Continuous Certification of Non-Repudiation in Cloud Storage Services

Maria Krotsiani, George Spanoudakis
Department of Computer Science,
City University London, UK
{Maria.Krotsiani.1, G.E.Spanoudakis}@city.ac.uk

***Abstract*— This paper presents a certification model for Non-repudiation (NR) of cloud storage services. NR, i.e., the possession of proofs that certain exchanges have taken place amongst interacting parties, is a significant security property for cloud data storage services but is less studied than other security properties (e.g., integrity, confidentiality). Our model for certifying NR is based on a continuous monitoring approach, i.e., a 3rd level certification in the reference certification framework of the Cloud Security Alliance.**

Keywords— non-repudiation; cloud; security certification; continuous monitoring;

I. INTRODUCTION

Despite the fast growth of cloud services, security is still a main barrier for their adoption. Cloud computing is aimed at providing users with efficient and flexible services. At the basic cloud infrastructure layer, these services include compute and data storage services [4][5]. Both these types of services must be secure [4][5][11][13]. Nevertheless, storing data in clouds is still a concern from a security point of view, as several incidents cast doubt on the level of security offered by cloud storage services. Examples of such incidents include the corruption of Amazon Simple Storage Service (S3) leading to stored files no longer matching customers' hashes [6] and an access-control bug in Google Docs that allowed unauthorized access to documents [14]. Although security properties as confidentiality, authentication, access control, and availability have been studied thoroughly for clouds, non-repudiation has only been recently investigated in this context [9].

Non-repudiation (NR) is a property of data storage, requiring that when a data owner (consumer) sends a request to a cloud provider for uploading (downloading) data, the data uploading (downloading) transaction should be conducted in a way such that neither the data owner (consumer) nor the cloud data storage provider could deny having participated in a part or the whole of this transaction. Several protocols have been proposed to realise non-repudiation (e.g., [4][7][11][12][15][16]). The basic principle that underpins these protocols is that along with a data uploading (downloading) request the data owner (consumer) sends a "Non-Repudiation of Origin" (NRO) token, i.e., a proof of sending the request, and expects to receive evidence of "Non-Repudiation of Receipt" (NRR) from the cloud provider, acknowledging that the specific request was received.

Whilst these protocols have been proven to provide NR under given assumptions their implementation can have bugs or suffer from attacks, such as man-in-the-middle, replay or timeline attacks [4]. Therefore, certifying the correct implementation of protocols and the robustness of their implementation to these types of attack is necessary for giving cloud customers the assurances required for NR.

In this paper we present an approach for certifying the implementation of a NR protocol mechanism that is based on the fair multi-party non-repudiation (MPNR) scheme proposed in [4]. Our certification scheme is based on a continuous monitoring approach that we introduced in [10], as part of the CUMULUS project [3]. More specifically, it is based on monitoring the actual operations of cloud services to gather evidence enabling a continuous assessment of the satisfaction of the security property of interest. Under this approach, a certificate for the security property is issued when the accumulated evidence is sufficient (e.g., it covers a required spectrum of service usage scenarios) and there is no violation of the security property within the monitoring period. Hence, our approach is compatible with certification of level 3 maturity in the reference certification maturity model of the Cloud Security Alliance [18].

The use of a monitoring based certification approach is necessary for assessing NR in cloud storage services, as the provision of this property depends on the correct realisation of an NR protocol not only by the cloud provider but also by the data producers/owners and consumers interacting with it. However, as the latter two partners are not under the control of the cloud provider and may change dynamically, they introduce uncertainties that require a dynamic form of NR assessment and certification.

The rest of the paper is structured as follows. Sect. II overviews related work. Sect. III presents the NR Protocol for cloud services assumed by the certification model. Sect. IV gives an overview of the specification of certification models. Sect. V presents the certification model for NR. Sect. VI gives an overview of the CUMULUS framework that is used to implement the NR certification model. Finally, Sect. VII provides concluding remarks and directions for future work.

II. RELATED WORK

There are two strands of research related to this paper: (a) research on NR protocols and (b) research on certification.

While non-repudiation can be achieved by standard cryptographic mechanisms, one of the key issues in NR protocols is that of fair message transfer between the involved parties, i.e., ensuring that the communication parties follow the rules of the protocol and do not abandon execution

intentionally. An approach addressing fairness by using an *inline* trusted third party (TTP), i.e., a TTP participating in every transmission of the protocol is presented in [2]. However, the constant involvement of TTP can lead to bottlenecks and reduced availability. Hence, other protocols use *online* TTPs, i.e., where TTPs that do not participate in all transmissions [15]. A further improvement is the use of *offline* (aka *optimistic*) TTPs [1], which are TTPs involved only in cases of disputes or network failure. This approach has been also adopted in [8][16]. Other work focuses on securing cloud storage. Popa et al. [13] presented an NR protocol for cloud storage and Feng et al. [5] introduced four variants of NR protocols, based on digital signature and authentication code.

Research on certification has traditionally produced methods based on the Common Criteria model [17]. These models use Evaluation Assurance Levels (EALs) to reflect any added assurance requirements in order to achieve a CC Certification. More recently the Cloud Security Alliance (CSA) has proposed the Open Certification Framework (OCF) [18]. OCF provides different maturity levels of certification. The 3rd maturity level of OCF refers to certification based on continuous monitoring, which is compatible with the approach we advocate in this paper.

III. NON- REPUDIATION PROTOCOL FOR CLOUD SERVICES

Our certification model aims to certify adherence to the enhanced NR Protocol for clouds presented in [4]. To enable the reader understand this model, in this section, we give an overview of the underpinning NR protocol.

This protocol involves four parties: (i) a data owner/provider ("A"), (ii) a data user ("B"), (iii) the Cloud Provider ("C") and (iv) a Trusted Third Party (TTP). These parties interact through three phases of the protocol, which are shown in Figure 1. These phases are: (1) the data *upload phase*, (2) the data *download phase*, and (3) a *recovery phase*.

