independent variables.

Lemma 2: Assume two random variables ξ and η over the space U are computationally (ε,t)-distinguishable. If these variables are independent, then there exists an efficiently computable relation R, such that $P\big(R(\xi,\eta)=1\big)>\varepsilon^2$.

Proof: Let Z denote an algorithm, which (ε,t)-distinguishes variables ξ and η, i.e.

$$P\big(Z(\xi)=0\big)-P\big(Z(\eta)=0\big)>\varepsilon$$

The statistical distance between probability distributions $D_\xi(y)$ and $D_\xi(y)$, $y\in U$ of random variables ξ and η is also at least ε. Let decompose space U to $U=U_0\cup U_1$, $U_0=\{y: D_\xi(y)\ge D_\eta(y)\}$, $U_1=U\setminus U_0$. It follows, that $P\big(\xi\in U_0\big)-P\big(\eta\in U_0\big)>\varepsilon$ and $P\big(\eta\in U_1\big)-P\big(\xi\in U_1\big)>\varepsilon$. Hence $P\big(\xi\in U_0\big)>\varepsilon$ and $P\big(\eta\in U_1\big)>\varepsilon$. Let R be defined the following way: $R(a,b)=1$, if $\{a\in U_0\}\cap\{b\in U_1\}$ and zero otherwise. Applying the assumed independence, we arrive at $P\big(R(\xi,\eta)=1\big)>\varepsilon^2$. Relation R is efficiently computable: let $R(a,b)=1$, if $\{Z(a)=0\}\cap\{Z(b)=1\}$ and zero otherwise.
□

A standard definition of security, in general, is a notion of breaking under different attack classes. Definition 2 of NM-CIA is also of this sort. In simulation-based approach (like in UC analysis) the essential point is what we consider the ideal notion of breaking. It is an important problem, in general, to find the relationship between these two definitional approaches of secure implementation. It is a research problem also for NM-CIA.

Instead of formula (1) we could define the interaction-proof property also by the following way:

Definition 2':
The protocol is (ε,t,r,c)-secure under chosen instance attack, if for arbitrary distribution D over the input space, for arbitrary pair of efficiently computable relation R and for any adversarial algorithm X^{CIA} with complexity limit t, request limit r and corruption level c, there exists an adversary Y with complexity parameters (t,c) such that

$$\hat{P}\left\{R\left(O(M),O\left(X^{CIA}(M)\right)\right)\right\}-\hat{P}\left\{R\left(O(M),O\left(Y(M)\right)\right)\right\}\leq\varepsilon\,. \qquad (1')$$

□

Note, adversary Y is an adversary against the stand-alone instance, therefore, by this definition we can focus better on the gain provided by the CIA-oracle.

4. An abstract time model: event-driven Clock

The goal of this section is to propose an abstract time model an event-driven Clock. Time is an abstraction. Heuristically, it is nothing else than a continuously growing index of irreversible changes which happen in the environment surrounding us. Indeed, if suddenly all these changes would be reversed we might feel it as a journey through the time into the past. The experience about the irreversibility of changes leads to the main property of time: it steps only forward. This means that the time is a handle for our thinking to arrange the events in order.

The following plausible hypothesis is a justification of our approach:
If we are not allowed to use a real time clock, then the only possible substitute is counting the number of appropriately chosen events within the instances of the protocol under examination.

Related theoretical questions are the following: How can a time-aware protocol be analyzed in a purely symbolic system? How the answer depends on the adversarial model?
Here we attempt to make a step towards the abstraction of the time source (the event-driven Clock).

We assume the standard asynchronous communication model ([5]), which is characterized by the following properties: within an instance only one party is communicating at any given time; reception of a message activates the sending of the

next message; messages are transmitted with the mediation of the adversary. This is a natural communication model for many cryptographic protocols as well as it models strong capabilities of the adversary in controlling communication between honest users. In such model the time of sending and receiving messages by honest users are the only relevant time-related events.