Before describing these phases, we provide some basic definitions necessary for understanding them:

- *NRO*: Evidence of Non-Repudiation of Origin, sent by a sender to a receiver. The receiver will hold this evidence as proof if the sender denies having sent the message.
- *NRR*: evidence of Non-Repudiation of Receipt, sent by the receiver to the sender. The sender will hold this evidence as a proof if the receiver denies having received the message.
- f_M: Flag indicating the intended purpose of a message M.
- *l*: Unique label chosen by A to link all messages.
- *M*: Message sent from a sender to a receiver.
- *H(M)*: Hash function applied to message M.
- *K*: Message key defined by the sender.
- B_L: Group of data users B_i who are authorised to download message M and are capable of decrypting it.
- Seq_i: Unique sequence number of each message.
- $EG_B()$: Group encryption scheme known only to B_L group.
- $E_X(Y)$: Asymmetric encryption of message Y produced by party X's public key.
- $S_X(Y)$: signature of message Y produced by X's private key.

The three phases of the protocol are described below.

A. Upload Phase

In this phase the data owner A sends a request to the cloud provider C, for uploading data. Firstly, A encrypts a message M (i.e., the data) with a key K and generates two different NROs: NRO_{AB} and NRO_{AC}. NRO_{AB} will be used by data users B to get the key K required to decrypt M and $S_A(H(M))$ to verify the data integrity after downloading M from C. A encrypts NRO_{AB} using the group encryption scheme $EG_B()$ to guarantee that only the intended recipients of the B_L can have access and decipher NRO_{AB} and M. NRO_{AC} is the proof of evidence that A sent the request to C and is encrypted with C's public key. This step is defined as:

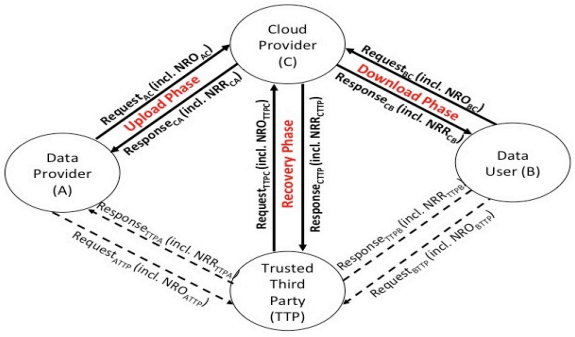

Figure 1. Non-Repudiation Protocol for Clouds

$A \rightarrow C$: $RQS_{AC} = \{f_{RQS_{AC}}, l, A, C, TTP, H(M), H(B_L), Seq_1, T_{g1}, T_1, EG_B(NRO_{AB}), E_C(NRO_{AC})\}$

Where:

- T_1 *is the maximum time that the sender will wait for an NRR to* RQS_{AC}.
- T_{g1} *is the time of the generation of* RQS_{AC}.
- NRO_{AB} *is an NRO sent from A to B users through C. It is visible to all* B_L *recipients, but not to C itself.* $NRO_{AB} = \{K, l, S_A(H(M))\}$
- NRO_{AC} *is an NRO sent from A to C, defined as* $NRO_{AC}=\{S_A(H(M),H(B_L),EG_B(NRO_{AB}),H(l,Seq_1,T_{g1}, T_1))\}$.

When C receives a RQS_{AC} it must produce a response to A. This step is defined as:

$C \rightarrow A$: $RSP_{CA} = \{f_{RSP_{CA}}, l, A, C, TTP, H(M), H(B_L), Seq_2, T_{g2}, T_S, E_A(NRR_{CA})\}$

Where:

- T_S *is time when data is stored*
- T_{g2} *is the time of the generation of* RSP_{AC}.
- NRR_{CA} *is the NRR sent from C to A, defined as* $NRR_{CA} = \{S_C(H(M)), S_C(H(l, Seq_2, T_{g2}, T_S, NRO_{AC}))\}$.

B. Download Phase

In this phase, the data user B downloads data from the cloud provider C. To do so, B sends a request with an NRO_{BC} to C. The request should include B's identity to enable C verify whether the B is authorised to download the requested data. This is done by checking B's identity against the B_L received for M from the data owner A. If B is in B_L, C will send the

requested data along with the encrypted NRO_{AB} ($EG_B(NRO_{AB})$) received from A and its own non-repudiation evidence NRR_{CB}. These exchanges are defined below:

$B_i \rightarrow C$: $RQS_{BiC} = \{f_{RQS_{BiC}}, l_i, A, C, B_i, TTP, Seq_3, T_{g3}, T_2, E_C(NRO_{BC})\}$

Where:

- $l_i = H(A, C, B_i, TTP)$
- NRO_{BC} *is the NRO sent from B to C, defined as* $NRO_{BC} = \{S_B(H(l_i, A, C, TTP, Seq_3, T_{g3}, T_2))\}$.

$C \rightarrow B_i$: $RSP_{CBi} = \{f_{RSP_{CBi}}, l, A, C, B_i, TTP, H(M), Seq_4, T_{g4}, EG_B(NRO_{AB}), E_{B_i}(NRR_{CB})\}$

Where:

- NRR_{CB} *is the NRR sent from C to B, defined as* $NRR_{CA}=\{S_C(H(M)),S_C(EG_B(NRO_{AB})),S_C(H(l,Seq_4,T_{g4}))\}$.

When B gets the data and the EG_B from C, it will obtain K and H(Data) by decrypting the NRO_{AB} and check the integrity of the data and the validity of NRR_{CB}.

C. Resolution of Disputation

If A does not receive the expected response from the C, it sends a request to TTP with its identification and the NRO_{AC}. TTP will subsequently send this request to C and C should respond with a corresponding NRR_{CA}. The latter exchanges are defined as:

$TTP \rightarrow C$: $RQS_{TC} = \{f_{RQS_{TC}}, l, A, C, TTP, Seq_5, T_{g5}, T_3, E_C(NRO_{AC}), E_C(NRO_{TC})\}$

$C \rightarrow TTP$: $RSP_{CT} = \{f_{RSP_{CT}}, l, A, C, TTP, Seq_6, T_{g6}, T_S, E_A(NRR_{CA}), E_T(NRR_{CT})\}$

Where:

- T_3 *is the maximum time that the sender will wait for an NRR to* RQS_{TC}.
- T_{g5} (T_{g6}) *is the time of the generation of* RQS_{TC} (RSP_{CT}).
- T_S *is the time when data was stored by C.*
- NRO_{TC} *is the NRO sent from TTP to C to resolve a disputation regarding an uploading session of A, defined as* $NRO_{TC}=\{S_T(H(l, A, C, TTP, Seq_5, Tg_5, T_5, E_C(NRO_{AC})))\}$.
- NRR_{CT} *is sent from C to TTP, defined as* $NRR_{CT} =\{S_C(H(l, Seq_5, T_{g5}, NRR_{CA}))\}$.

IV. Certification Model For Non-Repudiation

In CUMULUS, certificates may be generated on the basis of evidence gathered through continuous monitoring from the cloud provider. The cloud provider (i.e., the target of certification) and the security property to be certified, the extent of the monitoring evidence to be collected, and the process of certification are specified according to a monitoring based certification model (MBCM). This model drives the operation of the CUMULUS framework, which produces certificates that can are signed off by a certification authority that accepts MBCM either automatically or following some audit. In the following we present an MBCM for the NR property following an overview of the schema for specifying such models that is used by CUMULUS.

A. Monitoring based certification models: Background

A monitoring based certification model is specified in an XML based language whose top-level structure is shown in Figure 2. According to this schema, an MBCM specifies: (1) the cloud service to be certified (i.e., a *Target of Certification (ToC)*); (2) the *security property* to be certified for ToC; (3) the *certification authority* who will sign the certificates generated by the model; (4) an *assessment scheme* defining general conditions regarding the evidence that must be collected for being able to issue a certificate; (5) further validity tests regarding the configuration of the cloud provider and the CUMULUS framework itself that must be satisfied prior to issuing certificates; (6) the *monitoring configurations* that will be used in order to collect the evidence required for generating certificates; (7) the way in which the collected evidence will be aggregated in certificates (*evidence aggregation*); and (8) a *life cycle model* that defines the overall process of issuing certificates.

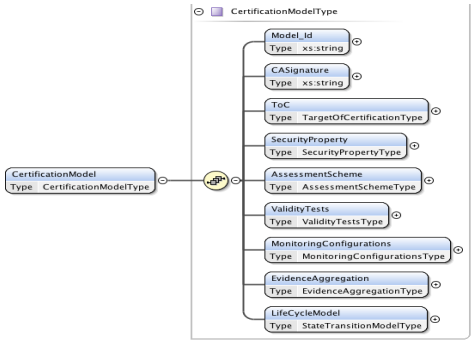

Figure 2. Monitoring Based Certification Model (MBCM) schema

In MBCMs, a ToC is specified as a concrete endpoint with a set of service interfaces that are offered by it to external parties (*provided interfaces*) and a set of interfaces required of external parties (*required interfaces*). The security property to be certified is specified by assertions expressed in EC-Assertion, i.e., an XML language based on Event Calculus[18].

The *assessment scheme* defines conditions regarding the evidence that must be collected in order to be able to issue a certificate. These conditions are related to (i) the sufficiency of the collected evidence, (ii) the expiration period for certificates, and (iii) anomalies and conflicts that should be monitored during the certification process. The evidence sufficiency conditions may relate to the minimum required period of time that the ToC should be monitored and the minimum number and representativeness of events (i.e., instances of ToC operations) that should be gathered before a certificate can be issued.

In an MBCM, anomalies refer to: (1) potential attacks on TOC, (2) other suspicious behaviour or (3) operational conditions related to the security property that is to be certified, which despite not having caused any violation of it, they may potentially affect its satisfiability and, therefore, lead to the suspension or revocation of certificate generated by the model. The definition of the potential "anomalies" that should be monitored as part of a certification model should be based on

an analysis of whether potential attacks, the ways in which the behaviour of different external actors that interact with TOC, and the overall operating conditions of the interaction between TOC and these actors may affect the satisfaction of the given security property by the TOC. Like security properties, anomalies are also specified as EC-Assertions, except that their violation does not lead automatically to the suspension/revocation of a certificate.

Conflicts aim to capture cases where a given security property would not be satisfied if it were to be assessed over different monitoring aggregation periods. The availability of a service may, for instance, be above 99% if assessed on a monthly basis by certification model whose security property refers to this period of assessment but be below this threshold if shorter/longer assessment intervals are considered. In an MBCM conflicts are defined by alternative assessment periods for the security property.

A life cycle model defines the process by which certificates of a given MBCM can be generated and managed (e.g., suspended, revoked). In an MBCM, a life cycle model (LCM) is defined by a state transition model expressed in XML, as shown in Figure 3.

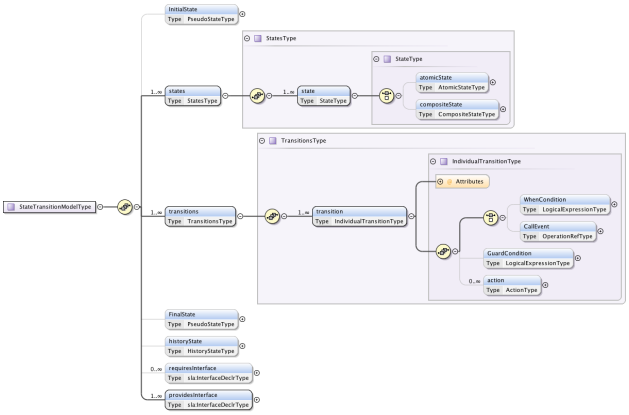

Figure 3. Life cycle models

In particular, a life cycle model is defined by a set of states and transitions between them. States can be composite or atomic. Composite states are refined into parallel or mutually exclusive substates. All state types can be associated with actions that are executed upon entry to or exit from the state, Transitions are associated by call events or triggering conditions (when-conditions). They can also be guarded by further conditions and be associated with actions that are executed when a transition is to be traversed and prior to arriving at the destination state. Actions correspond to invocations of operations in required and provided interfaces that are defined as part of an LTM. Provided interfaces include operations offered from the CUMULUS framework and required interfaces define operations of external tools.

Further details regarding the specification of MBCMs are available from [20].

V. CERTIFICATION MODEL FOR NR

A. Security property assertions for NR Protocol

In order to certify a cloud provider C for NR, we should monitor the responses (NRR) that C makes to NROs received from A, B and TTP during the different phases of the NR protocol shown in Figure 1. In particular, it is necessary to gather evidence demonstrating that, for every NRO that C receives from an NR party, C produces an NRR to it in a timely manner (i.e., without a delay that would make the relevant external party to be timed out).