Note, the Clock model is a tool used only in the analysis. The output of such an analysis could be considered as a "proof in event-driven Clock model" (coined after "proof in random oracle model").

4.1. The Clock

First, we have to define the underlying set of events. The most natural selection for events, are the actions of communication between parties, i.e. the sending and the reception of protocol messages. These are those time moments to which timing actions are usually set in time-aware protocols. At each occurrence of such an event, the Clock makes one step forward. The Clock has two input ports (*Step*, *Time request*) and one output port (*Time reply*) (Fig.3.)

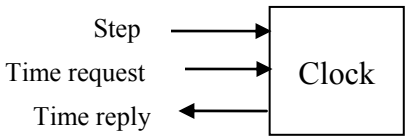

Fig.3. I/O ports of the Clock

The Clock is a common resource for the instances of the protocol. The Clock starts running with the first message sending in the first instance. Within an instance, the ideal functionality controls the Clock by having exclusive access to Step, Time request and Time reply ports of the Clock.

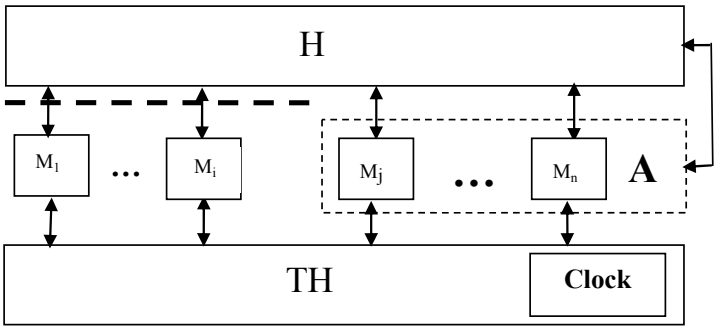

Fig.4. Clock is a part of the trusted host (ideal system)

The work, which provided samples for symbolic analysis of cryptographic primitives in UC-framework is the BPW-approach detailed in [1]. Below we refer to a few notations and technical elements from this approach.

In Fig.4 we see that parties have access to the Clock via the Trusted Host (*TH*). Machine *TH* models the protocol environment. In particular, it schedules the invocation of concurrent instances. In Fig.4 one instance is illustrated, where protocol machines $M_1,...,M_n$ are the parties of the instance and where parties $M_j,...,M_n$ are compromised.
In database *D* of the trusted host, *TH* an entry has the following attributes (see [1]):

$$(ind, type, \arg, hnd_1, ..., hnd_n, hnd_A, len)$$

which are the following: the index of the entry, the data type (data, list, nonce, enc,… etc.), arguments (e.g. a pair of "lists" representing indexes pointing to the plaintext and public key behind an encryption (type enc)), the handlers identifying who knows this entry and the length of the entry.

This database stores the history corresponding to the run of the protocol, i.e. all past and concurrently running instances of the protocol. Note, the index here is "correlated" with the time by the event driven Clock: it increases by one with each new entry. What is the point here: we can naturally include the event-driven Clock into this *TH* model. It is an index-like value as it is also an incrementally growing natural number. It could be a time data type, which is set by *TH* at corresponding requests of parties.

When machine *TH* receives an input from a party as well as when *TH* sends an output to a party the Clock makes one step forward. The order of processing a protocol message sent from Party *A* to Party *B* is shown in Fig.5.

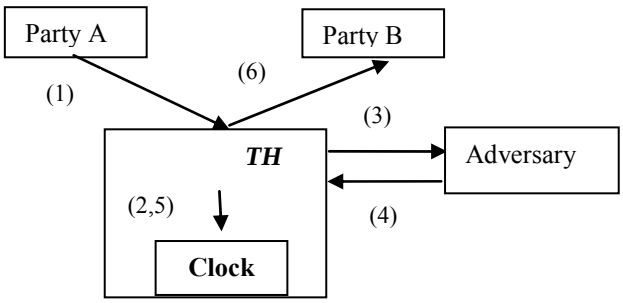

Fig.5. The order of processing a message and stepping the Clock
(*A* → *TH*→ Clock → Adversary → *TH* → Clock → *B*)

We add a publicly available event-driven Clock also to the real system (Fig.6.).