In the case of NROs sent during the upload phase of the protocol, the assertion in the NR MBCM contains the monitoring rule R1 and the monitoring assumptions R1.A1-R1.A4 listed below (for readability purposes we specify all assertions in the high level syntax of EC-Assertion rather than its XML counterpart. A detailed account of EC-Assertion is given in [18]):

SECURITY PROPERTY MONITORING RULE:

R1: Happens(e(_id1,_A,_C, RQS_{AC},_C),$_t_{AReq}$,[$_t_{AReq}$, $_t_{AReq}$]) ∧ ¬ **HoldsAt(**ResUpReq(RQS_{AC},_X,_t), $_t_{AReq}$)⇒

Happens(e(_id2,_C,_A,RSP_{CA},_C),$_t_{g2}$,[$_t_{AReq}$,$_t_{AReq}$+f($_t_1$)])

WHERE:

RQS_{AC} = {$_f_{RQS_{AC}}$, _l, _A, _C, _TTP, _H(M), _H(B_L), $_Seq_1$, $_tg_1$, $_t_1$, _EG_B(K, l, S_A(H(M))), _E$_C$(S_A(H(M), H(B_L), _EG_B(K, l, S_A(H(M))), H(l, Seq_1, tg_1,t_1)))}

RSP_{CA} = {$_f_{RSP_{CA}}$, _l, _A, _C, _TTP, _H(M), _H(B_L), $_Seq_2$, $_tg_2$, $_t_s$, _E$_A$(S_C(H(M)), S_C(H(l, Seq_2, tg_2, t_s, (S_A(H(M), H(B_L),_EG_B(K, l, S_A(H(M))), H(l, Seq_1, tg_1, t_1)))))}

SECURITY PROPERTY ASSUMPTIONS:

R1.A1: Initially(UplReq(_C, 0, systime()))

R1.A2: Happens(e(_id1,_A,_C,RQS_{AC},_C),$_t_{AReq}$,[$_t_{AReq}$,$_t_{AReq}$]) ∧ ¬ **HoldsAt(**ResUplReq(RQS_{AC},_X, _t), $_t_{AReq}$) ∧

Happens(e(_id2, _C, _A, RSP_{CA}, _C),$_t_{g2}$, [$_t_{AReq}$, $_t_{AReq}$ + f($_t_1$)])∧ **HoldsAt**(UplReq(_C, _UPN, _ST), $_t_{AReq}$)) ⇒

Terminates(e(_id1, _A, _C, RQS_{AC}, _C), UplReq(_C, _UPN, _ST), $_t_{AReq}$) ∧ **Initiates(**e(_id1, _A, _C, RQS_{AC}, _C), UplReq(_C, _UPN+1, $_t_{g2}$), $_t_{g2}$) ∧

Initiates(e(_id1, _A, _C, RQS_{AC}, _C),
ResUplReq(RQS_{AC}, RSP_{CA}, $_t_{g2}$), $_t_{g2}$)

R1.A3: Happens(e(_id1,_A,_C,RQS_{AC},_C),$_t_{AReq}$,[$_t_{AReq}$, $_t_{AReq}$]) ∧¬ **HoldsAt(**ResUplReq(RQS_{AC},_X, _t), $_t_{AReq}$) ∧

¬ **Happens**(e(_id2, _C, _A, RSP_{CA}, _C), $_t_{g2}$, [$_t_{AReq}$, $_t_{AReq}$ + f($_t_1$)]) ⇒ **Initiates**(e(_id1, _A, _C, RQS_{AC}, _C), NoResUplReq(RQS_{AC}, $_t_{AReq}$), $_t_{AReq}$ + f($_t_1$))

R1.A4: Happens(e(_id1,_A,_C,RQS_{AC},_C),$_t_{AReq}$,[$_t_{AReq}$,$_t_{AReq}$]) ∧¬ **HoldsAt(**ResUplReq(RQS_{AC},_X, _t), $_t_{AReq}$) ∧

Happens(e(_id2, _C, _A, RSP_{CA}, _C), $_t_{g2}$, [$_t_{g2}$,$_t_{g2}$]) ∧ $_t_{g2}$ > $_t_{g1}$ + f($_t_1$) ⇒ **Initiates(**e(_id2, _C, _A, RQS_{AC}, C), LateUplReq(RQS_{AC}, RSP_{CA}, $_t_{g2}$), $_t_{g2}$)

Rule R1 checks if for every request (RQS_{AC}) made by a data owner (_A) for uploading data to a cloud provider (_C) at some time $_t_{AReq}$ (i.e., the time that the request was received by _C) and for which there is no previous request with the same sequence number received by _C from _A, _C sends a

response to _A acknowledging the request (RSP_{CA}) within at most f($\_t_1$) time units after $\_t_{AReq}$, where $\_t_1$ is the time that the data owner will wait for the response. f($\_t_1$) is a function that is provided by _C and should satisfy the constraint f($\_t_1$) < $\_t_1$ in order to minimise the likelihood of _A be timed out due to a delayed response from _C.

The certification model keeps also a record of:

(a) Requests (RQS_{AC}) for which a matching response (RSP_{CA}) was produced within the required time period and when no other previous request was made with the same sequence number,
(b) Requests for which no matching response was produced within the required time period,
(c) Requests that had the same sequence number with requests responded previously, and
(d) The total number of responded and non-responded requests made from A to C for uploading data.

To keep these records it uses the monitoring assumptions R1.A1–R1.A4. R1.A1 is used to initiate the fluent *UplReq(_C, _UPN, _ST)*, which keeps the total number of the responded requests between _A and _C (i.e., the value of the variable *_UPN* and *systime()* is a standard system call that is executed by the monitor to obtain the current time of the system where the monitoring service is running. R1.A2 updates the fluent *UplReq(_C, _UPN, _ST)* in order to increase the number of the successfully responded request for uploading data made from A to C *(_UPN+1)*. It also initiates the fluent $ResUplReq(RQS_{AC}, RSP_{CA}, \_t_{g2})$, in order to record details of the data upload request that was responded in time. The third assumption (R1.A3) keeps a record of data upload requests that were not responded in time, using the fluent $NoResUplReq(RQS_{AC}, \_t_{AReq})$. These correspond to violations of the monitoring rule R1. R1.A4 monitors and keeps a record of data upload requests that were responded but not within the required time limit. The assumption initiates the fluent $LateUplReq(RQS_{AC}, RSP_{CA}, \_t_{g2})$ to keep record of the requests responded with delay.

The records of responded, non-responded and not in time responded requests are used as detailed evidence by the certification framework, in order to be able to demonstrate the correctness of the protocol implementation by a cloud provider to relevant stakeholders and inform further analysis related to anomalies that may be detected (see Sect. V.B). These stakeholders could be: (a) a cloud provider who might have failed to obtain a certificate or a cloud provider who has obtained a certificate and wishes to provide detailed evidence about it, (b) a data producer who wishes to choose a cloud provider that is certified for the NR security property by checking the evidence, or (c) an auditor who can use these evidence for auditing purposes of a cloud provider.

Keeping these records is necessary as the information recorded can be used as evidence in the recovery phase, when a TTP sends a request to C for resolving a disputation between C and A. More specifically, if C receives a request from a TTP regarding a previous request RQS_{AC} from A to C, having this information the framework will be able to check whether a late response RSP_{CA} from C to A lead to the resolution phase.

The certification model for NR includes monitors also the responses that C provides to data download requests received from data users B. The monitoring rules and assumptions for this are similar to R1 and R1.A1-R1.A4, expect that that the monitor RQS_{BC} requests and RSP_{CB} responses.

The NR certification model monitors also the behaviour of C in response to requests by the trusted third party (TTP), in order to resolve disputations that might occur between C and A or B. For this purpose, it uses the following EC-Assertion formulae:

SECURITY PROPERTY MONITORING RULE

R2: **Happens**(e(_id1,_TTP,_C,RQS_{TC},_C),$\_t_{TReq}$,[$\_t_{TReq}$,$\_t_{TReq}$]) ∧ ¬ **HoldsAt**(ResTReq(RQS_{TC}, _X, $\_t_{g6}$), $\_t_{TReq}$) ⇒ **Happens**(e(_id2,_C,_TTP,RSP_{CT},_C),$\_t_{TRes}$,[$\_t_{TReq}$,$\_t_{TReq}$+f($\_t_3$)]

WHERE:

RQS_{TC} = {$\_f_{RQS_{TC}}$, _l, _A, _C, _TTP, $\_Seq_5$, $\_tg_5$, $\_t_3$, $\_E_C(S_A(H(M),H(B_L),\_EG_B(K,l,S_A(H(M))),H(l,Seq_1,tg_1,t_1))),E_C(S_T(H(l,A,C,TTP,Seq_5,tg_5,t_5,E_C(S_T(H(M),H(B_L),\_EG_B(K,l,S_A(H(M))),H(l,Seq_1,tg_1,t_1))))))$}

RSP_{CT} = {$\_f_{RSP_{CT}}$, _l, _A, _C, _TTP, $\_Seq_6$, $\_tg_6$, $\_t_5$, $\_E_A(S_C(H(M)),S_C(H(l, Seq_2, tg_2, t_5, (S_A(H(M),H(B_L),\_EG_B(K,l,S_A(H(M))),H(l,Seq_1,tg_1, t_1))))))$, $\_E_T(S_C(H(l, Seq_5,tg_5, (S_C(H(M)),S_C(H(l, Seq_2, Tg_2, T_5, (S_A(H(M), H(B_L),\_EG_B(K, l, S_A(H(M))),H(l,Seq_1,Tg_1, T_1))))))))$}

SECURITY PROPERTY ASSUMPTIONS

R2.A1: **Initially**(RecReq(_C, 0, systime()))

R2.A2: **Happens**(e(_id5, _TTP, _C, RQS_{TC}, _C),$\_t_{TTPReq}$,[$\_t_{TTPReq}$, $\_t_{TTPReq}$]) ∧ ¬ **HoldsAt**(ResRecReq(RQS_{TC},_X, _t),$\_t_{TReq}$) ∧ **Happens**(e(_id1,_A,_C, RQS_{AC}, _C), $\_t_{AReq}$, [0,$\_t_{TTPReq}$]) ∧ **Happens**(e(_id6, _C, _TTP, RSP_{CT}, _C), $\_t_{g6}$, [$\_t_{TReq}$, $\_t_{TReq}$ + f($\_t_3$)]) ∧ ∃ _RPN: **HoldsAt**(RecReq (_C, _RPN, _ST), $\_t_{TTPReq}$)) ⇒ **Terminates**(e(_id5, _TTP, _C, RQS_{TC}, _C), RecReq (_C, _ RPN, _ST), $\_t_{TTPReq}$) ∧ **Initiates**(e(_id5, _TTP, _C, RQS_{TC}, _C), RecReq (_C, _ RPN+1, _ST), $\_t_{TTPReq}$) ∧ **Initiates**(e(_id5, _TTP, _C, RQS_{TC}, _C), ResRecReq(RQS_{TC},_X, _t), $\_t_{TTPReq}$)

R2.A3: **Happens**(e(_id5, _TTP, _C, RQS_{TC}, _C), $\_t_{TTPReq}$, [$\_t_{TTPReq}$, $\_t_{TTPReq}$]) ∧ ¬ **HoldsAt**(ResRecReq RQS_{TC},_X, _t),$\_t_{TTPReq}$) ∧ **Happens**(e(_id1, _A, _C, RQS_{AC}, _C),$\_t_{AReq}$,[0,$\_t_{TTPReq}$]) ∧ ¬ **Happens**(e(_id6, _C, _TTP, RSP_{CT}, _C), $\_t_{g6}$, [$\_t_{TTPReq}$, $\_t_{TTPReq}$ + f($\_t_3$)]) ⇒ **Initiates**(e(_id5, _TTP, _C, RQS_{TC}, _C), NoResReq(RQS_{TC}, $\_t_{TTPReq}$), $\_t_{TTPReq}$ + f($\_t_3$))

R2.A4: **Happens**(e(_id5, _TTP, _C, RQS_{TC}, _C), $\_t_{TTPReq}$,[$\_t_{TTPReq}$,$\_t_{TTPReq}$]) ∧ ¬**HoldsAt**(ResRecReq RQS_{TC},_X, _t), $\_t_{TTPReq}$) ∧ **Happens**(e(_id6, _C, _TTP,RSP_{CT},_C),$\_t_{g6}$,[$\_t_{g6}$,$\_t_{g6}$]) ∧ $\_t_{g6}$> $\_t_{g5}$ + _f(t_3) ⇒ **Initiates**(e(_id5, _TTP, _C, RQS_{TC}, _C), LateRecReq RQS_{TC}, RSP_{CT}, $\_t_{g6}$), $\_t_{g6}$)

R2 checks if every request RQS_{TC} made from a TTP to a cloud provider (_C) for resolving a disputation between *A* and *C*, at some time ($\_t_{TReq}$), for which there was no previous request from the same TTP that has already been responded,

there is a response RSP_{CT} from *C* to the *TTP* that matches with RQS_{TC} within at most $f(\_t_3)$ time units where $f(\_t_3) < \_t_3$ ($\_t_3$ is the time that TTP will wait for a response).