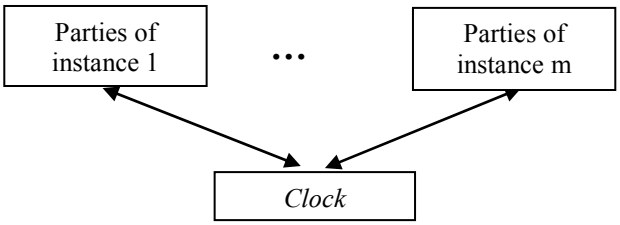

Fig.6. Clock functionality as a common resource in the real system

The order of processing a message during message transmission between party *A* and *B* is the following:

A → Clock → Adversary → Clock → *B*

We assume the synchronism of Clocks can be kept in a natural way between the event-driven Clock in the real and the ideal system.

4.1. Properties of the event driven Clock

Recall, an event-driven Clock model is thought to be used only in the analysis. The output of such an analysis could be considered as a "proof in event-driven Clock model". What we can say about the reliability of an analysis carried out in the Clock model? In particular: What we can say about the set of potential attacks against the real protocol which are not captured in the Clock model?
A thorough answer needs further research. Here we summarize a few preliminary thoughts and straightforward properties.

When, in general, we speak about a secure emulation of a specification, we consider all the possible adversarial algorithms under complexity limitations. Among all algorithms we may find also algorithms which manipulate the speed of Clock. Note, it is done unintentionally, because the Clock model is used only for the purpose of analysis.

For brevity, we use notation *e-time* for event-driven time and *r-time* for real-time.

The natural properties of r-time are the following:

- it steps only forward (it is mapped to an increasing integer number);
- r-time is consistent with the "earlier/later" property: a larger r-time value corresponds to "later";
- the difference between two r-time values correspond to their time distance

A usual assumption that the source of r-time cannot be manipulated, which means, that we do not consider physical attacks against the real-time clock. Note, here we are talking about the time source and not about the communication with it.

The manipulations by which an adversary could have effect on the e-time are the following: *deletion*, *insertion*, *delaying* of protocol messages. We do not assume a DoS attacker, therefore we exclude deletion attack. By insertion we mean generation of extra "traffic", e.g. by invoking extra instances of the protocol.

Now we consider which r-time properties are retained by e-time. The answer will depend on the assumed model of the adversary, i.e. in contrary to real-time source it is sensitive to certain adversarial behavior.

In particular: How e-time retains the (r-)time order and the relative (r-)time duration under attacks?

First we consider the strongest adversary with respect to time manipulation, which is the standard assumption in the standard asynchronous communication model ([5]): the adversary sees all communication between the parties of the protocol and she is allowed to delay the protocol messages by her wish. The other e-time sensitive attack is when the adversary generates extra communication, i.e. by invoking concurrent instances. For reference, we call such an adversary, the *strongest adversary*.

Property 1: E-time retains the r-time order under the attack of the strongest adversary. (earlier/later consistency)

Proof:

Assume messages B and C arrive to TH by r–time t_1 and t_2, respectively, where $t_1 < t_2$.
By delaying message B, the adversary is able to exchange the r-time order of delivery of messages B and C. Note, the e-time order of delivery is changing, accordingly.
When the adversary generates extra message traffic, she is able to "accelerate" e-time. Assume the adversary generates extra traffic between messages B and C. Note, such a manipulation neither affect the r-time or e-time order of the messages B and C.
□

E-time distance between consecutive events is fixed to 1, therefore it cannot reflect the magnitude of r-time distance between them:

Property 2: E-time is not consistent with r-time with respect to time durations under the attack of the strongest adversary: to different r-time intervals with the same length, e-time intervals with different length may correspond.