As in the case of the interactions with the data owner, the certification model uses the following assumptions to keep record of (a) every request (RQS_{TC}) for which a matching response (RSP_{CT}) was produced within the required time period, (b) every request RQS_{TTPC} for which no matching response RSP_{CTTP} was produced within the required time period, (c) every request made with a same sequence number of a previous responded request, and (d) the total number of responded and non responded requests made from TTP to C for resolving a disputation between the other parties. To keep this record it uses the following monitoring assumptions:

B. Anomaly Detection

In the following we present anomalies of three different types introduced in Sect. IV.A for the NR certification model.

1) Potential attacks

In the case of NR, As and Bs may be non trusted parties. Both of them, for instance, may try to launch a denial-of-service attack on C. This may happen directly by, for example, issuing a high volume of data uploading and downloading requests to C and/or re-issuing previous requests (replay attack). It should be noted that the monitoring rule R1 in the certification model would require C to respond only to a request from a data provider only if this request has not been responded before. Hence, the certification model assumes that C should not respond to repeated requests. However, even if no response of C is expected in such cases, high volume of repetaed requests may escalate to a DOS attack that will prevent C from satisfy the NR property.

Hence the purpose of anomaly monitoring is not to detect the individual instances of repeated requests from A to C but to detect whether this unexpected activity appears in high volume. To monitor and keep a record of the repeated requests from particular data owners, the NR certification model should include the following anomaly detection monitoring assumptions:

ANOMALY ASSUMPTIONS

A.A1: Initially(RepeatedUplReq(_A, systime()))

A.A2: Happens(e(_id1,_A,_C,RQS_{AC},_C), $\_t_{AReq}$,[$\_t_{AReq}$, $\_t_{AReq}$]) ∧ **HoldsAt(**ResUpReq(RQS_{AC},_X,_t), $\_t_{AReq}$)⇒

Terminates(e(_id1, _A, _C, RQS_{AC}, _C), RepeatedUplReq(_A, _N), $\_t_x$) ∧

Initiates(e(_id1, _A, _C, RQS_{AC}, _C), RepeatedUplReq(_A, _N + 1), $\_t_x$)

The first of these assumptions initialises the counter of repeated requests from a given data owner _A to 0 and the second increases it whenever a new previously responded requests is re-played by _A.

Further to these anomaly-monitoring assumptions, the NR certification model can include a warning to the certification authority regarding the potential compromise of NR as soon as the number of repeated data upload requests exceeds a given threshold (i.e., N repeated data upload requests per minute). This is specified in the life cycle model of the NR certification model, as indicated in Figure 4 below.

To cover the potential of a similar type of attack from data users (B), the NR certification model includes also anomaly-monitoring assumptions similar to those listed above for data downloading requests RQS_{BC}.

2) Suspicious behaviour

An example of suspicious behaviour that the NR certification model should monitor is the receipt of requests for recovery from TTP corresponding to requests for data uploading (downloading) from A (B), which have been acknowledged by C. Such requests are suspicious since, in normal circumstances, TTP should not be asking for a recovery of a request that has been acknowledged by C (i.e., a request from A or B for which C has sent an NRR).

This anomalous behaviour from TTP may be due to different reasons. To issue a recovery request, TTP should know the details of the original data uploading (downloading) request from A (B). There are four different ways in which TTP can obtain this knowledge: (i) A or B might have sent the original request to TTP and ask it to initiate the recovery phase, (ii) an attacker, who has managed to obtain the details of the original request of A and B and impersonate them, sends it to TTP, or (iii) TTP has itself acted as an attacker (as in (ii)), obtained the details of the original request from A or B and sent the recovery request to C.

Case (i) itself may be the result of a malicious attempt to initiate the recovery phase by A or B. The reason for this could be, for example, to test via TTP how C and TTP would react to such non normal requests and whether it would be possible to launch some DoS attack onto C (via TTP) or onto TTP itself. However, (i) can also be the result of A or B being timed out due to a (non malicious) delay in the arrival of the $NRR_{A(B)}$ sent to them by C caused by the network connection between A(B) and C.

To monitor requests for recovery from TTP corresponding to already responded requests for data uploading from A, the NR certification model uses the following monitoring assumption:

ANOMALY ASSUMPTIONS

TTP.A1:

Happens(e(_id1, _TTP, _C, RQS_{TC}, _C), $\_t_{TReq}$, [$\_t_{TReq}$, $\_t_{TReq}$]) ∧ **HoldsAt(**ResUpReq(RQS_{AC},_X,_t), $\_t_{TReq}$) ⇒

Initiates(e(_id1, _TTP, _C, RQS_{TC}, _C), SuspTTPReq(_TTP, RQS_{TC}, $\_t_{TReq}$), $\_t_{TReq}$)

WHERE

RQS_{TC} = {$\_f_{RQS_{TC}}$, _l, _A, _C, _TTP, $\_Seq_5$, $\_tg_5$, $\_t_3$, $\_E_C(S_A(H(M),H(B_L),\_EG_B(K,l,S_A(H(M))),H(l,Seq_1,tg_1,t_1))),E_C(S_T(H(l,A,C,TTP,Seq_5,tg_5,t_5,E_C(S_T(H(M),H(B_L),\_EG_B(K,l,S_A(H(M))),H(l,Seq_1,tg_1,t_1))))))$}

RQS_{AC} = {$\_f_{RQS_{AC}}$, _l, _A, _C, _TTP, _H(M), _H(B_L), $\_Seq_1$, $\_tg_1$, $\_t_1$, $\_EG_B(K, l, S_A(H(M)))$, $\_E_C(S_A(H(M), H(B_L), \_EG_B(K, l, S_A(H(M))), H(l, Seq_1, tg_1,t_1)))$}

3) (Anomalous) Operating conditions

An example of an operating condition that should be monitored by the NR certification model is the average time that it takes for a response from C to reach its intended recipient party (i.e., A, B or TTP). Monitoring this time is important as it might indicate that responses to A, B or TTP reach them with delays that can get them timed out, despite C having issued these responses within the time period required by the NR protocol (i.e., within the period $[\_t_{AReq}, \_t_{AReq}+f(\_t_1)]$ required by rule R1 in the case of NRRs from C to A). Such delays might be due to network delays or some man-in-the-middle attack on the communication C and A, B and TTP.

Monitoring the exact average time of the arrival of a NRR from C to A, B or TTP is not possible as in general the monitoring framework of the certification authority that realises the NR certification model does not have access to events occurring at A and B. An approximate estimate of this average time is, however, possible by monitoring the average time of network traffic in the opposite direction, i.e., the average time that it takes for RQS_{AC}, RQS_{BC} and RQS_{TC} to reach C after being dispatched by A, B or TTP.

The following anomaly monitoring assumptions show how the NR certification model monitors the network delay for traffic from A to C:

ANOMALY ASSUMPTIONS

A.A3: Initially(avgAC(_A,_C, 0,0), systime())

A.A4: Happens(e(_id1,_A,_C, RQS_{AC},_C),$\_t_{AReq}$,[$\_t_{AReq}$, $\_t_{AReq}$]) ∧ **HoldsAt**(avgAC(_A,_C, _avg,_N), $\_t_{AReq}$) ⇒
Terminates(e(_id1,_A,_C, RQS_{AC},_C), avgAC(_A,_C, _avg,_N),$\_t_{AReq}$) ∧
Initiates(e(_id1,_A,_C, RQS_{AC},_C), avgAC(_A,_C,(_avg*_N + ($\_t_{AReq}$ −$\_t_{g1}$))/(_N+1),(_N+1)), $\_t_{AReq}$)

The anomaly assumption A.A3 initiates the fluent *avgAC(_A,_C, _avg,_N)* that is used to keep a record of the average time that it takes for data upload requests RQS_{AC} to reach C from A; _avg is the variable keeping the average value and _N is the variable keeping the number of requests that have been taken into account for calculating this average. The second formula updates this fluent by re-calculating the values of *_avg* to take into account the travelling time of the last RQS_{AC} , i.e., $\_t_{AReq} - \_t_{g1}$ ($\_t_{g1}$ is recorded in RQS_{AC} as shown in the specification of rule R1). In this case, the certification model should raise a warning to the certification authority in cases where _avg > f(t1), as this would lead to A being systematically timed out due to delays in the network traffic between A and C. This warning is also shown in the NR certificate life cycle model in Sect. IV.3. The NR certification model includes also anomaly-monitoring assumptions similar to Anomaly.A.A3 and Anomaly.A.A4 for Bs and TTPs.

C. Life Cycle model

The life-cycle model of the NR certification model is shown in Figure 4. According to this model, the first state in the certificate's lifecycle is called "Activated", where the certification process i activated. After being activated, the certificate process moves to the state *Continuous Monitoring*. Whilst being in this state, the security property and anomaly detection monitoring rules and assumptions of the model are being monitored (by the monitor of the CUMULUS platform) and the related monitoring evidence is sent to the framework (see transition *evidence(e:MonResult)*). When the accumulated evidence meets the sufficient conditions of the model and the security property monitoring rules are satisfied, the process moves to the state *Pre-Issued* (see transition *when(sec-assertion-satisfied AND sufficiency-conditions-satisfied)* from *D1* to *Pre-Issued*). At this state, the framework will check if the extra validity conditions for the certificate type (see action *CheckValidityConditions*) and, if they are satisfied, the process will move to the state *Issued*, at which a concrete certificate for NR of the specific provider is generated internally by the platform and can be obtained by an interested external party upon request (see transition *retrieveCertificate*).

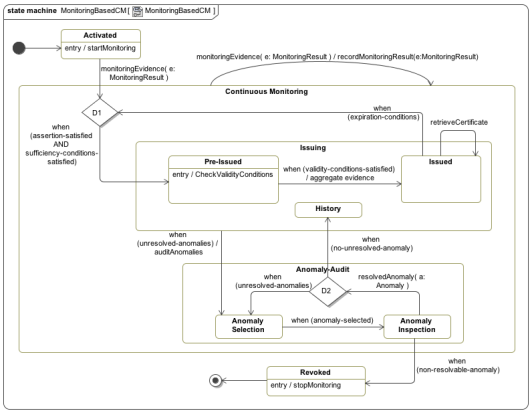

Figure 4. Monitoring-based CM: UML diagram of Life Cycle Model

Whilst in *Issuing* state, if an anomaly is detected, the certification process will move to the state *Anomaly-Audit*, (see transition *when(unresolved-anomalies)*) where all the detected anomalies must be selected (see state *AnomalySelection*) and inspected (see state *AnomalyInspection*) one by one. This is the responsibility of the certification authority that will sign off the certificates. If all the detected anomalies can be resolved, the process moves back to *History* state, i.e., the state where it was prior to moving to *Anomaly-Audit*. Otherwise, if there are anomalies that cannot be resolved (i.e., accepted as affordable risks), the process moves to the state *Revoke*, where any certificates issued for the particular TOC will be revoked and no further certificates will be issued.

When the expiration date of an issued certificate is reached, as stated in the *ExpirationCondition* of MBCM, the certification process will move to state *D1* (see the transition *when (expiration-conditions)*). At this point if a sufficient body evidence has already been accumulated for issuing a new instance of the certificate, the process will move automatically to the state Issuing or otherwise it will continue gathering evidence until a new certificate instance can be issued.

VI. CUMULUS FRAMEWORK

MBCMs are enacted by the CUMULUS framework to produce and manage certificates. The CUMULUS framework consists of a *certification communicator (CC)*, a *certificate*

generator (CG), a *monitoring manager (MM),* and a *certification models (MBCM),* an *evidence (EDB) and* a *certificate database (CeDB)*. It also interacts with *external monitors* (MON), as shown in Figure 5.