Proof:

Consider messages B and C from the previous proof.
By delaying messages beyond r–time t_2, the adversary can shrink the e-time distance between messages B and C (at delivery).
If the adversary generates extra message traffic between r–time t_1 and t_2, she can widen the e-time distance between message B and C, while their r-time distance remains t_2-t_1.
□

In sum, in case of the strongest adversary our best hope from e–time is to retain earlier/later consistency, however we cannot attain consistency with respect to (r-)time duration.

Recall, when we talk about adversarial attack against e-time, it means that we imagine an analysis in the event-driven Clock model and we consider all possible adversarial algorithms (under complexity constraint), among them also those which are attacks against e-time. A more fair approach to e-time is when we restrict the adversary to a smaller set of attacks which are not directed against e-time.

For example, if we do not allow the adversary to generate extra traffic in order to distort e-time, the adversary is able only to shrink e-time intervals by her delaying capabilities (cf. the proof of Property 2). If such time interval corresponds to a

message-acceptance time-window by an honest party, then the expected goal of an adversary is to widen and not to shrink time gates between interacting instances.

Note, furthermore, if an attack exists only in the Clock model, then a security claim in this model is a conservative statement.

With Section 4 our intention was to propose a potential direction towards a completely symbolic analysis of time-aware protocols, and consider (straightforward) limitations. Further research is needed to provide a thorough insight to how reliable an analysis can be carried out in the Clock model. Adversarial manipulations of e-time which open a gate to "artificial" attacks against the protocol amplify the power of the adversary and may lead to needlessly strong security requirements against a protocol or in other words to a security claim, which is a conservative statement. On the other side, we do not expect relevant attacks efficient against r-time model and inefficient (negligible) against e-time model.

References

[1] M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library. *IACR Cryptology ePrint Archive*, Report 2003/015, http://eprint.iacr.org/, January 2003.

[2] M. Backes and B. Pfitzmann. A General Composition Theorem for Secure Reactive Systems. *Theory of Cryptograpy Conference (TCC 2004)*, LNCS 2951, pp. 336-354, 2004.

[3] M. Backes , I. Cervesato , A. D. Jaggard , A. Scedrov and J. K. Tsay. Cryptographically Sound Security Proofs for Basic And Public-Key Kerberos. *Proc. 11th European Symp. on Research. in Comp. Sec, 2006.*

[4] A. Buldas , P. Laud , M. Saarepera and J. Willemson. Universally Composable Time-Stamping Schemes with Audit. In ISC05, LNCS 3650. *Cryptology ePrint Archive: Report 2005/198*

[5] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols". *Cryptology ePrint Archive: Report 2000/067*. (received 22 Dec 2000, revised 13 Dec 2005)

[6] R.Canetti and T.Rabin. Universal Composition with Joint State. *Crypto'03*, 2003.

[7] T.Matsuo and S.Matsuo. On Universal Composable Security of Time-Stamping Protocols. *Cryptology ePrint Archive: Report 2005/148*

[8] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. 7th ACM CCS*, pages 245–254, 2000

[9] I.Vajda. Cryptographically Sound Security Proof for On-Demand Source Routing Protocol EndairA. *Cryptology ePrint Archive Report 2011/103*. http://eprint.iacr.org/2011/103.pdf

[10] I.Vajda. Framework for Security Proofs for Reactive Routing Protocols in Multi-Hop Wireless Networks. *Cryptology ePrint Archive Report 2011/237*. http://eprint.iacr.org/2011/237.pdf

[11] I.Vajda. New look at impossibility result on Dolev-Yao models with hashes. *Cryptology ePrint Archive Report 2011/335*. http://eprint.iacr.org/2011/335.pdf

[12] I.Vajda. Non-malleable public key encryption in BRSIM/UC. *Cryptology ePrint Archive Report 2011/470*. http://eprint.iacr.org/2011/470.pdf

[13] I.Vajda. UC framework for anonymous communication. *Cryptology ePrint Archive Report 2011/682*. http://eprint.iacr.org/2011/682.pdf