The monitoring manager (MM) is responsible for creating and modifying MBCMs according to the actors' requirements, and for managing the monitoring process for certifying a specific security property. Moreover, MM provides the monitoring configuration of an MBCM to MON, and polls the monitor at regular intervals in order to collect monitoring results. Once retrieved, these results are stored in EDB. The certification communicator (CC) allows external actors to retrieve generated (issued) certificates. CC retrieves such certificates from the *CeDB*. The certificate generator *(CG)* has responsibility for enacting MBCMs in order to generate and manage certificates. CG acts as an executor of the life cycle model of MBCM. During this execution it also access and updates information in EDB. When a certificate is generated, it is stored in the *CeDB*.

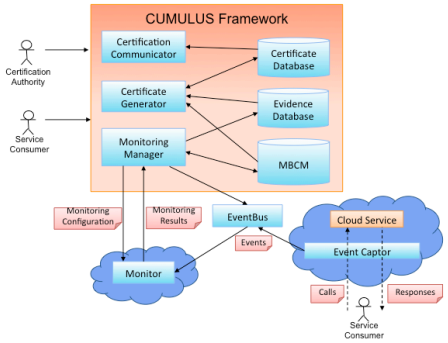

Figure 5. CUMULUS Framework

The *monitor (MON)* is responsible for monitoring the security property and anomaly assertions specified in MBCM. To do so, it is given these assertions by the framework and checks them against cloud event streams that are generated by *event captors* (EC) placed on cloud infrastructures. The communication between EC and MON is based on an event bus, which is implemented as a pub/sub infrastructure.

VII. Conclusion

In this paper, we have introduced a certification model (CM) for the NR protocol of cloud services that is based on continuous monitoring. The CM defines the conditions that should be monitored at runtime in order to confirm that a cloud provider adheres to the protocol and therefore offers the NR property. Furthermore, it defines conditions regarding the sufficiency of monitoring evidence for issuing certificates, anomalies that should be monitored during the certification, and the overall life cycle model (process) for generating NR certificates. Our model has been implemented using the CUMULUS certification framework.

Currently, we are conducting an evaluation of the performance of this model and investigate how to deploy model checking techniques in order to statically verify certain properties of it (e.g., soundness) and more generally of certification models following the CUMULUS approach, prior to putting them in operation. This work is exploring the use of model checking techniques for this purpose.

Acknowledgment

This work has been partially funded by the EU FP7 project CUMULUS [3] (grant no 318580).

References

[1] Asokan, N., Schunter, M. and Waidner, M., "Optimistic protocols for fair exchange", 4th ACM Conf. on Computier and Communications Security, vol. 6, pp. 8-17, 1997.

[2] Coffey, T. and Saidha, P., "Non-Repudiation protocol with mandatory proof of receipt", SIGCOMM Computer Communication, Review, 26(1):6-17, 1996.

[3] CUMULUS project, http://www.cumulus-project.eu/

[4] Feng, J., Chen, Y. and Summerville, D. H., "A Fair Multi-Party Non-Repudiation Scheme for Storage Clouds", Int. Conf. on Collaboration Technologies and Systems (CTS), pp. 457-465, May 2011.

[5] Feng, J., Chen, Y., Ku, W. and Liu, P., "Analysis of Intergrity Vulnerabilities and a Non-Repudiation Protocol for Cloud Data Storage Platrforms", 39th Int. Conf. on Parallel Processing Workshops, 2010.

[6] Ferdowsi, A., S3 data corruption?, 2008, available from https://forums.aws.amazon.com/thread.jspa?start=0&threadID=22709&tstart=0, 2008

[7] Gurgens, S., Rudolph, C. and Vogt, H., "On the Security of Fair Non-Repudiation Protocols", Int. Journal of Information Security, 4(4): 253-262, 2005.

[8] Kremer, S. and Markowitch, O., "Optimistic non-reputiable information exchange", In: Biemond, J. (Ed), 21st Symposium on Information Theory in the Benelux, Wassenaar (NL), pp. 126-132, 2000.

[9] Kremer, S., Markowitch, O., Zhou, J., "An intensive survey of fair non-repudiation protocols", Journal of Computer Communications 25:17, pp. 1606- 1621, 2002.

[10] Krotsiani, M., Spanoudakis, G. and Mahbub, K., "Incremental Certification of Cloud Services", 7th Int. Conf. on Emerging Security Information, Systems and Technologies, 2013.

[11] Ma., W., Wu, Q. and Tan, Y., "A TPA Based Efficient Non-Repudiation Scheme for Cloud Storage" 2nd Int. Conf. On Systems Engineering and Modeling (ICSEM -13), pp. 1630-1635, 2013.

[12] Markowitch, O. and Roggeman, Y., "Probabilistic Non-Repudiation without trusted third party" 2nd Conf. on Security in Communication Networks, 1999.

[13] Popa, R.A., Lorch, J. R., Molhan, D., Wang, H. and Zhuang, L., "Enabling Security in Cloud Storage SLAs with CloudProof", Microsoft Tech Report, 2010.

[14] Thomson, T., Google docs leaks private user data online, available from http://www.v3.co.uk/v3-uk/news/1951954/google-docs-leaks-private-user-online, 2009.

[15] Zhou, J. and Gollmann, D., "A Fair Non-Repudiation Protocol" In Proce. of IEEE Symposium on Security and Provacy, pp. 55-61, 1996.

[16] Zhou, J. and Gollmann, D., "An Efficient Non-Repudiation Protocol", 10th Computer Security Foundations Workshop, pp. 126-132, 1997.

[17] Common Criteria, v3.1, Parts 1, 2 and 3, 2012.

[18] Open Certification Framework, Vision Statement, Rev.1, Cloud Security Alliance. Available from https://downloads.cloudsecurityalliance.org/initiatives/ocf/OCF_Vision_Statement_Final.pdf, August 2013.

[19] Spanoudakis, G., Kloukinas, C., & Mahbub, K. "The SERENITY runtime monitoring framework". In *Security and Dependability for Ambient Intelligence*, pp. 213-237, Springer US. 2009.

[20] CUMULUS Deliverable D2.3, "Certification Models v2", May 2014. Available from: http://www.cumulus-project.eu/